Embedded Packet Capture

Introduction to Embedded Packet Capture

Packet capture is a fundamental network diagnostic technique used to intercept and record data packets as they traverse a network. These packets contain the raw information exchanged between devices, including protocol headers and payload data. By capturing and analyzing these packets, network engineers gain visibility into network behavior, enabling troubleshooting of connectivity issues, performance bottlenecks, and security incidents.

Packet Data Capture

This is the process of capturing data packets, which are then stored in a buffer for analysis. You can create packet captures by assigning unique names and defining specific parameters.

The following actions can be performed:

- Activating captures on any interface.
- Applying access control lists (ACLs) or class maps to capture points.
- Destroying captures when they are no longer needed.
- Specifying buffer storage parameters, such as size and type. Buffer size can range from 1 MB to 100 MB. The default buffer type is linear, with circular buffering available as an alternative.
- Defining match criteria based on protocol, IP address, or port number to filter the captured traffic.

Packet capture is essential for understanding complex network interactions and validating protocol operations. Traditionally, packet captures were performed using external devices such as network taps or span/mirror ports, which duplicate traffic to an analysis tool.

The packet capture can be used locally or exported for offline analysis using tools such as Embedded Packet Capture (EPC) and Wireshark. EPC refers to Cisco’s on-device packet capture feature that enables capturing

and filtering packets directly on devices without requiring external hardware, while Wireshark is a powerful, open-source packet analyzer widely used for in-depth inspection and troubleshooting of network traffic.

What is Embedded Packet Capture?

Embedded Packet Capture (EPC) is a Cisco feature that enables packet capture directly on the network device without the need for external hardware or port mirroring. EPC leverages the device's internal resources to capture traffic on specified interfaces, store the data temporarily in onboard buffers or files, and then export the captured packets for analysis.

Advantages of EPC include:

On-device capture: No additional hardware is needed.

Granular filtering: Capture specific traffic types or flows.

Low impact: Efficient resource use to minimize device performance degradation.

Flexibility: Capture on physical interfaces or other logical interfaces.

Benefits of Embedded Packet Capture

- The device can capture IPv4 and IPv6 packets, as well as non-IP packets using a MAC filter or by matching any MAC address.
- Extensible infrastructure for enabling packet capture points. A capture point is a traffic transit location used for capturing packets and associating them with a buffer.
- The packet capture can be exported in a packet capture file (PCAP) format. This format is suitable for analysis using any external tool.
- Methods to decode data packets captured with varying degrees of detail.

Prerequisites for Configuring Embedded Packet Capture

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

Table 1: System requirements for the EPC subsystem

System resources	Requirements
Hardware	CPU utilization requirements are platform-dependent.
Memory	The DRAM stores the packet buffer. The size of the packet buffer is user specified.
Disk space	Packets can be exported to external devices. No intermediate storage on flash disk is required.

Restrictions for Configuring Embedded Packet Capture

The following restrictions apply to Embedded Packet Capture (EPC):

- You cannot use VRFs, management ports, or private VLANs as attachment points.
- A VLAN interface that is in shutdown state does not support EPC.
- If you change an interface from switch port to routed port (Layer 2 to Layer 3), or vice versa, you must delete the capture point and create a new one once the interface comes back up. Stopping and starting 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 including TTL, VLAN tag, CoS, checksum, MAC addresses, DSCP, precedent, and UP.
- 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 for both ingress and egress packets is not captured.
- CPU-injected packets are considered control plane packets, and these types of packets will not be captured on an interface egress capture.
- Control plane packets are not rate limited and impact performance. Use filters to limit control plane packet capture.
- DNA Advantage supports decoding of protocols such as Control and Provisioning of Wireless Access Points (CAPWAP).
- You can define up to eight capture points, but only one can be active at a time. Stop one before 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 won't be supported on a Layer 3 port or SVI.
- MAC filter cannot capture Layer 2 packets (ARP) on Layer 3 interfaces.
- VACL does not support IPv6-based ACLs.
- EPC cannot capture based on the underlying routing protocols in MPLS packets.
- EPC is not supported on Locator/ID Separation Protocol (LISP) interface and tunnel interface.
- EPC is not supported with Ethernet-over-MPLS (EoMPLS).
- Network Based Application Recognition (NBAR) and MAC-style class maps are not supported.

What is Wireshark?

Wireshark is a widely adopted, open-source packet analysis tool used to examine network captures in detail. While EPC captures traffic on the switch itself, Wireshark provides an environment to analyze that data on a PC or workstation.

The typical workflow involves:

1. **Capture:** EPC collects packets on the device according to configured filters and interfaces.
2. **Export:** The captured packets are exported from the switch as `.pcap` files using protocols.
3. **Analysis:** These capture files are opened in Wireshark, where you can apply protocol decodes, filters, and visualizations to pinpoint network issues or validate behavior.

This integrated process combines Cisco's embedded capture capability with Wireshark's analysis tools, enabling comprehensive network troubleshooting without requiring dedicated capture appliances.

How Does Wireshark Work?

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.

The following sections describe the components involved in Wireshark:

Capture Points

A capture point is the central policy definition of the Wireshark feature. It outlines the characteristics of a specific Wireshark instance, such as the packets to capture, their sources, the actions to take with the captured packets, and the stopping conditions. Capture points can be modified after creation but remain inactive until explicitly activated with a **start** command. This process is called activating or starting the capture point. Capture points are identified by name and can be deactivated or stopped manually or automatically.

Multiple 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.

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

Attachment Points

An attachment point is a point in the logical packet process path associated with a capture point. An attachment point, which is an attribute of the capture point, is tested against capture point filters.

Packets that match the filters are copied and sent to the associated Wireshark instance. A specific capture point can associate with multiple attachment points, but it is limited in mixing attachment points of different types. Some restrictions apply when you specify attachment points of different types. Attachment points are directional (input or output or both) with the exception of the Layer 2 VLAN attachment point, which is always bidirectional.

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

Filters

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

A capture point has the following types of filters:

- Core system filter—The core system filter is applied by hardware, and its match criteria is limited by hardware. This filter determines whether to copy the hardware-forwarded traffic to software for Wireshark purposes.
- Capture filter—Wireshark applies the capture filter. The match criteria are more granular than those supported by the core system filter. Packets that pass the core filter but fail the capture filter are copied. They are sent to the CPU/software, but are discarded by the Wireshark process. The capture filter syntax matches that of the display filter.
- Display filter—Wireshark applies the display filter. Its match criteria are similar to the criteria of the capture filter. Packets that fail the display filter aren't displayed.

Note Wireshark does not use the syntax of the capture filter.

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, obtaining authorization to modify the device configuration may lead to significant delays if the approval process is lengthened. 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.

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

Display Filter

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

Actions

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

- Capture to buffer in memory to decode, analyze and store
- Store to a .pcap file
- Decode and display
- Store and display.

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

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

Storage of Captured Packets

Storage of Captured Packets to Buffer in Memory

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

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

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

Storage of Captured Packets to a .pcap File

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

Attempts to store packet captures on devices other than flash or USB flash devices connected to the active switch 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 (flash:)
- USB drive (usbflash0:)

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

Packet Decoding and Display

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

Note Decoding and displaying packets may be CPU intensive.

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

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

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

Live Traffic Display

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

.pcap File Display

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

Packet Storage and Display

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

Features of Wireshark

- If you apply port security and Wireshark on an ingress capture, a packet dropped by port security is still captured by Wireshark. If you apply port security on an ingress capture, and Wireshark on an egress capture, a packet that is dropped by port security is not be captured by Wireshark.
- Wireshark does not capture packets dropped by Dynamic ARP Inspection (DAI).
- f you use a port that is in STP blocked state as an attachment point and the core filter is matched, Wireshark captures the packets that come into the port, even though the packets are 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.

  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 captures the packet. Otherwise, Wireshark won't 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 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 the CPU. To capture these packets, include the control plane as an attachment point.
- VLANs—Starting with Cisco IOS Release 16.1, when you use a VLAN as a Wireshark attachment point, packet capture is supported on L2 and L3 in both input and output directions.
- Redirection features—In the input direction, features traffic redirected by Layer 3 (such as PBR and WCCP) are logically later than Layer 3 Wireshark attachment points. Wireshark captures these packets even though they might later be redirected out another Layer 3 interface. Symmetrically, output features redirected by Layer 3 (such as egress WCCP) are logically prior to Layer 3 Wireshark attachment points, and Wireshark won't capture them.
- SPAN—Wireshark 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 than 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. When performing a Wireshark packet capture, packets are copied and sent to the CPU, which increases CPU usage. .
- You might experience high CPU (or memory) usage if:
  - You leave a capture session enabled and unattended for a long period, resulting in unanticipated bursts of traffic.
  - You launch a capture session with ring files or capture buffer and leave it unattended for a long time, resulting in performance or system health issues.
- To minimize high CPU usage, follow these steps:
  - 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.
  - Closely follow the filter rules. Restrict the traffic type (for example, IPv4 only) with a strict ACL instead of a relaxed ACL, which can attract unwanted traffic.
  - When using Wireshark to capture live traffic, consider applying a QoS policy temporarily to limit the actual traffic until the capture process concludes.
- Always limit packet capture to either a shorter duration or a smaller packet number. The parameters of the capture command enable you to specify the following:
  - Capture duration
  - Number of packets captured
  - File size
  - Packet segment size
- During a capture session, watch for high CPU usage and memory consumption due to Wireshark that may impact device performance or health. If these situations arise, stop the Wireshark session immediately.
- Run a capture session without limits if you know that little traffic matches the core filter.
- You can define 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. However, only one of the instances can be active.
- When you modify an Access Control List (ACL) that is associated with a running capture, rrestart the capture to apply the changes. Otherwise, it continues to use the original ACL as if it has not been modified.
- Writing to the flash disk is a CPU-intensive operation. If the capture rate is insufficient, consider using a buffer capture.
- Avoid decoding and displaying packets from a .pcap file for a large file. Instead, transfer the .pcap file to a PC and run Wireshark on the PC.

- If you plan to store packets to a storage file, ensure that sufficient space is available before beginning a Wireshark capture process.
- To avoid packet loss, consider the following:
  - Use store-only (when you don't specify the display option) while capturing live packets. Do not use it for decode and display, which is an CPU-intensive operation (especially in detailed mode).
  - If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss.
  - If you use the default buffer size and see that you're losing packets, you can increase the buffer size to avoid losing packets.
- If you want to decode and display live packets in the console window, ensure that you bound the Wireshark session by a short capture duration.
- The core filter can be an explicit filter, access list, or class map. Specifying a newer filter of these types replaces the existing one.

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's 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 GigabitEthernet1/0/1 in** where GigabitEthernet1/0/1 is an attachment point. If you also need to attach interface GigabitEthernet1/0/2, enter it as **monitor capture mycap interface GigabitEthernet1/0/2 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.
- If the file during creation of the capture point, Wireshark asks you if the file can be overwritten. If the file exists at the time of activating the capture point, Wireshark overwrites the existing file.
- You can terminate a Wireshark session with an explicit **stop** command or by entering **q** in automore mode. The session could terminate itself automatically when it meets a stop condition such as duration or packet capture limit is met. It can terminate if an internal error occurs, or resource is full (specifically if disk is full in file mode).
- Dropped packets won't be shown at the end of the capture. However, only the count of dropped and oversized packets are displayed.

Prerequisites for Configuring Wireshark

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

Restrictions for Configuring Wireshark

- Wireshark does not support global packet capture.
- Wireshark does not support limiting circular file storage by file size.
- If you delete the file used by an active capture session, the capture session cannot create a new file. All further packets captured are lost. You have to restart the capture point.
- File limit is limited to the size of the flash in .
- 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, restart the capture manually.
- The streaming capture mode supports approximately 1000 pps; lock-step mode supports approximately 2 Mbps (measured with 256-byte packets). When the matching traffic rate exceeds this number, you may experience packet loss.
- You cannot change a capture point when the capture is active.
- Wireshark does not capture packets dropped by floodblock.
- A Wireshark class map allows only one ACL (IPv4, IPv6, or MAC).
- ACL logging and Wireshark are incompatible. Once you activate Wireshark, it takes priority. All traffic, including that captured by ACL logging on any ports, is redirected to Wireshark. We recommended that you deactivate ACL logging before starting Wireshark. Otherwise, Wireshark traffic will be contaminated by ACL logging traffic.
- If you capture both PACL and RACL on the same port, only one copy is sent to the CPU. If you capture a DTLS-encrypted CAPWAP interface, two copies are sent to Wireshark, one encrypted and the other decrypted. The same behavior occurs if you capture a Layer 2 interface carrying DTLS-encrypted CAPWAP traffic. The core filter is based on the outer CAPWAP header.
- The CLI for configuring Wireshark requires that the feature is executed only from EXEC mode. Actions that usually occur in configuration submode (such as defining capture points), are handled at the EXEC mode instead. All key commands are not NVGEN'd and 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 noy 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.
- Packet length range as a filter for packet capture is not supported for non- IPv4/IPv6 packets and fragmented packets.
- Packet length range as a filter cannot be used with any other filters.

How to Configure Packet Capture

The following sections provide information on configuring packet capture.

Managing Packet Data Capture

Note You can export the active capture point only after stopping the active capture.

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

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **monitor capture** *capture-name* **access-list** *access-list-name*

Example:

```
Device# monitor capture mycap access-list v4acl
```

Configures a monitor capture specifying an access list as the core filter for the packet capture.

Step 3 **monitor capture** *capture-name* **limit duration** *seconds*

Example:

```
Device# monitor capture mycap limit duration 1000
```

Configures monitor capture limits.

Step 4 **monitor capture** *capture-name* **interface** *interface-name* **both**

Example:

```
Device# monitor capture mycap interface GigabitEthernet 0/0/1 both
```

Configures monitor capture specifying an attachment point and the packet flow direction.

Step 5 **monitor capture** *capture-name* **buffer circular size** *bytes*

Example:

```
Device# monitor capture mycap buffer circular size 10
```

Configures a buffer to capture packet data.

Step 6 **monitor capture** *capture-name* **start**

Example:

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

Starts the capture of packet data at a traffic trace point into a buffer.

Step 7 **monitor capture** *capture-name* **stop**

Example:

```
Device# monitor capture mycap stop
```

Stops the capture of packet data at a traffic trace point.

Step 8 **monitor capture** *capture-name* **export** *file-location/file-name*

Example:

```
Device# monitor capture mycap export tftp://10.1.88.9/mycap.pcap
```

Exports captured data for analysis.

Step 9 **end**

Example:

```
Device# end
```

Returns to privileged EXEC mode.

Monitoring and Maintaining Captured Data

Perform this task to monitor and maintain the packet data captured. The details of the capture buffer and capture point are displayed during the procedure.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **show monitor capture** *capture-buffer-name* **buffer dump**

Example:

```
Device# show monitor capture mycap buffer dump
```

(Optional) Displays a hexadecimal dump of captured packet and its metadata.

Step 3 **show monitor capture** *capture-buffer-name* **parameter**

Example:

```
Device# show monitor capture mycap parameter
```

(Optional) Displays a list of commands that were used to specify the capture.

Step 4 **debug epc capture-point**

Example:

```
Device# debug epc capture-point
```

(Optional) Enables packet capture point debugging.

Step 5 **debug epc provision**

Example:

```
Device# debug epc provision
```

(Optional) Enables packet capture provisioning debugging.

Step 6 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

How to Configure Wireshark

To configure Wireshark, complete these basic steps:

1. Define a capture point.
2. Add or modify parameters for the capture point.
3. Activate or deactivate a capture point.

4. Delete the capture point when it is no longer needed.

Defining a Capture Point

The example in this procedure defines a simple capture point. Optionally, you can define all parameters for the capture point through with one **monitor capture** command.

Note To have a functional capture point, define an attachment point, specify the direction of capture, and configure the core filter.

You do not need to define a core filter when creating a wireless capture point using a CAPWAP tunneling interface.

To define a capture point, use these steps.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **monitor capture** {*capture-name*} {**interface** *interface-type interface-id* | **control-plane**} {**in** | **out** | **both**}

Example:

```
Device# monitor capture mycap interface GigabitEthernet1/0/1 in
```

Defines the capture point, specifies the attachment point with which the capture point is associated, and specifies the direction of the capture.

The keywords have these meanings:

- *capture-name*—Specifies the name of the capture point to be defined (mycap is used in the example). Capture Name should be less than or equal to eight characters. Only alphanumeric characters and underscore (_) is permitted.
- (Optional) **interface** *interface-type interface-id*—Specifies the attachment point with which the capture point is associated (GigabitEthernet1/0/1 is used in the example).

  Note
  Optionally, you can define multiple attachment points and all the parameters for this capture point with this one command instance. These parameters are discussed in the instructions for modifying capture point parameters. Range support is also available both for adding and removing attachment points.

  Use one of these options for *interface-type*:

  - **AppGigabitEthernet**—Specifies the attachment point as AppGigabitEthernet.

- **GigabitEthernet**—Specifies the attachment point as GigabitEthernet.
- **vlan**—Specifies the attachment point as a VLAN.

  Note
  Only ingress capture (in) is allowed when using this interface as an attachment point.

- **capwap**—Specifies the attachment point as a CAPWAP tunnel.

  Note
  When using this interface as an attachment point, you cannot use a core filter.

- (Optional) **control-plane**—Specifies the control plane as an attachment point.
- **in** | **out** | **both**—Specifies the direction of capture.

Step 3 **monitor capture** {*capture-name*} [**match** {**any** | **ipv4 any any** | **ipv6**} **any any**}]

Example:

```
Device# monitor capture mycap interface GigabitEthernet1/0/1 in match any
```

Defines the core system filter.

Note
When using the CAPWAP tunneling interface as an attachment point, don't perform this step because a core filter cannot be used.

The keywords have these meanings:

- *capture-name*—Specifies the name of the capture point to be defined (mycap is used in the example).
- **match**—Specifies a filter. The first filter defined is the core filter.

  Note
  If a capture point does not have a core system filter or attachment points defined, it cannot be activated. Ensure the capture point meets all requirements and avoids errors before activation.

- **ipv4**—Specifies an IP version 4 filter.
- **ipv6**—Specifies an IP version 6 filter.

Step 4 **show monitor capture** {*capture-name*} [**parameter**]

Example:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 in
   monitor capture mycap match any
```

Displays the capture point parameters defined in Step 2 and confirms that you defined a capture point.

Step 5 **show 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

Add more attachment points, modify the capture point parameters, and activate it. If you want to use your capture point just as it is, you can now activate it.

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

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

Adding or Monitoring Capture Point Parameters

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

Use these steps to modify the parameters of a capture point:

Before you begin

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

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **monitor capture** {*capture-name*} **match** {**any** | **mac** *mac-match-string* | **ipv4** {**any** | **host** | **protocol**} {**any** | **host**} | **ipv6** {**any** | **host** | **protocol**} {**any** | **host**}}

Example:

```
Device# monitor capture mycap match ipv4 any any
```

Defines the core system filter (**ipv4 any any**), defined either explicitly, through ACL or through a class map.

You can define the core system filter through an ACL. You can configure the Ethertype of a protocol in the ACL. You can configure the same ACL in Wireshark to enable the capture of packets with a specific Ethertype.

Step 3 **monitor capture** {*capture-name*} **limit** {[**duration** *seconds*] [**packet-length** *size*] [**packets** *num*]}

Example:

```
Device# monitor capture mycap limit duration 60 packet-len 400
```

Specifies the session limit in seconds (60), packets captured, or the packet segment length retained by Wireshark (400).

Step 4 **monitor capture** {*capture-name*} **file** {**location** *filename*}

Example:

```
Device# monitor capture mycap file location flash:mycap.pcap
```

Specifies the file association, if the capture point intends to capture packets rather than only display them.

Note
If the file exists, confirm if it can be overwritten.

Step 5 **monitor capture** {*capture-name*} **file** {**buffer-size** *size*}

Example:

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

Specifies the size of the memory buffer used by Wireshark to handle traffic bursts.

Step 6 **show monitor capture** {*capture-name*}[**parameter**]

Example:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap limit duration 60 packet-len 400
   monitor capture point mycap file location bootdisk:mycap.pcap
   monitor capture mycap file buffer-size 100
```

Displays the capture point parameters that are already defined.

Step 7 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Modifying Parameters

Associating or Disassociating a Capture File

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

Specifying a Memory Buffer Size for Packet Burst Handling

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

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

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

Specifying an ether type for packets

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

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

What to do next

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

Deleting Capture Point Parameters

The steps to delete parameters can be executed in any order, even though they are listed sequentially. You can also delete them in one, two, or several lines. All parameters can be deleted except for attachment points, which may include multiple entries.

To delete capture point parameters, perform the following steps:

Before you begin

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

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **no monitor capture** {*capture-name*} **match**

Example:

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

Deletes all filters defined on capture point (mycap).

Step 3 **no monitor capture** {*capture-name*} **limit** [**duration**][**packet-length**][**packets**]

Example:

```
Device# no monitor capture mycap limit duration packet-len
Device# no monitor capture mycap limit
```

Deletes the session time limit and the packet segment length retained by Wireshark. It leaves other specified limits in place.

Deletes all limits on Wireshark.

Step 4 **no monitor capture** {*capture-name*} **file** [**location**] [**buffer-size**]

Example:

```
Device# no monitor capture mycap file
Device# no monitor capture mycap file location
```

Deletes the file association. The capture point will no longer capture packets. It only displays them.

Deletes the file location association. The file location is no longer associated with the capture point. However, other defined file association is unaffected by this action.

Step 5 **show monitor capture** {*capture-name*}[**parameter**]

Example:

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 in
```

Displays the capture point parameters that remain defined after your parameter deletion operations. You can run this command at any point in the procedure to see what parameters are associated with a capture point.

Step 6 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

What to do next

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

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

Deleting a Capture Point

To delete a capture point, follow these steps:

Before you begin

Define a capture point before using these instructions to delete it. Stop the capture point before deleting it.

Procedure

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:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 5 **show running-config**

Example:

```
Device# show running-config
```

Verifies your entries.

Step 6 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

What to do next

Define a new capture point with the same name as the one you deleted. Perform these instructions to start over with defining a capture point.

Activating and Deactivating a Capture Point

Follow these steps to activate or deactivate a capture point.

Before you begin

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

You can activate a capture point without an associated filename for display purposes. If no filename is specified, the packets are captured in the buffer. The live display (display during capture) works in both file and buffer modes.

If no display filters are specified, packets are not displayed live. However, packets captured by the core system filter are displayed, and the default display mode is brief.

Note If a CAPWAP tunneling interface is used as an attachment point, core filters are not used. Therefore, defining them is unnecessary.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **monitor capture** {*capture-name*} **start** [**display** [**display-filter** *filter-string*]] [**brief** | **detailed** | **dump**]

Example:

```
Device# monitor capture mycap start display display-filter "stp"
```

Activates a capture point and filters the display, so it displays only packets containing "stp".

Step 3 **monitor capture** {*capture-name*} **stop**

Example:

```
Device# monitor capture name stop
```

Deactivates a capture point.

Step 4 **end**

Example:

```
Device(config)# end
```

Returns to privileged EXEC mode.

Step 5 **show running-config**

Example:

```
Device# show running-config
```

Verifies your entries.

Step 6 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

What to do next

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

The error 'Missing attachment point on activation' occurs when the attachment point is not defined.

```
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

To clear the buffer contents or save them to an external file for storage, follow these steps:

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. Do not try to clear buffer on an active capture point.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **monitor capture** {*capture-name*} [**clear** | **export** *filename*]

Example:

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

Clear - Completely deletes the buffer.

Note
When you run the clear command,

- 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)# end
```

Returns to privileged EXEC mode.

Step 4 **show running-config**

Example:

```
Device# show running-config
```

Verifies your entries.

Step 5 **copy running-config startup-config**

Example:

```
Device# copy running-config startup-config
```

(Optional) Saves your entries in the configuration file.

Examples: Capture Point Buffer Handling

Exporting Capture to a File

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

Storage configured as File for this capture
```

Clearing Capture Point Buffer

```
Device# monitor capture mycap clear

 Capture configured with file options
```

What to do next

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