Management Configuration Guide

First Published: 2026-02-26

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1 **Embedded Packet Capture** 1

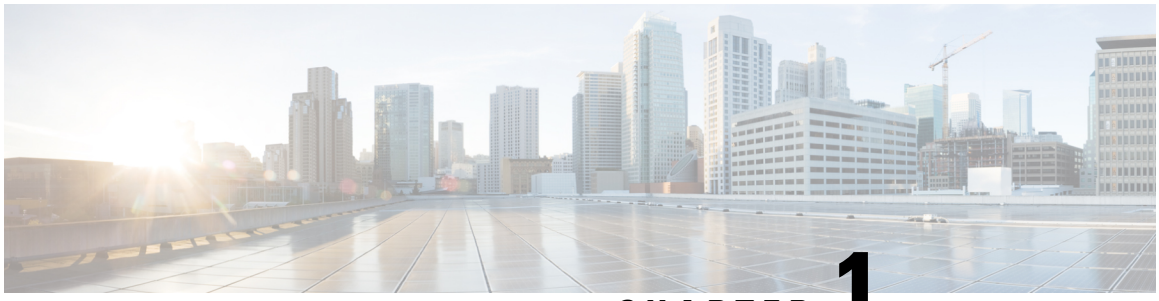
- Introduction to Embedded Packet Capture 1
- What is Embedded Packet Capture? 2
  - Prerequisites for Configuring Embedded Packet Capture 2
  - Restrictions for Configuring Embedded Packet Capture 3
- What is Wireshark? 4
  - How Does Wireshark Work? 4
  - Storage of Captured Packets 6
  - Features of Wireshark 8
  - Guidelines for Configuring Wireshark 9
  - Prerequisites for Configuring Wireshark 11
  - Restrictions for Configuring Wireshark 11
- How to Configure Packet Capture 12
  - Managing Packet Data Capture 12
  - Monitoring and Maintaining Captured Data 13
- How to Configure Wireshark 14
  - Defining a Capture Point 15
  - Adding or Monitoring Capture Point Parameters 19
  - Deleting Capture Point Parameters 21
  - Deleting a Capture Point 22
  - Activating and Deactivating a Capture Point 23
  - Clearing the Capture Point Buffer 26

CHAPTER 2 **Simple Network Management Protocol** 29

- Understanding SNMP 29
  - What is SNMP? 29

SNMP Versions 30
SNMP Manager Functions 31
SNMP Agent Functions 32
SNMP MIB Variables Access 32
SNMP Flash MIB 33
SNMP Notifications 33
SNMP ifIndex MIB Object Values 35
SNMP ENTITY-MIB Identifiers 35
SNMP and Syslog over IPv6 35
SNMP UDP ports 36
Default SNMP Configuration 36
Restrictions for SNMP 36
How to Configure SNMP 37
SNMP Configuration Guidelines 37
Configuring SNMP Groups and Users 38
Opening or Closing SNMP UDP Ports 43
Setting the Agent Contact and Location Information 44
Limiting TFTP Servers Used Through SNMP 45
Disabling the SNMP Agent 46
Monitoring SNMP Status 48
SNMP Examples 48

CHAPTER 3 **Switched Port Analyzer** 51
What is SPAN? 51
How SPAN works 51
SPAN Concepts and Terminology 52
SPAN Interaction with Other Features 56
SPAN and Device Stacks 57
Restrictions for SPAN 57
How to Configure SPAN 58
Creating a local SPAN session 58
Configuration Examples for SPAN 60



CHAPTER 1

Embedded Packet Capture

- Introduction to Embedded Packet Capture, on page 1
- What is Embedded Packet Capture?, on page 2
- What is Wireshark?, on page 4
- How to Configure Packet Capture, on page 12
- How to Configure Wireshark, on page 14

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
```

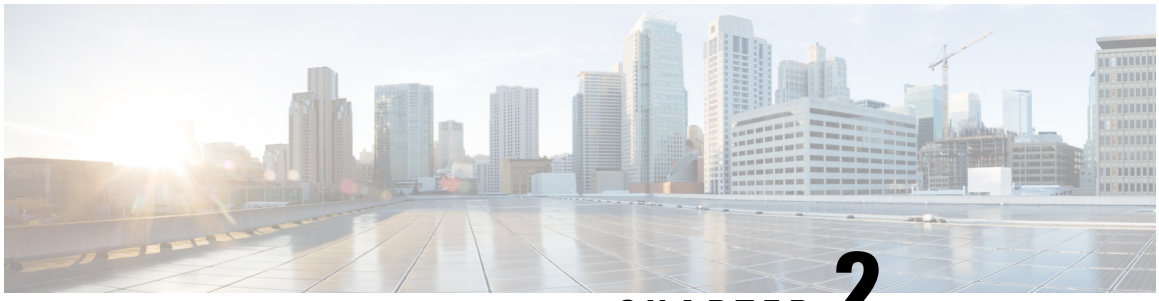
Clearing Capture Point Buffer

```
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*".



CHAPTER 2

Simple Network Management Protocol

- Understanding SNMP, on page 29
- SNMP Manager Functions, on page 31
- SNMP Agent Functions, on page 32
- SNMP MIB Variables Access, on page 32
- SNMP Flash MIB, on page 33
- SNMP Notifications, on page 33
- SNMP ifIndex MIB Object Values, on page 35
- SNMP ENTITY-MIB Identifiers, on page 35
- SNMP and Syslog over IPv6, on page 35
- SNMP UDP ports, on page 36
- Default SNMP Configuration, on page 36
- Restrictions for SNMP, on page 36
- How to Configure SNMP, on page 37
- SNMP Examples, on page 48

Understanding SNMP

SNMP (Simple Network Management Protocol) is a standard protocol used for monitoring and managing devices on IP networks.

What is SNMP?

Simple Network Management Protocol (SNMP) is an application-layer protocol for communication between managers and agents. An SNMP system consists of an SNMP manager, an SNMP agent, and a Management Information Base (MIB).

The SNMP manager can be part of a network management system (NMS), such as Cisco Prime Infrastructure. The agent and MIB reside on the device. To configure SNMP on the device, you define the relationship between the manager and the agent.

The SNMP agent contains MIB variables whose values the SNMP manager can request or change. A manager can get a value from an agent or store a value into the agent. The agent gathers data from the MIB, the repository for information about device parameters and network data. The agent can also respond to a manager's requests to get or set data.

An agent can send unsolicited traps to the manager. Traps are messages alerting the SNMP manager to a condition on the network. Traps can indicate improper user authentication, restarts, link status (up or down), MAC address tracking, closing of a Transmission Control Protocol (TCP) connection, loss of connection to a neighbor, or other significant events.

SNMP Versions

SNMP versions provide different capabilities for managing network devices. This software release supports SNMPv1, SNMPv2C, and SNMPv3.

- SNMPv1: The Simple Network Management Protocol, is a Full Internet Standard defined in RFC 1157.
- SNMPv2C: SNMPv2C updates SNMPv2Classic by replacing its Party-based Administrative and Security Framework with a community-string-based framework. It retains SNMPv2Classic's bulk retrieval and improved error handling.

  SNMPv2C includes:

  - SNMPv2: Version 2 of the Simple Network Management Protocol, a Draft Internet Standard defined in RFCs 1902 through 1907.
  - SNMPv2C: The community-string-based Administrative Framework for SNMPv2, an Experimental Internet Protocol defined in RFC 1901.

  Both SNMPv1 and SNMPv2C use a community-based security model. The community of managers able to access the agent's Management Information Base (MIB) is defined by an IP address access control list and password.

  SNMPv2C includes a bulk retrieval function that retrieves tables and large quantities of information, minimizing the number of required round-trips. It also provides improved error handling with expanded error codes that distinguish different error conditions, which are reported through a single error code in SNMPv1.
- SNMPv3: SNMPv3, Version 3 of the SNMP, is an interoperable standards-based protocol defined in RFCs 2273 to 2275. SNMPv3 provides secure access to devices by authenticating and encrypting packets over the network. It includes these security features:

  - message integrity: Ensures that a packet was not tampered with in transit.
  - authentication: Determines that the message is from a valid source.
  - encryption: Mixes the contents of a package to prevent it from being read by an unauthorized source.

Note To select encryption, enter the priv keyword.

SNMPv3 provides for both security models and security levels. A security model is an authentication strategy set up for a user and the group within which the user resides. A security level is the permitted level of security within a security model. A combination of the security level and the security model determines which security method is used when handling an SNMP packet. Available security models are SNMPv1, SNMPv2C, and SNMPv3.

The following table identifies characteristics and compares different combinations of security models and levels:

Table 2: Table 1. SNMP Security Models and Levels

Model	Level	Authentication	Encryption	Result
SNMPv1	noAuthNoPriv	Community string	No	Uses a community string match for authentication.
SNMPv2C	noAuthNoPriv	Community string	No	Uses a community string match for authentication.
SNMPv3	noAuthNoPriv	Username	No	Uses a username match for authentication.
SNMPv3	authNoPriv	Message Digest 5 (MD5) or Secure Hash Algorithm (SHA)	No	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms.
SNMPv3	authPriv	MD5 or SHA	Data Encryption Standard (DES) or Advanced Encryption Standard (AES)	Provides authentication based on the HMAC-MD5 or HMAC-SHA algorithms. Allows specifying the User-based Security Model (USM) with these encryption algorithms: • DES 56-bit encryption in addition to authentication based on the CBC-DES (DES-56) standard. • 3DES 168-bit encryption • AES 128-bit, 192-bit, or 256-bit encryption

SNMP Manager Functions

The SNMP manager uses information in the MIB to perform various operations, as described in this table.

Table 3: SNMP Operations

Operation	Description
get-request	Retrieves a value from a specific variable.
get-next-request	Retrieves a value from a variable within a table. With this operation, an SNMP manager does not need to know the exact variable name; a sequential search is performed to find the needed variable from within a table.

Operation	Description
get-bulk-request	Retrieves large blocks of data, such as multiple rows in a table, that would otherwise require the transmission of many small blocks of data. **Note** This command only works with SNMPv2 or later.
get-response	Replies to a get-request, get-next-request, and set-request sent by an NMS.
set-request	Stores a value in a specific variable.
trap	An unsolicited message sent by an SNMP agent to an SNMP manager when some event has occurred.

Note We recommend that the SNMP Manager exclude the ciscoFlashFileDate MIB object from its query to avoid performance-related issues. This is because, though the ciscoFlashFileDate object is published in the MIB, the product does not support it.

SNMP Agent Functions

The SNMP agent can receive requests from one or more SNMP managers. Each request carries the NMS IP address, the number of times an NMS polls the agent, and a timestamp of polling. You can track this information for both IPv4 and IPv6 servers.

The SNMP agent responds to SNMP manager requests as follows:

- Get a MIB variable: The SNMP agent retrieves the value of the requested MIB variable in response to an NMS request and responds to the NMS with that value.
- Set a MIB variable: The SNMP agent changes the value of the MIB variable to the value requested by the NMS in response to an NMS message.

Use the **show snmp stats hosts** command to display the list of SNMP manager requests in the queue. Use the **clear snmp stats hosts** command to clear the queue.

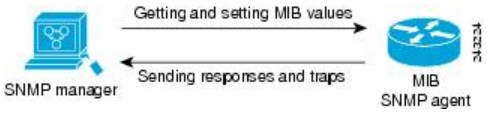
The SNMP agent also sends unsolicited trap messages to notify an NMS that a significant event has occurred on the agent. Examples of trap conditions include, but are not limited to, when a port or module goes up or down, when spanning-tree topology changes occur, and when authentication failures occur.

SNMP MIB Variables Access

Cisco Prime Infrastructure 3.1 software is an example of an NMS. It uses device MIB variables to set device variables and to poll devices on the network for specific information. The results of a poll can be displayed as a graph and analyzed to troubleshoot internetworking problems, increase network performance, verify device configurations, and monitor traffic loads.

The SNMP agent gathers data from the MIB. The agent can send traps, or notifications of certain events, to the SNMP manager, which receives and processes the traps. Traps alert the SNMP manager to conditions on the network such as improper user authentication, restarts, link status (up or down), and MAC address tracking. The SNMP agent also responds to MIB-related queries sent by the SNMP manager in **get-request**, **get-next-request**, and **set-request** format.

Figure 1: SNMP Network



SNMP Flash MIB

The Cisco Flash MIB allows you to query flash file data from Cisco devices. The Flash MIB fetches all files from the flash file system.

To perform a Flash MIB walk, you must use the **snmp mib flash cache** command. This command prefetches all files into the local Flash MIB cache.

SNMP Notifications

SNMP allows the device to send notifications to SNMP managers when particular events occur. SNMP notifications can be sent as traps or inform requests. In command syntax, unless a command option allows you to select either traps or informs, the keyword traps refers to either traps or informs, or both. Use the **snmp-server host**command to specify whether to send SNMP notifications as traps or informs.

Note SNMPv1 does not support informs.

Traps and Informs

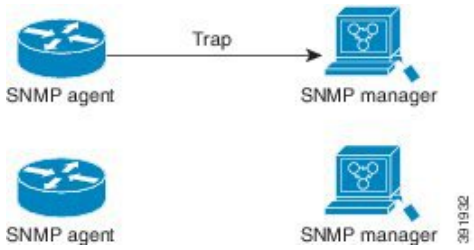
Traps are unreliable because the receiver does not send an acknowledgment when it receives a trap, and the sender cannot determine if the trap was received. When an SNMP manager receives an inform request, it acknowledges the message with an SNMP response protocol data unit (PDU). If the sender does not receive a response, the inform request can be sent again. Because informs can be resent, they are more likely than traps to reach their intended destination.

The characteristics that make informs more reliable than traps also consume more resources in the device and in the network. Unlike a trap, which is discarded as soon as it is sent, an inform request is held in memory until a response is received or the request times out. Traps are sent only once, but an inform might be resent or retried several times. The retries increase traffic and contribute to a higher overhead on the network. Therefore, traps and informs require a trade-off between reliability and resources. If it is important that the SNMP manager receive every notification, use inform requests. If traffic on the network or memory in the device is a concern and notification is not required, use traps.

The figures below illustrate the differences between traps and informs.

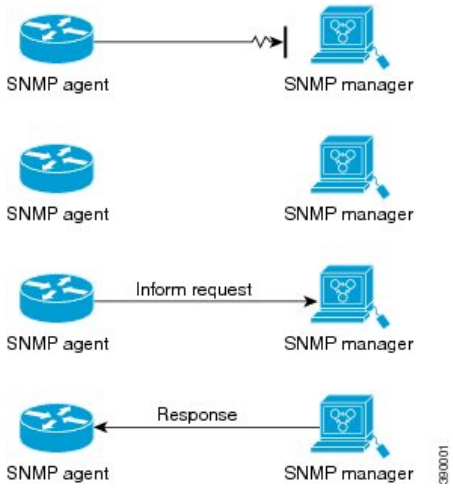
The figure below shows that an agent successfully sends a trap to an SNMP manager. Although the manager receives the trap, it does not send an acknowledgment. The agent has no way of knowing that the trap reached its destination.

Figure 2: Trap Successfully Sent to SNMP Manager



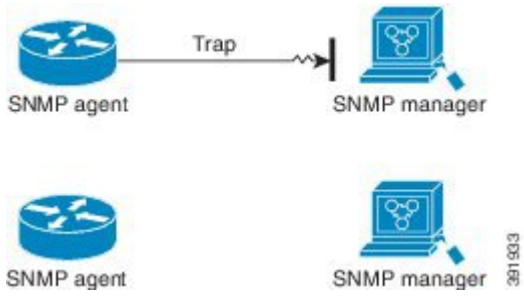
In the figure below, the agent successfully sends an inform to the manager. When the manager receives the inform, a response is sent to the agent, and the agent knows that the inform reached its destination. Note that in this example, the traffic generated is twice as much as in the interaction shown in the figure above.

Figure 3: Inform Request Successfully Sent to SNMP Manager



The figure below shows an agent sending a trap to a manager that the manager does not receive. The agent has no way of knowing that the trap did not reach its destination. The manager never receives the trap because traps are not resent.

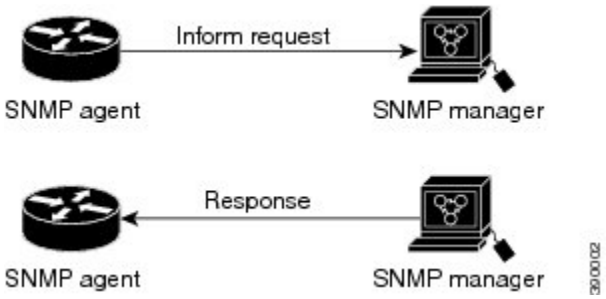
Figure 4: Trap Unsuccessfully Sent to SNMP Manager



The figure below shows an agent sending an inform to a manager that does not reach the manager. Because the manager did not receive the inform, it does not send a response. After a period of time, the agent resends

the inform. The manager receives the inform from the second transmission and replies. In this example, more traffic is generated than in the scenario shown in the figure above, but the notification reaches the SNMP manager.

Figure 5: Inform Unsuccessfully Sent to SNMP Manager



Note Whenever an SNMP process comes up, the reserved ports 161 and 162 are used. In addition to these two reserved ports, a dynamic port is also opened to run the SNMP proxy forwarder application.

SNMP ifIndex MIB Object Values

The SNMP agent's IF-MIB module comes up shortly after a reboot. As various physical interface drivers initialize, they register with the IF-MIB module, requesting an ifIndex number. The IF-MIB module assigns the next available ifIndex number on a first-come, first-served basis. Minor differences in driver initialization order from one reboot to another can result in the same physical interface getting a different ifIndex number than it had before the reboot, unless ifIndex persistency is enabled.

SNMP ENTITY-MIB Identifiers

The ENTITY-MIB contains information for managing physical entities, such as field-replaceable units (FRUs), fans, or power supplies on a device.

Each entity is identified by a unique index number, entPhysicalIndex, which accesses information about the entity in current and other MIBs. An online insertion and removal (OIR) of the entity results in the entity being assigned the next available entPhysicalIndex number, regardless of whether you insert a new entity or reinsert an existing entity.

SNMP and Syslog over IPv6

To support both IPv4 and IPv6, IPv6 network management requires both IPv6 and IPv4 transports. Syslog over IPv6 supports address data types for these transports.

Simple Network Management Protocol (SNMP) and syslog over IPv6 provide these features:

- Support for both IPv4 and IPv6.
- IPv6 transport for SNMP and modification of the SNMP agent to support traps for an IPv6 host.

- SNMP- and syslog-related MIBs to support IPv6 addressing.
- Configuration of IPv6 hosts as trap receivers.

For support over IPv6, SNMP modifies the existing IP transport mapping to simultaneously support IPv4 and IPv6. These SNMP actions support IPv6 transport management:

- Opens User Datagram Protocol (UDP) SNMP socket with default settings.
- Provides a new transport mechanism called SR_IPV6_TRANSPORT.
- Sends SNMP notifications over IPv6 transport.
- Supports SNMP-named access lists for IPv6 transport.
- Supports SNMP proxy forwarding using IPv6 transport.
- Verifies that the SNMP Manager feature works with IPv6 transport.

SNMP UDP ports

The SNMP process uses User Datagram Protocol (UDP) ports 161 and 162. Port 161 is for polling the device, and port 162 is for sending notifications from the agent to the server. These ports remain closed unless you configure one of the requisite commands. This design provides additional security by opening the ports only when needed, preventing a device from unnecessarily listening on a port.

Default SNMP Configuration

Table 4: Table 3. Default SNMP Configuration Settings.

Feature	Default Setting
SNMP agent	Disabled[1].
SNMP trap receiver	None configured.
SNMP traps	None enabled except the trap for TCP connections (tty).
SNMP version	If no version keyword is present, the default is Version 1.
SNMPv3 authentication	If no keyword is entered, the default is the **noauth** (noAuthNoPriv) security level.
SNMP notification type	If no type is specified, all notifications are sent.

[1] This is the default when the device starts and the startup configuration does not have any **snmp-server** global configuration commands.

Restrictions for SNMP

- SNMPv1 does not support informs.
- SNMPv3 authentication is not supported in these scenarios:

- If there is a change in the switch priority followed by a stack reload.
- If a device with a lower MAC address is added to the stack, the device will be elected as the active switch if all the switches in the stack have the same priority.

- To avoid SNMPv3 authentication failure, manually configure the SNMP engineID on the device before SNMPv3 user configuration. This configuration ensures that you can manage and administer the device, as the user is tied to the engineID.
- The SNMP ENTITY-MIB is not supported for the Ethernet management port.

How to Configure SNMP

The following sections provide information on how to configure SNMP.

SNMP Configuration Guidelines

To open SNMP User Datagram Protocol (UDP) ports 161 and 162 and enable the SNMP agent, the device requires one of these global configuration commands:

snmp-server host, **snmp-server user**, **snmp-server community**, or **snmp-server manager**

When configuring SNMP, follow these guidelines:

- Do not specify a notify view when configuring an SNMP group. The snmp-server host global configuration command auto-generates a notify view for the user and adds it to the group associated with that user. Modifying the group's notify view affects all users associated with that group.
- To configure a remote user, specify the IP address or port number for the remote SNMP agent of the device where the user resides.
- Before you configure remote users for a particular agent, configure the SNMP engine ID using the**snmp-server engineID** global configuration command with the remote option. The remote agent's SNMP engine ID and user password compute the authentication and privacy digests. If you do not configure the remote engine ID first, the configuration command fails.
- When configuring SNMP informs, configure the SNMP engine ID for the remote agent in the SNMP database before you send proxy requests or informs to it.
- If a local user is not associated with a remote host, the device does not send informs for the **auth** (authNoPriv) and the **priv** (authPriv) authentication levels.
- Exercise caution when changing the SNMP engine ID. A user's password (entered on the command line) converts to an MD5 or SHA security digest based on the password and the local engine ID. The command-line password is then destroyed, as required by RFC 2274. If the value of the engine ID changes, the security digests of SNMPv3 users become invalid. You must then reconfigure SNMP users using the snmp-server user username global configuration command. Similar restrictions require the reconfiguration of community strings when the engine ID changes.
- When you configure the snmp-server host command with the default UDP port 162, the output of the show running-config command does not display the UDP port value. If you specify a UDP port value other than the default using the snmp-server host {host-addr} community-string udp-port value command, the UDP port number displays in the show running-config command output. You can configure the

snmp-server host command with or without the default UDP port 162; however, you cannot configure both simultaneously.

The following examples are correct:

```
Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community
Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community udp-port 162
```

The following examples are incorrect:

```
Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community
Device(config)# snmp-server host 10.10.10.10 community udp-port 162
Device(config)# snmp-server host 10.10.10.10 community udp-port 163
Device(config)# snmp-server host 10.10.10.10 community udp-port 162
Device(config)# snmp-server host 10.10.10.10 community
```

Configuring SNMP Groups and Users

Before you begin

You can specify an identification name (engine ID) for the local or remote SNMP server engine on the device. Configure an SNMP server group that maps SNMP users to SNMP views, and add new users to the SNMP group.

This section explains how to configure SNMP groups and users on the device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** Device> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** Device# **configure terminal**	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**snmp-server engineID** { **local** *engineid-string* | **remote** *ip-address* [**udp-port** *port-number*] *engineid-string*} **Example:** Device(config)# **snmp-server engineID local 1234**	Configures a name for either the local or remote copy of SNMP. • The *engineid-string* is a 24-character ID string with the name of the copy of SNMP. You need not specify the entire 24-character engine ID if it has trailing zeros. Specify only the portion of the engine ID up to the point where only zeros remain in the value. The Step Example configures an engine ID of 123400000000000000000000. • If you select **remote** , specify the *ip-address* of the device that contains the remote copy of SNMP and the optional User Datagram Protocol (UDP) port on the remote device. The default is 162.

	Command or Action	Purpose
Step 4	**snmp-server group** *group-name* { **v1** | **v2c** | **v3** { **auth** | **noauth** | **priv**} } [**read** *readview*] [**write** *writeview*] [**notify** *notifyview*] [**access** *access-list*] **Example:** Device(config)# **snmp-server group public v2c access lmnop**	Configures a new SNMP group on the remote device. For *group-name* , specify the name of the group. Specify one of the following security models: • **v1** is the least secure of the possible security models. • **v2c** is the second least secure model. It allows transmission of informs and integers twice the normal width. • **v3** , the most secure, requires you to select one of the following authentication levels: **auth** —Enables the Message Digest 5 (MD5) and the Secure Hash Algorithm (SHA) packet authentication. **noauth** —Enables the noAuthNoPriv security level. This is the default if no keyword is specified. **priv** —Enables Data Encryption Standard (DES) packet encryption (also called privacy). (Optional) Enter **read** *readview* with a string (not to exceed 64 characters) that is the name of the view in which you can only view the contents of the agent. (Optional) Enter **write** *writeview* with a string (not to exceed 64 characters) that is the name of the view in which you enter data and configure the contents of the agent. (Optional) Enter **notify** *notifyview* with a string (not to exceed 64 characters) that is the name of the view in which you specify a notify, inform, or trap. (Optional) Enter **access** *access-list* with a string (not to exceed 64 characters) that is the name of the access list.

	Command or Action	Purpose
Step 5	**snmp-server user** *username group-name* { **remote** *host* [**udp-port** *port*] } { **v1** [**access** *access-list*] | **v2c** [**access** *access-list*] | **v3** [**encrypted**] [**access** *access-list*] [**auth** { **md5** | **sha**} *auth-password*] } [*priv* { **des** | **3des** | **aes** { **128** | **192** | **256**} } *priv-password*] **Example:** Device(config)# **snmp-server user Pat public v2c**	

	Command or Action	Purpose
		Adds a new user for an SNMP group. The *username* is the name of the user on the host that connects to the agent. The *group-name* is the name of the group to which the user is associated. Enter **remote** to specify a remote SNMP entity to which the user belongs and the hostname or IP address of that entity with the optional UDP port number. The default is 162. Enter the SNMP version number (**v1** , **v2c** , or **v3**). If you enter **v3** , you have these additional options: • **encrypted** specifies that the password appears in encrypted format. This keyword is available only when the **v3** keyword is specified. • **auth** is an authentication level setting session that can be either the HMAC-MD5-96 (**md5**) or the HMAC-SHA-96 (**sha**) authentication level and requires a password string *auth-password* (not to exceed 64 characters). If you enter **v3** you can also configure a private (**priv**) encryption algorithm and password string *priv-password* using the following keywords (not to exceed 64 characters): • **priv** specifies the User-based Security Model (USM). • **des** specifies the use of the 56-bit DES algorithm. • **3des** specifies the use of the 168-bit DES algorithm. • **aes** specifies the use of the DES algorithm. You must select either 128-bit, 192-bit, or 256-bit encryption. (Optional) Enter **access** *access-list* with a string (not to exceed 64 characters) that is the name of the access list. **Note** The algorithms — **md5** , **des** , **3des** is not supported in a SNMPv3 group when the compliance shield is disabled. You need to enable the compliance shield using the **crypto engine compliance shield enable** command and reboot

	Command or Action	Purpose
		the device to configure the algorithms — **md5** , **des** and **3des** .
Step 6	**end** **Example:** Device(config)# **end**	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** Device# **show running-config**	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** Device# **copy running-config startup-config**	(Optional) Saves your entries in the configuration file.

Opening or Closing SNMP UDP Ports

Beginning in user EXEC mode, follow these steps to open the SNMP UDP ports.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **snmp-server** {**host** | **user** | **community** | **manager**}

Example:

```
Device(config)# snmp-server host
```

Opens SNMP UDP ports 161 and 162.

Configuring any one of the options (**host**, **user**, **community**, **manager**) opens both ports.

To close the ports, enter the **no** form of all the options that you have configured. The ports remain open as long as even one of the keywords is configured.

If you enter the **no snmp-server** command, without any of the keywords, the SNMP process is shut down and not just the SNMP UDP ports.

Step 4 **end**

Example:

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

Returns to privileged EXEC mode.

Step 5 **show udp**

Example:

```
Device# show udp
```

Displays the SNMP UDP ports.

If one of the requisite commands is configured, ports 161 and 162 will display value listen under the remote field.

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

Example:

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

(Optional) Saves your entries in the configuration file.

Setting the Agent Contact and Location Information

Follow these steps to set the system contact and location of the SNMP agent so that these descriptions can be accessed through the configuration file.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

configure terminal

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **snmp-server contact** *text*

Example:

```
Device(config)# snmp-server contact Dial System Operator at beeper 21555
```

Sets the system contact string.

Step 4 **snmp-server location** *text*

Example:

```
Device(config)# snmp-server location Building 3/Room 222
```

Sets the system location string.

Step 5 **end**

Example:

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

Returns to privileged EXEC mode.

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.

Limiting TFTP Servers Used Through SNMP

Follow these steps to limit the TFTP servers used for saving and loading configuration files through SNMP to the servers specified in an access list.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **snmp-server tftp-server-list** *access-list-number*

Example:

```
Device(config)# snmp-server tftp-server-list 44
```

Limits the TFTP servers used for configuration file copies through SNMP to the servers in the access list.

For *access-list-number*, enter an IP standard access list numbered from 1 to 99 and 1300 to 1999.

Step 4 **access-list** *access-list-number* { deny | permit} *source* [*source-wildcard*]

Example:

```
Device(config)# access-list 44 permit 10.1.1.2
```

Creates a standard access list, repeating the command as many times as necessary.

- For *access-list-number*, enter the access list number specified in Step 3.
- The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched.
- For *source*, enter the IP address of the TFTP servers that can access the device.
- (Optional) For *source-wildcard*, enter the wildcard bits, in dotted decimal notation, to be applied to the source. Place ones in the bit positions that you want to ignore.

  The access list is always terminated by an implicit deny statement for everything.

Step 5 **end**

Example:

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

Returns to privileged EXEC mode.

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.

Disabling the SNMP Agent

Follow these steps to disable the SNMP agent.

Before you begin

The SNMP Agent must be enabled before it can be disabled. The SNMP agent is enabled by the **first snmp-server** global configuration command entered on the device.

The **no snmp-server** global configuration command disables all running versions (Version 1, Version 2C, and Version 3) of the SNMP agent on the device and shuts down the SNMP process. You can reenable all versions of the SNMP agent by entering one of the following commands in global configuration mode: **snmp-server host**, or **snmp-server user**, or **snmp-server community** , or s**nmp-server manager**. There is no Cisco IOS command specifically designated for enabling SNMP.

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **no snmp-server**

Example:

```
Device(config)# no snmp-server
```

Disables the SNMP agent operation

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.

Monitoring SNMP Status

To display SNMP input and output statistics, including the number of illegal community string entries, errors, and requested variables, use the **show snmp** privileged EXEC command. You also can use the other privileged EXEC commands listed in the table to display SNMP information.

Table 5: Commands for displaying SNMP information

Command	Purpose
show snmp	Displays SNMP statistics.
show snmp engineID	Displays information on the local SNMP engine and all remote engines that have been configured on the device.
show snmp group	Displays information on each SNMP group on the network.
show snmp pending	Displays information on pending SNMP requests.
show snmp sessions	Displays information on the current SNMP sessions.
show snmp user	Displays information on each SNMP user name in the SNMP users table. **Note** You must use this command to display SNMPv3 configuration information for **auth** | **noauth** | **priv** mode. This information is not displayed in the **show running-config** output.

SNMP Examples

This example shows how to enable all versions of SNMP. The configuration permits any SNMP manager to access all objects with read-only permissions using the community string *public* . This configuration does not cause the device to send any traps.

```
Device(config)# snmp-server community public
```

This example shows how to permit any SNMP manager to access all objects with read-only permission using the community string *public* . The device also sends VTP traps to the hosts 192.180.1.111 and 192.180.1.33 using SNMPv1 and to the host 192.180.1.27 using SNMPv2C. The community string *public* is sent with the traps.

```
Device(config)# snmp-server community public
Device(config)# snmp-server enable traps vtp
Device(config)# snmp-server host 192.180.1.27 version 2c public
Device(config)# snmp-server host 192.180.1.111 version 1 public
Device(config)# snmp-server host 192.180.1.33 public
```

This example shows how to allow read-only access for all objects to members of access list 4 that use the *comaccess* community string. No other SNMP managers have access to any objects. SNMP Authentication Failure traps are sent by SNMPv2C to the host *cisco.com* using the community string *public*.

```
Device(config)# snmp-server community comaccess ro 4
Device(config)# snmp-server enable traps snmp authentication
Device(config)# snmp-server host cisco.com version 2c public
```

This example shows how to send Entity MIB traps to the host *cisco.com* . The community string is restricted. The first line enables the device to send Entity MIB traps in addition to any traps previously enabled. The second line specifies the destination of these traps and overwrites any previous **snmp-server** host commands for the host *cisco.com*.

```
Device(config)# snmp-server enable traps entity
Device(config)# snmp-server host cisco.com restricted entity
```

This example shows how to enable the device to send all traps to the host myhost.cisco.com using the community string public :

```
Device(config)# snmp-server enable traps
Device(config)# snmp-server host myhost.cisco.com public
```

This example shows how to associate a user with a remote host and to send auth (authNoPriv) authentication-level informs when the user enters global configuration mode:

```
Device(config)# snmp-server engineID remote 192.180.1.27 00000063000100a1c0b4011b
Device(config)# snmp-server group authgroup v3 auth
Device(config)# snmp-server user authuser authgroup remote 192.180.1.27 v3 auth md5 mypassword

Device(config)# snmp-server user authuser authgroup v3 auth md5 mypassword
Device(config)# snmp-server host 192.180.1.27 informs version 3 auth authuser config
Device(config)# snmp-server enable traps
Device(config)# snmp-server inform retries 0
```

This example shows how to display the entries of SNMP Managers polled to an SNMP Agent:

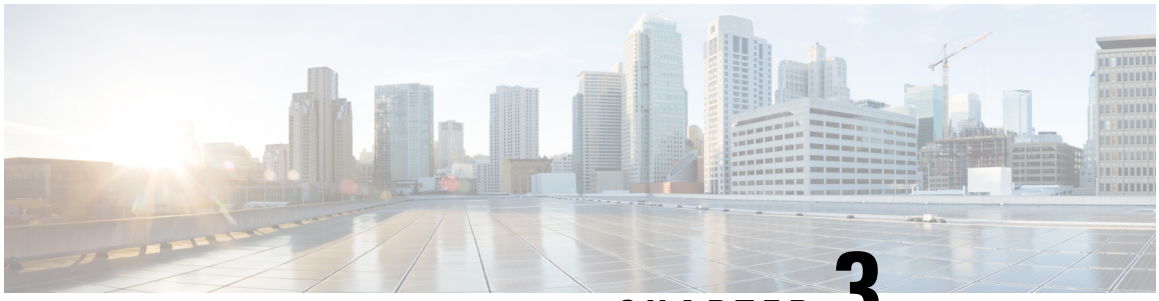
```
Device# show snmp stats host
Request Count Last Timestamp Address
2 00:00:01 ago 3.3.3.3
1 1w2d ago 2.2.2.2
```

This example shows the message displayed by the device when you configure any of the three algorithms — **md5** , **des** , **3des** in a SNMPv3 group when complaince shield is disabled:

```
Device(config)# snmp-server user md5user grp v3 auth md5 cisco1234 priv des
Sep 1 00:14:51.582 IST: %SNMP-6-AUTHPROTOCOLMD5: Authentication protocol md5 support will
be deprecated in future
Sep 1 00:14:51.582 IST: %SNMP-6-PRIVPROTOCOLDES: Privacy protocol des support will be
deprecated in future
Sep 1 00:14:51.645 IST: %SNMP-5-WARMSTART: SNMP agent on host Switch is undergoing a warm
start
```

This example shows the message displayed by the device when you configure any of the three algorithms — **md5**, **des**, **3des** in a SNMPv3 group when compliance shield is enabled. The crypto algorithms is supported along with a warning message:

```
Device(config)# snmp-server user md5user grp v3 auth md5 cisco1234
weaker algorithm MD5, DES and 3DES is not allowed for snmp user
```



CHAPTER 3

Switched Port Analyzer

Switched Port Analyzer (SPAN) is a feature in Cisco switches that provides network administrators with a powerful tool for monitoring, analyzing, and troubleshooting network traffic. This capability is essential for maintaining network health, ensuring security, and optimizing performance by offering deep visibility into data flows without disrupting live operations.

- What is SPAN?, on page 51
- How SPAN works, on page 51
- SPAN Concepts and Terminology, on page 52
- SPAN Interaction with Other Features, on page 56
- SPAN and Device Stacks, on page 57
- Restrictions for SPAN, on page 57
- How to Configure SPAN, on page 58
- Configuration Examples for SPAN, on page 60

What is SPAN?

Switched Port Analyzer (SPAN) allows network administrators to analyze network traffic passing through ports or VLANs by sending a copy of that traffic to another port on the device. This destination port is typically connected to a network analyzer or other monitoring or security device. SPAN copies (or mirrors) traffic received or sent (or both) on source ports or source VLANs to a designated destination port for analysis. A key advantage of SPAN is that it does not affect the normal switching of network traffic on the source ports or VLANs.

How SPAN works

SPAN operates by mirroring traffic from specified network locations to a dedicated monitoring port. These stages describe how SPAN works:

-

1. Identify sources: The network administrator configures one or more source ports or VLANs from which to mirror traffic. These sources can include traffic entering the switch (ingress), traffic leaving the switch (egress), or both

Note Only traffic that enters or leaves source ports or traffic that enters or leaves source VLANs can be monitored by using SPAN. Traffic routed to a source VLAN cannot be monitored. For example, if incoming traffic is being monitored, traffic that gets routed from another VLAN to the source VLAN cannot be monitored; however, traffic that is received on the source VLAN and routed to another VLAN can be monitored.

2. Designate destination: The administrator specifies a single destination port on the switch. A monitoring device, such as a network analyzer or an intrusion detection system, connects to this destination port

Note You must dedicate the destination port for SPAN use. Except for traffic that is required for the SPAN session, destination ports do not receive or forward other network traffic

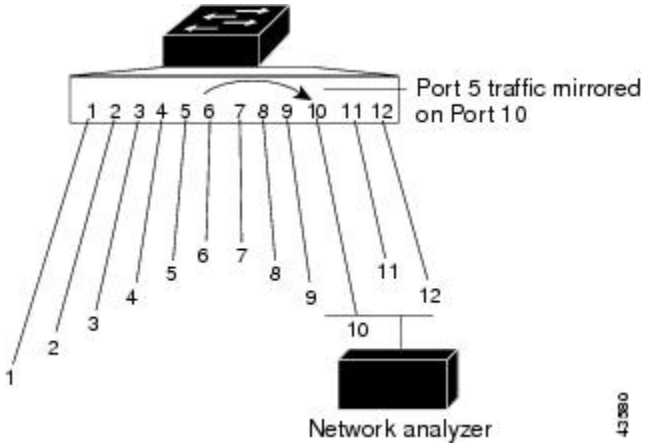
3. Traffic mirroring: The switch duplicates all traffic passing through the configured source ports or VLANs. It then sends these duplicated packets to the designated destination port. The original traffic continues its intended path without interruption.
4. Analysis and active use: The monitoring device connected to the destination port captures and analyzes the mirrored traffic, providing insights into network behavior, application performance, and potential security threats.
5. Traffic injection: You can also use the SPAN destination port to inject traffic from a network security device. For example, if you connect a Cisco Intrusion Detection System (IDS) sensor appliance to a destination port, the IDS device can send TCP reset packets to close down the TCP session of a suspected attacker.

SPAN Concepts and Terminology

Local SPAN

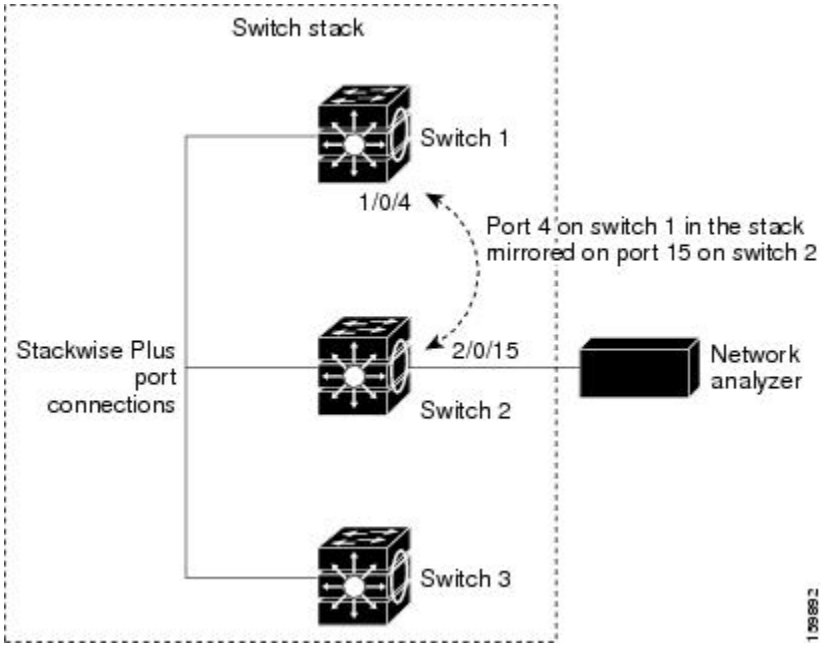
Local SPAN supports a SPAN session entirely within one device; all source ports or source VLANs and destination ports are in the same device or device stack. Local SPAN copies traffic from one or more source ports in any VLAN or from one or more VLANs to a destination port for analysis.

Figure 6: Example of Local SPAN Configuration on a Single Device.



All traffic on port 5 (the source port) is mirrored to port 10 (the destination port). A network analyzer on port 10 receives all network traffic from port 5 without being physically attached to port 5.

Figure 7: Example of Local SPAN Configuration on a Device Stack



SPAN Sessions

A SPAN session allows you to monitor traffic on one or more ports or VLANs, sending the monitored traffic to one or more destination ports. A local SPAN session is an association of a destination port with source ports or source VLANs, all configured on a single network device. These sessions gather specified ingress and egress packets, forming them into a stream of SPAN data directed to the destination port.Key characteristics of SPAN sessions include:

- Both switched and routed ports can be configured as SPAN sources and destinations.

- SPAN sessions do not interfere with the normal operation of the device. However, an oversubscribed SPAN destination (for example, a 10-Mb/s port monitoring a 100-Mb/s port) can result in dropped or lost packets.
- When SPAN is enabled, each monitored packet is sent twice: once as normal traffic and once as a monitored packet. Monitoring a large number of ports or VLANs could potentially generate significant network traffic.
- You can configure SPAN sessions on disabled ports. However, a SPAN session becomes active only when the destination port and at least one source port or VLAN for that session are enabled.

Monitored Traffic

SPAN sessions can monitor these traffic types:

- Receive (Rx) SPAN—Receive (or ingress) SPAN monitors as much as possible all of the packets received by the source interface or VLAN before any modification or processing is performed by the device. A copy of each packet received by the source is sent to the destination port for that SPAN session.
  - Packets that are modified because of routing or Quality of Service (QoS)—for example, modified Differentiated Services Code Point (DSCP)—are copied before modification.
  - Features that can cause a packet to be dropped during receive processing have no effect on ingress SPAN; the destination port receives a copy of the packet even if the actual incoming packet is dropped. These features include IP standard and extended input Access Control Lists (ACLs), ingress QoS policing, VLAN ACLs, and egress QoS policing.
- Transmit (Tx) SPAN—Transmit (or egress) SPAN monitors as much as possible all of the packets sent by the source interface after all modification and processing is performed by the device. A copy of each packet sent by the source is sent to the destination port for that SPAN session. The copy is provided after the packet is modified.
  - Packets that are modified because of routing (for example, with modified time-to-live (TTL), MAC address, or QoS values) are duplicated (with the modifications) at the destination port.
  - Features that can cause a packet to be dropped during transmit processing also affect the duplicated copy for SPAN. These features include IP standard and extended output ACLs and egress QoS policing.
- Both—In a SPAN session, you can also monitor a port or VLAN for both received and sent packets. This is the default.

By default, local SPAN sessions replicate source packets along with encapsulation:

- Packets are sent on the destination port with the same encapsulation (untagged or IEEE 802.1Q) that they had on the source port.
- Packets of all types, including BPDU and Layer 2 protocol packets, are monitored.

Therefore, a local SPAN session can have a mixture of untagged and IEEE 802.1Q tagged packets appear on the destination port.

Device congestion can cause packets to be dropped at ingress source ports, egress source ports, or SPAN destination ports. In general, these characteristics are independent of one another. For example:

- A packet might be forwarded normally but dropped from monitoring due to an oversubscribed SPAN destination port.
- An ingress packet might be dropped from normal forwarding, but still appear on the SPAN destination port.
- An egress packet dropped because of device congestion is also dropped from egress SPAN.

In some SPAN configurations, multiple copies of the same source packet are sent to the SPAN destination port. For example, a bidirectional (both Rx and Tx) SPAN session is configured for the Rx monitor on port A and Tx monitor on port B. If a packet enters the device through port A and is switched to port B, both incoming and outgoing packets are sent to the destination port. Both packets are the same unless a Layer 3 rewrite occurs, in which case the packets are different because of the packet modification.

Source Ports

A source port (also called a monitored port) is a switched or routed port that you monitor for network traffic analysis.

In a local SPAN session, you can monitor source ports or VLANs for traffic in one or both directions.

The device supports any number of source ports (up to the maximum number of available ports on the device) and up to 1500 source VLANs.

A source port has these characteristics:

- Each source port can be configured with a direction (ingress, egress, or both) to monitor.
- It can be any port type (for example, EtherChannel, Gigabit Ethernet, and so forth).
- For EtherChannel sources, you can monitor traffic for the entire EtherChannel or individually on a physical port as it participates in the port channel.
- It can be an access port, trunk port, routed port, or voice VLAN port.
- It cannot be a destination port.
- Source ports can be in the same or different VLANs.
- You can monitor multiple source ports in a single session.

Source VLANs

VLAN-based SPAN (VSPAN) is the monitoring of the network traffic in one or more VLANs. The SPAN source interface in VSPAN is a VLAN ID, and traffic is monitored on all the ports for that VLAN.

VSPAN has these characteristics:

- All active ports in the source VLAN are included as source ports and can be monitored in either or both directions.
- On a given port, only traffic on the monitored VLAN is sent to the destination port.
- If a destination port belongs to a source VLAN, it is excluded from the source list and is not monitored.
- If ports are added to or removed from the source VLANs, the traffic on the source VLAN received by those ports is added to or removed from the sources being monitored.
- You can monitor only Ethernet VLANs.

- The number of source VLANs per session must not exceed 1,500. This limit is a combined total for both receive (RX) and transmit (TX) directions.

Destination Ports

Each local SPAN session must have a destination port (also called a monitoring port) that receives a copy of traffic from the source ports or VLANs and sends the SPAN packets to the user, usually a network analyzer.

A destination port has these characteristics:

- A SPAN session can have one destination port per session. It can participate in only one SPAN session at a time (a destination port in one SPAN session cannot be a destination port for a second SPAN session).
- For a local SPAN session, the destination port must reside on the same device or device stack as the source port.
- When a port is configured as a SPAN destination port, the configuration overwrites the original port configuration. When the SPAN destination configuration is removed, the port reverts to its previous configuration. If a configuration change is made to the port while it is acting as a SPAN destination port, the change does not take effect until the SPAN destination configuration had been removed.
- If the port was in an EtherChannel group, it is removed from the group while it is a destination port. If it was a routed port, it is no longer a routed port.
- It can be any Ethernet physical port. It cannot be a secure port or a source port.
- When it is active, incoming traffic is disabled. The port does not transmit any traffic except that required for the SPAN session. Incoming traffic is never learned or forwarded on a destination port.
- It does not participate in any of the Layer 2 protocols (STP, VTP, CDP, DTP, PagP).
- A destination port that belongs to a source VLAN of any SPAN session is excluded from the source list and is not monitored.
- The maximum number of destination ports in a device or device stack is 64.

For local SPAN, source packets at the destination port appear with the original encapsulation (untagged, ISL, or IEEE802.1Q) by default. The output of a local SPAN session can contain a mixture of untagged, ISL, or IEEE 802.1Q-tagged packets.

SPAN Interaction with Other Features

SPAN interacts with these features:

- Routing—SPAN does not monitor routed traffic. VSPAN only monitors traffic that enters or exits the device, not traffic that is routed between VLANs. For example, if a VLAN is being Rx-monitored and the device routes traffic from another VLAN to the monitored VLAN, that traffic is not monitored and not received on the SPAN destination port.
- STP—A destination port does not participate in STP while its SPAN session is active. The destination port can participate in STP after the SPAN session is disabled. On a source port, SPAN does not affect the STP status.
- CDP—A SPAN destination port does not participate in CDP while the SPAN session is active. After the SPAN session is disabled, the port again participates in CDP.

- VLAN and trunking—You can modify VLAN membership or trunk settings for source or destination ports at any time. However, changes in VLAN membership or trunk settings for a destination port do not take effect until you remove the SPAN destination configuration. Changes in VLAN membership or trunk settings for a source port immediately take effect, and the respective SPAN sessions automatically adjust accordingly.
- EtherChannel—You can configure an EtherChannel group as a source port. When a group is configured as a SPAN source, the entire group is monitored.
  - If a physical port is added to a monitored EtherChannel group, the new port is added to the SPAN source port list. If a port is removed from a monitored EtherChannel group, it is automatically removed from the source port list.
  - A physical port that belongs to an EtherChannel group cannot be configured as a SPAN source port. However, if a physical port that belongs to an EtherChannel group is configured as a SPAN destination, it is removed from the group. After the port is removed from the SPAN session, it rejoins the EtherChannel group. Ports removed from an EtherChannel group remain members of the group, but they are in the inactive or suspended state.
  - If a physical port that belongs to an EtherChannel group is a destination port and the EtherChannel group is a source, the port is removed from the EtherChannel group and from the list of monitored ports.
- Multicast traffic can be monitored. For egress and ingress port monitoring, only a single unedited packet is sent to the SPAN destination port. It does not reflect the number of times the multicast packet is sent.
- A private-VLAN port cannot be a SPAN destination port.
- A secure port cannot be a SPAN destination port.
- An IEEE 802.1x port can be a SPAN source port. You can enable IEEE 802.1x on a port that is a SPAN destination port; however, IEEE 802.1x is disabled until the port is removed as a SPAN destination.

SPAN and Device Stacks

Because the stack of switches represents one logical switch, local SPAN source ports and destination ports can be in different switches in the stack. Therefore, the addition or deletion of switches in the stack can affect a local SPAN session. An active session can become inactive when a switch is removed from the stack or an inactive session can become active when a switch is added to the stack.

Restrictions for SPAN

Observe these restrictions when configuring SPAN sessions:

- A device supports a maximum of 66 monitoring sessions, with up to 8 designated as source sessions to monitor and capture traffic from specified source ports or VLANs.
- Do not mix source ports and source VLANs within the same SPAN session.
- A destination port cannot also be a source port within any SPAN session.

- A SPAN session can have multiple destination ports, but a device stack supports a maximum of 64 destination ports.
- An oversubscribed SPAN destination (for example, a 10-Mb/s port monitoring a 100-Mb/s port) can result in dropped or lost packets.
- A SPAN source port or source VLAN cannot be part of more than one SPAN session.
- A SPAN session can have only one destination port, and must use a unique destination port.
- An EtherChannel group cannot be a SPAN destination port.
- An EtherChannel member cannot be a SPAN source port.

How to Configure SPAN

SPAN operates by mirroring traffic from specified network locations to a dedicated monitoring port. These stages describe how SPAN works:

1. Identify sources: The network administrator configures one or more source ports or VLANs from which to mirror traffic. These sources can include traffic entering the switch (ingress), traffic leaving the switch (egress), or both.

   Only traffic that enters or leaves source ports or traffic that enters or leaves source VLANs can be monitored by using SPAN. Traffic routed to a source VLAN cannot be monitored. For example, if incoming traffic is being monitored, traffic that gets routed from another VLAN to the source VLAN cannot be monitored; however, traffic that is received on the source VLAN and routed to another VLAN can be monitored.

2. Designate destination: The administrator specifies a single destination port on the switch. A monitoring device, such as a network analyzer or an intrusion detection system, connects to this destination port.

   You must dedicate the destination port for SPAN use. Except for traffic that is required for the SPAN session, destination ports do not receive or forward other network traffic.

3. Traffic mirroring: The switch duplicates all traffic passing through the configured source ports or VLANs. It then sends these duplicated packets to the designated destination port. The original traffic continues its intended path without interruption.

4. Analysis and active use: The monitoring device connected to the destination port captures and analyzes the mirrored traffic, providing insights into network behavior, application performance, and potential security threats.

   Traffic injection: You can also use the SPAN destination port to inject traffic from a network security device. For example, if you connect a Cisco Intrusion Detection System (IDS) sensor appliance to a destination port, the IDS device can send TCP reset packets to close down the TCP session of a suspected attacker.

Creating a local SPAN session

Follow these steps to create a SPAN session and specify the source (monitored) ports or VLANs and the destination (monitoring) ports.

Procedure

Step 1 Enable

Example:

```
Device# configure terminal
```

Enables privileged EXEC mode.

Step 2 configure terminal

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **no monitor session** {*session_number* | **all** | **local** | **remote**}

Example:

```
Device(config)# no monitor session all
```

Removes any existing SPAN configuration for the session.

- For **session_number** , the range is 1 to 66.
- **all** —Removes all SPAN sessions.
- **local** —Removes all local sessions.
- **remote** —Removes all remote SPAN sessions.

Step 4 **monitor session** *session_number* **source** {**interface** *interface-id* | **vlan** *vlan-id*} [**,** | **-**] [**both** | **rx** | **tx**]

Example:

```
Device(config)# monitor session 1 source interface gigabitethernet1/0/1
```

Specifies the SPAN session and the source port (monitored port).

- For **session_number,** the range is 1 to 66.
- For interface-id , specify the source port to monitor. Valid interfaces include physical interfaces and port-channel logical interfaces (**port-channel** *port-channel-number*). Valid port-channel numbers are 1 to 48.
- For **vlan-id** , specify the source VLAN to monitor. The range is 1 to 4094.

  Note
  A single session can include multiple sources (ports or VLANs) defined in a series of commands, but you cannot combine source ports and source VLANs in one session.

- (Optional) **[, | -]** Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen.
- (Optional) **both | rx | tx** —Specifies the direction of traffic to monitor. If you do not specify a traffic direction, the source interface sends both sent and received traffic.

- both —Monitors both received and sent traffic.
- rx —Monitors received traffic.
- tx —Monitors sent traffic.

Note
You can use the **monitor session** *session_number* **source** command multiple times to configure multiple source ports.

Step 5 **monitor session** *session_number* **destination** {**interface** *interface-id* [**,** | **-**]

Example:

```
Device(config)# monitor session 1 destination interface gigabitethernet1/0/2
```

Specifies the SPAN session and the destination port (monitoring port). The port LED changes to amber when the configuration changes take effect. The LED returns to its original state (green) only after removing the SPAN destination configuration.

- For local SPAN, you must use the same session number for the source and destination interfaces.
- For **session_number** , specify the session number entered in step 4.
- For **interface-id** , specify the destination port. The destination interface must be a physical port; it cannot be an EtherChannel, and it cannot be a VLAN.

  (Optional) **[, | -]** Specifies a series or range of interfaces. Enter a space before and after the comma; enter a space before and after the hyphen.

Step 6 end

Example:

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

Returns to privileged EXEC mode.

Step 7 show running-config

Example:

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

Verifies your entries.

Step 8 copy running-config startup-config

Example:

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

(Optional) Saves your entries in the configuration file.

Configuration Examples for SPAN

The following sections provide configuration examples for SPAN.

This example shows how to set up SPAN session 1 for monitoring source port traffic to a destination port. First, any existing SPAN configuration for session 1 is deleted, and then bidirectional traffic is mirrored from source Gigabit Ethernet port 1 to destination Gigabit Ethernet port 2, retaining the encapsulation method.

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1
Device(config)# monitor session 1 source interface gigabitethernet1/0/1
Device(config)# monitor session 1 destination interface gigabitethernet1/0/2
Device(config)# end
```

This example shows how to remove port 1 as a SPAN source for SPAN session 1:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1 source interface gigabitethernet1/0/1
Device(config)# end
```

This example shows how to disable received traffic monitoring on port 1, which was configured for bidirectional monitoring:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 1 source interface gigabitethernet1/0/1 rx
```

The monitoring of traffic received on port 1 is disabled, but traffic sent from this port continues to be monitored.

This example shows how to remove any existing configuration on SPAN session 2, configure SPAN session 2 to monitor received traffic on all ports belonging to VLANs 1 through 3, and send it to destination Gigabit Ethernet port 2. The configuration is then modified to also monitor all traffic on all ports belonging to VLAN 10.

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 2
Device(config)# monitor session 2 source vlan 1 - 3 rx

Device(config)# monitor session 2 destination interface gigabitethernet1/0/2
Device(config)# monitor session 2 source vlan 10
Device(config)# end
```

This example shows how to remove any existing configuration on SPAN session 2, configure SPAN session 2 to monitor traffic received on Gigabit Ethernet trunk port 2:

```
Device> enable
Device# configure terminal
Device(config)# no monitor session 2
Device(config)# monitor session 2 source interface gigabitethernet1/0/2 rx
Device(config)# monitor session 2 destination interface gigabitethernet1/0/1
Device(config)# end
```