Configuring Wireshark

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Wireshark

- Wireshark is supported on Supervisor Engine 7-E, Supervisor Engine 7L-E, Catalyst 3850, Catalyst 3650, Wireless LAN Controller 5700 Series, Catalyst 4500X-16, and Catalyst 4500X-32.

Restrictions for Wireshark

- Starting in Cisco IOS Release XE 3.3.0(SE), global packet capture on Wireshark is not supported.
- Capture filters are not supported.

- The CLI for configuring Wireshark requires that the feature be executed only from EXEC mode. Actions that usually occur in configuration submode (such as defining capture points), are handled at the EXEC mode instead. All key commands are not NVGEN'd and are not synchronized to the standby supervisor in NSF and SSO scenarios.
- Packets captured in the output direction of an interface might not reflect the changes made by switch rewrite (includes TTL, VLAN tag, CoS, checksum, MAC addresses, DSCP, precedent, UP, etc.).
- Limiting circular file storage by file size is not supported.

Wireless Packet Capture

- The only form of wireless capture is a CAPWAP tunnel capture.
- When capturing CAPWAP tunnels, no other interface types can be used as attachment points on the same capture point.
- Capturing multiple CAPWAP tunnels is supported.
- Core filters are not applied and should be omitted when capturing a CAPWAP tunnel.
- To capture a CAPWAP data tunnel, each CAPWAP tunnel is mapped to a physical port and an appropriate ACL will be applied to filter the traffic.
- To capture a CAPWAP non-data tunnel, the switch is set to capture traffic on all ports and apply an appropriate ACL to filter the traffic.

Configuration Limitations

- 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.
- Neither VRFs, management ports, nor private VLANs can be used as attachment points.
- Only one ACL of each type (IPv4, IPv6, MAC) is allowed in a Wireshark class map. There can be a maximum of three ACLs in a class map: one for IPv4, one for IPv6, and the other for MAC.
- Wireshark cannot capture packets on a destination SPAN port.
- Wireshark will stop 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 switch. To resume capturing, the capture must be restarted manually.
- CPU-injected packets are considered control plane packets. Therefore, these types of packets will not be captured on an interface egress capture.
- MAC ACL is only used for non-IP packets such as ARP. It will not be supported on a Layer 3 port or SVI.
- IPv6-based ACLs are not supported in VACL.
- Layer 2 and Layer 3 EtherChannels are not supported.
- ACL logging and Wireshark are incompatible. Once Wireshark is activated, it takes priority. All traffic, including that being captured by ACL logging on any ports, will be redirected to Wireshark. We

recommended that you deactivate ACL logging before starting Wireshark. Otherwise, Wireshark traffic will be contaminated by ACL logging traffic.

- Wireshark does not capture packets dropped by floodblock.
- If you capture both PACL and RACL on the same port, only one copy is sent to the CPU. If you capture a DTLS-encrypted CAPWAP interface, two copies are sent to Wireshark, one encrypted and the other decrypted. The same behavior will occur if we capture a Layer 2 interface carrying DTLS-encrypted CAPWAP traffic. The core filter is based on the outer CAPWAP header.

Information About Wireshark

Wireshark Overview

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

The ability to capture and analyze traffic provides data on network activity. Prior to Cisco IOS Release XE 3.3.0(SE), only two features addressed this need: SPAN and debug platform packet. Both have limitations. SPAN is ideal for capturing packets, but can only deliver them by forwarding them to some specified local or remote destination; it provides no local display or analysis support. The **debug platform packet** command is specific to the Catalyst 4500 series and only works on packets that come from the software process-forwarding path. Also, the **debug platform packet** command has limited local display capabilities and no analysis support.

So the need exists for a traffic capture and analysis mechanism that is applicable to both hardware and software forwarded traffic and that provides strong packet capture, display, and analysis support, preferably using a well known interface.

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

Capture Points

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

Multiple capture points can be defined, but only one can be active at a time. You need to stop one before you can start the other.

Attachment Points

An attachment point is a point in the logical packet process path associated with a capture point. An attachment point is an attribute of the capture point. Packets that impact an attachment point are tested against capture point filters; packets that match are copied and sent to the associated Wireshark instance of the capture point.

A specific capture point can be associated with multiple attachment points, with limits on mixing attachment points of different types. Some restrictions apply when you specify attachment points of different types. Attachment points are directional (input or output or both) with the exception of the Layer 2 VLAN attachment point, which is always bidirectional.

Filters

Filters are attributes of a capture point that identify and limit the subset of traffic traveling through the attachment point of a capture point, which is copied and passed to Wireshark. To be displayed by Wireshark, a packet must pass through an attachment point, as well as all of the filters associated with the capture point.

A capture point has the following types of filters:

- Core system filter—The core system filter is applied by hardware, and its match criteria is limited by hardware. This filter determines whether hardware-forwarded traffic is copied to software for Wireshark purposes.
- Display filter—The display filter is applied by Wireshark. Packets that fail the display filter are not displayed.

Core System Filter

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

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

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

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

Note The 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.

Related Topics

Additional References, on page 37

Actions

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

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

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

Storage of Captured Packets to Buffer in Memory

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

The capture buffer can be in linear or circular mode. In linear mode, new packets are discarded when the buffer is full. In circular mode, if the buffer is full, the oldest packets are discarded to accommodate the new packets. Although the buffer can also be cleared when needed, this mode is mainly used for debugging network traffic.

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

Storage of Captured Packets to a .pcap File

Note When WireShark is used on switches in a stack, packet captures can be stored only on flash or USB flash devices connected to the active switch.

For example, if 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:

- Switch 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.

When configuring a Wireshark capture point, you can associate a filename. When the capture point is activated, Wireshark creates a file with the specified name and writes packets to it. If the file already exists when the file is associated or the capture point is activated, Wireshark queries you as to whether the file can be overwritten. Only one capture point may be associated with a given filename.

If the destination of the Wireshark writing process is full, Wireshark fails with partial data in the file. You must ensure that there is sufficient space in the file system before you start the capture session. With Cisco IOS Release IOS XE 3.3.0(SE), the file system full status is not detected for some storage devices.

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

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

Packet Decoding and Display

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

Note Decoding and displaying packets may be CPU intensive.

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

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

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

Live Traffic Display

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

.pcap File Display

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

Packet Storage and Display

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

Wireshark Capture Point Activation and Deactivation

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

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

Note *When performing a wireless capture with a CAPWAP tunneling interface, the core system filter is not required and cannot be used.

The display filters are specified as needed.

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

When a Wireshark capture point is activated, a fixed rate policer is applied automatically in the hardware so that the CPU is not flooded with Wireshark-directed packets. The disadvantage of the rate policer is that you cannot capture contiguous packets beyond the established rate even if more resources are available.

Wireshark Features

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

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

  On ingress, a packet goes through a Layer 2 port, a VLAN, and a Layer 3 port/SVI. On egress, the packet goes through a Layer 3 port/SVI, a VLAN, and a Layer 2 port. If the attachment point is before the point where the packet is dropped, Wireshark will capture the packet. Otherwise, Wireshark will not capture the packet. For example, Wireshark capture policies connected to Layer 2 attachment points in the input direction capture packets dropped by Layer 3 classification-based security features. Symmetrically, Wireshark capture policies attached to Layer 3 attachment points in the output direction capture packets dropped by Layer 2 classification-based security features.
- Routed ports and switch virtual interfaces (SVIs)—Wireshark cannot capture the output of an SVI because the packets that go out of an SVI's output are generated by CPU. To capture these packets, include the control plane as an attachment point.

- VLANs—When a VLAN is used as a Wireshark attachment point, packets are captured in the input direction only.
- Redirection features—In the input direction, features traffic redirected by Layer 3 (such as PBR and WCCP) are logically later than Layer 3 Wireshark attachment points. Wireshark captures these packets even though they might later be redirected out another Layer 3 interface. Symmetrically, output features redirected by Layer 3 (such as egress WCCP) are logically prior to Layer 3 Wireshark attachment points, and Wireshark will not capture them.
- SPAN—Wireshark and SPAN sources are compatible. You can configure an interface as a SPAN source and as a Wireshark attachment point simultaneously. Configuring a SPAN destination port as a Wireshark attachment point is not supported.
- 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.

Wireless Packet Capture in Wireshark

- Wireless traffic is encapsulated inside CAPWAP packets. However, capturing only a particular wireless client's traffic inside a CAPWAP tunnel is not supported when using the CAPWAP tunnel as an attachment point. To capture only a particular wireless client's traffic, use the client VLAN as an attachment point and formulate the core filter accordingly.
- Limited decoding of inner wireless traffic is supported. Decoding of inner wireless packets inside encrypted CAPWAP tunnels is not supported.
- No other interface type can be used with the CAPWAP tunneling interface on the same capture point. A CAPWAP tunneling interface and a Level 2 port cannot be attachment points on the same capture point.
- You cannot specify a core filter when capturing packets for Wireshark via the CAPWAP tunnel. However, you can use the Wireshark display filters for filtering wireless client traffic against a specific wireless client.
- You can capture packets from a maximum of 135 CAPWAP tunnels at a time if no ACLs are applied. If ACLs are applied, the hardware memory will have less space for Wireshark to use. As a result, the maximum number of CAPWAP tunnels than can be used for packet capture at a time will be lower. Using more than 135 CAPWAP tunnels at a time or unsing 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 Wireshark

- During Wireshark packet capture, hardware forwarding happens concurrently.
- Before starting a Wireshark capture process, ensure that CPU usage is moderate and that sufficient memory (at least 200 MB) is available.
- If you plan to store packets to a storage file, ensure that sufficient space is available before beginning a Wireshark capture process.
- The CPU usage during Wireshark capture depends on how many packets match the specified conditions and on the intended actions for the matched packets (store, decode and display, or both).
- Where possible, keep the capture to the minimum (limit by packets, duration) to avoid high CPU usage and other undesirable conditions.
- Because packet forwarding typically occurs in hardware, packets are not copied to the CPU for software processing. For Wireshark packet capture, packets are copied and delivered to the CPU, which causes an increase in CPU usage.

  To avoid high CPU usage, do the following:

  - Attach only relevant ports.
  - Use a class map, and secondarily, an access list to express match conditions. If neither is viable, use an explicit, in-line filter.
  - Adhere closely to the filter rules. Restrict the traffic type (such as, IPv4 only) with a restrictive, rather than relaxed ACL, which elicits unwanted traffic.
- 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
- Run a capture session without limits if you know that very little traffic matches the core filter.
- You might experience high CPU (or memory) usage if:
  - You leave a capture session enabled and unattended for a long period of time, resulting in unanticipated bursts of traffic.
  - You launch a capture session with ring files or capture buffer and leave it unattended for a long time, resulting in performance or system health issues.
- During a capture session, watch for high CPU usage and memory consumption due to Wireshark that may impact switch performance or health. If these situations arise, stop the Wireshark session immediately.
- 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.

- 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.
- Whenever an ACL that is associated with a running capture is modified, you must restart the capture for the ACL modifications to take effect. If you do not restart the capture, it will continue to use the original ACL as if it had not been modified.
- To avoid packet loss, consider the following:
    - Use store-only (when you do not specify the display option) while capturing live packets rather than decode and display, which is an CPU-intensive operation (especially in detailed mode).
    - If you have more than one capture that is storing packets in a buffer, clear the buffer before starting a new capture to avoid memory loss.
    - If you use the default buffer size and see that you are losing packets, you can increase the buffer size to avoid losing packets.
    - Writing to flash disk is a CPU-intensive operation, so if the capture rate is insufficient, you may want to use a buffer capture.
    - The Wireshark capture session operates normally in streaming mode where packets are both captured and processed. However, when you specify a buffer size of at least 32 MB, the session automatically turns on lock-step mode in which a Wireshark capture session is split into two phases: capture and process. In the capture phase, the packets are stored in the temporary buffer. The duration parameter in lock-step mode serves as capture duration rather than session duration. When the buffer is full or the capture duration or packet limit has been attained, a session transitions to the process phase, wherein it stops accepting packets and starts processing packets in the buffer. You can also stop the capture manually. You will see a message in the output when the capture stops. With this second approach (lock-step mode), a higher capture throughput can be achieved.

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

    - 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.
- If you want to decode and display live packets in the console window, ensure that the Wireshark session is bounded by a short capture duration.

Note **Warning**: 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.

- When using Wireshark to capture live traffic that leads to high CPU, usage, consider applying a QoS policy temporarily to limit the actual traffic until the capture process concludes.
- All Wireshark-related commands are in EXEC mode; no configuration commands exist for Wireshark.

If you need to use access list or class-map in the Wireshark CLI, you must define an access list and class map with configuration commands.

- No specific order applies when defining a capture point; you can define capture point parameters in any order, provided that CLI allows this. The Wireshark CLI allows as many parameters as possible on a single line. This limits the number of commands required to define a capture point.
- All parameters except attachment points take a single value. Generally, you can replace the value with a new one by reentering the command. After user confirmation, the system accepts the new value and overrides the older one. A **no** form of the command is unnecessary to provide a new value, but it is necessary to remove a parameter.
- Wireshark allows you to specify one or more attachment points. To add more than one attachment point, reenter the command with the new attachment point. To remove an attachment point, use the **no** form of the command. You can specify an interface range as an attachment point. For example, enter **monitor capture mycap interface GigabitEthernet1/0/1 in** where interface GigabitEthernet1/0/1 is an attachment point.

  If you also need to attach interface GigabitEthernet1/0/2, specify it in another line as follows:

  monitor capture mycap interface GigabitEthernet1/0/2 in
- You can modify any of the parameters of a capture point while a session is active, but you must restart the session for the modifications to take effect.
- 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 capture file already exists, it provides a warning and receives confirmation before proceeding. This prevents you from mistakenly overwriting a file.
- 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.
- You can terminate a Wireshark session with an explicit **stop** command or by entering **q** in automore mode. The session could terminate itself automatically when a stop condition such as duration or packet capture limit is met.

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)

Feature	Default Setting
File size	No limit
Ring file storage	No
Buffer storage mode	Linear

How to Configure Wireshark

To configure Wireshark, perform these basic steps.

1. Define a capture point.
2. (Optional) Add or modify the capture point's parameters.
3. Activate or deactivate a capture point.
4. Delete the capture point when you are no longer using it.

Related Topics

Defining a Capture Point, on page 12
Adding or Modifying Capture Point Parameters, on page 17
Deleting Capture Point Parameters, on page 18
Deleting a Capture Point, on page 19
Activating and Deactivating a Capture Point, on page 20
Clearing the Capture Point Buffer, on page 21

Defining a Capture Point

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

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

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

In privileged EXEC mode, follow these steps to define a capture point.

Procedure

	Command or Action	Purpose
Step 1	**show capwap summary** **Example:**	Displays the CAPWAP tunnels available as attachment points for a wireless capture.

	Command or Action	Purpose
	`Device# show capwap summary`	**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 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).
		• (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 of the parameters for this capture point with this one command instance. These parameters are discussed in the instructions for modifying capture point parameters. Range support is also available both for adding and removing attachment points.
		Use one of the following for *interface-type*:
		• **GigabitEthernet**—Specifies the attachment point as GigabitEthernet.
		• **vlan**—Specifies the attachment point as a VLAN.
		Note Only ingress capture (in) is allowed when using this interface as an attachment point.
		• **capwap**—Specifies the attachment point as a CAPWAP tunnel.

	Command or Action	Purpose
		Note When using this interface as an attachment point, a core filter cannot be used. • (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, do not 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** A capture point cannot be activated if it has neither a core system filter nor attachment points defined. Attempting to activate a capture point that does not meet these requirements generates an error. • **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 that you defined in Step 1 and confirms that you defined a capture point.

Example

To define a capture point with a CAPWAP attachment point:

```
Device# show capwap summary

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


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


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

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

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

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

Status Information for Capture mycap
  Target Type:
  Interface: CAPWAP,
    Ingress:
 0
    Egress:
 0
   Status : Active
  Filter Details:
    Capture all packets
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:mycap.pcap
   Size of buffer(in MB): 1
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Packets per second: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
Device#
Device# show monitor capture file flash:mycap.pcap
  1   0.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  2   0.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  3   2.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  4   2.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  5   3.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  6   4.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  7   4.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  8   5.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
  9   5.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 10   6.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 11   8.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 12   9.225986   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 13   9.225986   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 14   9.225986   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 15   9.231998   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 16   9.231998   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 17   9.231998   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 18   9.236987   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 19  10.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 20  10.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 21  12.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 22  12.239993   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 23  12.244997   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 24  12.244997   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 25  12.250994   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 26  12.256990   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 27  12.262987   10.10.14.2 -> 10.10.14.32  DTLSv1.0 Application Data
 28  12.499974 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
 29  12.802012   10.10.14.3 -> 10.10.14.255 NBNS Name query NB WPAD.<00>
 30  13.000000 00:00:00:00:00:00 -> 3c:ce:73:39:c6:60 IEEE 802.11 Probe Request, SN=0, FN=0,
 Flags=........
```

What to do next

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

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

Related Topics

How to Configure Wireshark, on page 12
Adding or Modifying Capture Point Parameters, on page 17
Deleting Capture Point Parameters, on page 18
Deleting a Capture Point, on page 19
Activating and Deactivating a Capture Point, on page 20

Adding or Modifying Capture Point Parameters

Although listed in sequence, the steps to specify values for the parameters can be executed in any order. You can also specify them in one, two, or several lines. Except for attachment points, which can be multiple, you can replace any value with a more recent value by redefining the same option.

In privileged EXEC mode, follow these steps to modify a capture point's parameters.

Before you begin

A capture point must be defined before you can use these instructions.

Procedure

	Command or Action	Purpose

Step 1	**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.
		Note If you are defining a wireless capture point using a CAPWAP tunneling interface, this command will have no effect, so it should not be used.

Step 2	**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 to be retained by Wireshark (400).

Step 3	**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.

Step 4	**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 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 you defined previously.

	Command or Action	Purpose
	`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`	

Examples

Modifying Parameters

Associating or Disassociating a Capture File

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

Specifying a Memory Buffer Size for Packet Burst Handling

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

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

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

What to do next

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

Related Topics

- How to Configure Wireshark, on page 12
- Defining a Capture Point, on page 12
- Deleting Capture Point Parameters, on page 18
- Deleting a Capture Point, on page 19

Deleting Capture Point Parameters

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

In privileged EXEC mode, follow these steps to delete a capture point's parameters.

Before you begin

A capture point parameter must be defined before you can use these instructions to delete it.

Procedure

	Command or Action	Purpose
Step 1	**no monitor capture** {*capture-name*} **match** **Example:** `Device# no monitor capture mycap match`	Deletes all filters defined on capture point (mycap).

Step 2	**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 to be retained by Wireshark. It leaves other specified limits in place.
		Deletes all limits on Wireshark.

Step 3	**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 will only display them.
		Deletes the file location association. The file location will no longer be associated with the capture point. However, other defined fille association will be unaffected by this action.

Step 4	**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. This command can be run at any point in the procedure to see what parameters are associated with a capture point.

What to do next

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

Related Topics

How to Configure Wireshark, on page 12
Defining a Capture Point, on page 12
Adding or Modifying Capture Point Parameters, on page 17

Deleting a Capture Point

In privileged EXEC mode, follow these steps to delete a capture point.

Before you begin

A capture point must be defined before you can use these instructions to delete it.

Procedure

	Command or Action	Purpose

Step 1	**no monitor capture** {*capture-name*}
	
	Example:
	`Device# no monitor capture mycap`
		Deletes the specified capture point (mycap).

Step 2	**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 does not exist because it has been deleted.

What to do next

You can define a new capture point with the same name as the one you deleted. These instructions are usually performed when one wants to start over with defining a capture point.

Related Topics

How to Configure Wireshark, on page 12
Defining a Capture Point, on page 12
Adding or Modifying Capture Point Parameters, on page 17

Activating and Deactivating a Capture Point

In privileged EXEC mode, follow these steps to activate or deactivate a capture point.

Before you begin

A capture point cannot be activated unless an attachment point and a core system filter have been defined and the associated filename (if any) does not already exist. A capture point with no associated filename can only be activated to display. If no capture or display filters are specified, all of the packets captured by the core system filter are displayed. The default display mode is brief.

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

Procedure

	Command or Action	Purpose
Step 1	**monitor capture** {*capture-name*} **start** [**display** [**display-filter** *filter-string*]] [**brief** | **detailed** | **dump**] **Example:**	Activates a capture point and filters the display, so only packets containing "stp" are displayed.

	Command or Action	Purpose
	`Device# monitor capture mycap start display display-filter "stp"`	
Step 2	**monitor capture** {*capture-name*} **stop** **Example:** `Device# monitor capture name stop`	Deactivates a capture point.

Related Topics

How to Configure Wireshark, on page 12
Defining a Capture Point, on page 12

Clearing the Capture Point Buffer

In privileged EXEC mode, follow these steps to clear the buffer contents or save them to an external file for storage.

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

Procedure

	Command or Action	Purpose
Step 1	**monitor capture** {*capture-name*} [**clear** | **export** *filename*] **Example:** `Device# monitor capture mycap clear`	Clears capture buffer contents or stores the packets to a 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
```

Related Topics

How to Configure Wireshark, on page 12

Monitoring Wireshark

The commands in this table are used to monitor Wireshark.

Command	Purpose
show monitor capture [*capture-name*]	Displays the capture point state so that you can see what capture points are defined, what their attributes are, and whether they are active. When capture point name is specified, it displays specific capture point's details.
show monitor capture [*capture-name* **parameter**]	Displays the capture point parameters.
show capwap summary	Displays all the CAPWAP tunnels on the switch. Use this command to determine which CAPWAP tunnels are available to use for a wireless capture.

Configuration Examples for Wireshark

Example: Displaying a Brief Output from a .pcap File

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

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

  1   0.000000   10.1.1.140 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  2   1.000000   10.1.1.141 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  3   2.000000   10.1.1.142 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  4   3.000000   10.1.1.143 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  5   4.000000   10.1.1.144 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  6   5.000000   10.1.1.145 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  7   6.000000   10.1.1.146 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  8   7.000000   10.1.1.147 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  9   8.000000   10.1.1.148 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 10   9.000000   10.1.1.149 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 11  10.000000   10.1.1.150 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 12  11.000000   10.1.1.151 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 13  12.000000   10.1.1.152 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

```
14  13.000000   10.1.1.153 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
15  14.000000   10.1.1.154 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
16  15.000000   10.1.1.155 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
17  16.000000   10.1.1.156 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
18  17.000000   10.1.1.157 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
19  18.000000   10.1.1.158 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
20  19.000000   10.1.1.159 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
21  20.000000   10.1.1.160 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
22  21.000000   10.1.1.161 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
23  22.000000   10.1.1.162 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
24  23.000000   10.1.1.163 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
25  24.000000   10.1.1.164 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
26  25.000000   10.1.1.165 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
27  26.000000   10.1.1.166 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
28  27.000000   10.1.1.167 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
29  28.000000   10.1.1.168 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
30  29.000000   10.1.1.169 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
31  30.000000   10.1.1.170 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
32  31.000000   10.1.1.171 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
33  32.000000   10.1.1.172 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
34  33.000000   10.1.1.173 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
35  34.000000   10.1.1.174 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
36  35.000000   10.1.1.175 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
37  36.000000   10.1.1.176 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
38  37.000000   10.1.1.177 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
39  38.000000   10.1.1.178 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
40  39.000000   10.1.1.179 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
41  40.000000   10.1.1.180 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
42  41.000000   10.1.1.181 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
43  42.000000   10.1.1.182 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
44  43.000000   10.1.1.183 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
45  44.000000   10.1.1.184 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

```
 46  45.000000   10.1.1.185 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 47  46.000000   10.1.1.186 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 48  47.000000   10.1.1.187 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 49  48.000000   10.1.1.188 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 50  49.000000   10.1.1.189 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 51  50.000000   10.1.1.190 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 52  51.000000   10.1.1.191 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 53  52.000000   10.1.1.192 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 54  53.000000   10.1.1.193 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 55  54.000000   10.1.1.194 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 56  55.000000   10.1.1.195 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 57  56.000000   10.1.1.196 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 58  57.000000   10.1.1.197 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 59  58.000000   10.1.1.198 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Example: Displaying Detailed Output from a .pcap File

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

```
Device# show monitor capture file flash:mycap.pcap detailed

Frame 1: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits)
    Arrival Time: Mar 21, 2012 14:35:09.111993000 PDT
    Epoch Time: 1332365709.111993000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 256 bytes (2048 bits)
    Capture Length: 256 bytes (2048 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:data]
Ethernet II, Src: 00:00:00:00:03:01 (00:00:00:00:03:01), Dst: 54:75:d0:3a:85:3f
(54:75:d0:3a:85:3f)
    Destination: 54:75:d0:3a:85:3f (54:75:d0:3a:85:3f)
        Address: 54:75:d0:3a:85:3f (54:75:d0:3a:85:3f)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Source: 00:00:00:00:03:01 (00:00:00:00:03:01)
        Address: 00:00:00:00:03:01 (00:00:00:00:03:01)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: IP (0x0800)
    Frame check sequence: 0x03b07f42 [incorrect, should be 0x08fcee78]
Internet Protocol, Src: 10.1.1.140 (10.1.1.140), Dst: 20.1.1.2 (20.1.1.2)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 238
    Identification: 0x0000 (0)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x5970 [correct]
        [Good: True]
        [Bad: False]
    Source: 10.1.1.140 (10.1.1.140)
    Destination: 20.1.1.2 (20.1.1.2)
User Datagram Protocol, Src Port: 20001 (20001), Dst Port: 20002 (20002)
    Source port: 20001 (20001)
    Destination port: 20002 (20002)
    Length: 218
    Checksum: 0x6e2b [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
Data (210 bytes)

0000  00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f   ................
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
0030  30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f   0123456789:;<=>?
0040  40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f   @ABCDEFGHIJKLMNO
0050  50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f   PQRSTUVWXYZ[\]^_
0060  60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f   `abcdefghijklmno
0070  70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f   pqrstuvwxyz{|}~.
0080  80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f   ................
0090  90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f   ................
00a0  a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af   ................
00b0  b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf   ................
00c0  c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf   ................
00d0  d0 d1                                             ..
    Data: 000102030405060708090a0b0c0d0e0f1011121314151617...
    [Length: 210]

Frame 2: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits)
    Arrival Time: Mar 21, 2012 14:35:10.111993000 PDT
Example: Displaying a Hexadecimal Dump Output from a .pcap File
You can display the hexadecimal dump output by entering:
Switch# show monitor capture file bootflash:mycap.pcap dump
  1   0.000000   10.1.1.140 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

0000  54 75 d0 3a 85 3f 00 00 00 00 03 01 08 00 45 00   Tu.:.?........E.
0010  00 ee 00 00 00 00 40 11 59 70 0a 01 01 8c 14 01   ......@.Yp......
0020  01 02 4e 21 4e 22 00 da 6e 2b 00 01 02 03 04 05   ..N!N"..n+......
0030  06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   ................
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45   6789:;<=>?@ABCDE
0070  46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55   FGHIJKLMNOPQRSTU
0080  56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65   VWXYZ[\]^_`abcde
0090  66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75   fghijklmnopqrstu
00a0  76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85   vwxyz{|}~.......
00b0  86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95   ................
```

```
00c0  96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5   ................
00d0  a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5   ................
00e0  b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5   ................
00f0  c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 03 b0 7f 42   ...............B

  2   1.000000   10.1.1.141 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

0000  54 75 d0 3a 85 3f 00 00 00 00 03 01 08 00 45 00   Tu.:.?........E.
0010  00 ee 00 00 00 00 40 11 59 6f 0a 01 01 8d 14 01   ......@.Yo......
0020  01 02 4e 21 4e 22 00 da 6e 2a 00 01 02 03 04 05   ..N!N"..n*......
0030  06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   ................
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45   6789:;<=>?@ABCDE
0070  46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55   FGHIJKLMNOPQRSTU
0080  56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65   VWXYZ[\]^_`abcde
0090  66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75   fghijklmnopqrstu
00a0  76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85   vwxyz{|}~.......
00b0  86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95   ................
00c0  96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5   ................
00d0  a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5   ................
00e0  b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5   ................
00f0  c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 95 2c c3 3f   .............,.?

  3   2.000000   10.1.1.142 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

0000  54 75 d0 3a 85 3f 00 00 00 00 03 01 08 00 45 00   Tu.:.?........E.
0010  00 ee 00 00 00 00 40 11 59 6e 0a 01 01 8e 14 01   ......@.Yn......
0020  01 02 4e 21 4e 22 00 da 6e 29 00 01 02 03 04 05   ..N!N"..n)......
0030  06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   ................
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45   6789:;<=>?@ABCDE
0070  46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55   FGHIJKLMNOPQRSTU
0080  56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65   VWXYZ[\]^_`abcde
0090  66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75   fghijklmnopqrstu
00a0  76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85   vwxyz{|}~.......
00b0  86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95   ................
00c0  96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5   ................
00d0  a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5   ................
00e0  b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5   ................
00f0  c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 6c f8 dc 14   ............l...

  4   3.000000   10.1.1.143 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

0000  54 75 d0 3a 85 3f 00 00 00 00 03 01 08 00 45 00   Tu.:.?........E.
0010  00 ee 00 00 00 00 40 11 59 6d 0a 01 01 8f 14 01   ......@.Ym......
0020  01 02 4e 21 4e 22 00 da 6e 28 00 01 02 03 04 05   ..N!N"..n(......
0030  06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   ................
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
Example: Displaying Packets from a .pcap File with a Display Filter
You can display the .pcap file packets output by entering:
Switch# show monitor capture file bootflash:mycap.pcap display-filter "ip.src == 10.1.1.140"
 dump
  1   0.000000   10.1.1.140 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

0000  54 75 d0 3a 85 3f 00 00 00 00 03 01 08 00 45 00   Tu.:.?........E.
0010  00 ee 00 00 00 00 40 11 59 70 0a 01 01 8c 14 01   ......@.Yp......
0020  01 02 4e 21 4e 22 00 da 6e 2b 00 01 02 03 04 05   ..N!N"..n+......
0030  06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   ................
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45   6789:;<=>?@ABCDE
```

```
0070  46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55   FGHIJKLMNOPQRSTU
0080  56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65   VWXYZ[\]^_`abcde
0090  66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75   fghijklmnopqrstu
00a0  76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85   vwxyz{|}~.......
00b0  86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95   ................
00c0  96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5   ................
00d0  a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5   ................
00e0  b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5   ................
00f0  c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 03 b0 7f 42   ...............B
```

Example: Simple Capture and Display

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

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

```
Device# monitor capture mycap interface GigabitEthernet1/0/1 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 100
Device# monitor capture mycap buffer size 100
```

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

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

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap buffer size 100
   monitor capture mycap limit packets 100 duration 60

Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/1, Direction: in
   Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  Buffer Details:
   Buffer Type: LINEAR (default)
   Buffer Size (in MB): 100
  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 100
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packets per second: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

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

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

Step 4: Delete the capture point by entering:

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

Example: Simple Capture and Store

This example shows how to capture packets to a filter:

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

```
Device# monitor capture mycap interface GigabitEthernet1/0/1 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 100
Device# monitor capture mycap file location flash:mycap.pcap
```

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

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

Device# show monitor capture mycap

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

Step 3: Launch packet capture by entering:

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

Step 4: After sufficient time has passed, stop the capture by entering:

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

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

The mycap.pcap file now contains the captured packets.

Step 5: Display the packets by entering:

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

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

Step 6: Delete the capture point by entering:

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

Example: Using Buffer Capture

This example shows how to use buffer capture:

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

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

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

```
Device# show monitor capture mycap

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

Step 3: Display the packets in the buffer by entering:

```
Device# show monitor capture mycap buffer brief

  0.000000   10.1.1.215 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000   10.1.1.216 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000   10.1.1.217 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000   10.1.1.218 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000   10.1.1.219 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000   10.1.1.220 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000   10.1.1.221 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000   10.1.1.222 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000   10.1.1.223 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000   10.1.1.224 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 10.000000   10.1.1.225 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 11.000000   10.1.1.226 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 12.000000   10.1.1.227 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 13.000000   10.1.1.228 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 14.000000   10.1.1.229 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 15.000000   10.1.1.230 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 16.000000   10.1.1.231 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 17.000000   10.1.1.232 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 18.000000   10.1.1.233 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 19.000000   10.1.1.234 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 20.000000   10.1.1.235 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 21.000000   10.1.1.236 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Notice that the packets have been buffered.

Step 4: Display the packets in other display modes.

```
Device# show monitor capture mycap buffer detailed

Frame 1: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits)
    Arrival Time: Apr 15, 2012 15:50:02.398966000 PDT
    Epoch Time: 1334530202.398966000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 256 bytes (2048 bits)
    Capture Length: 256 bytes (2048 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:data]
Ethernet II, Src: 00:00:00:00:03:01 (00:00:00:00:03:01), Dst: 54:75:d0:3a:85:3f
(54:75:d0:3a:85:3f)
    Destination: 54:75:d0:3a:85:3f (54:75:d0:3a:85:3f)
        Address: 54:75:d0:3a:85:3f (54:75:d0:3a:85:3f)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Source: 00:00:00:00:03:01 (00:00:00:00:03:01)
        Address: 00:00:00:00:03:01 (00:00:00:00:03:01)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
…
Device# show monitor capture mycap buffer dump

  0.000000   10.1.1.215 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

0000  54 75 d0 3a 85 3f 00 00 00 00 03 01 08 00 45 00   Tu.:.?........E.
0010  00 ee 00 00 00 00 40 11 59 25 0a 01 01 d7 14 01   ......@.Y%......
```

```
0020  01 02 4e 21 4e 22 00 da 6d e0 00 01 02 03 04 05   ..N!N"..m.......
0030  06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15   ................
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25   .......... !"#$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35   &'()*+,-./012345
0060  36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45   6789:;<=>?@ABCDE
0070  46 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55   FGHIJKLMNOPQRSTU
0080  56 57 58 59 5a 5b 5c 5d 5e 5f 60 61 62 63 64 65   VWXYZ[\]^_`abcde
0090  66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75   fghijklmnopqrstu
00a0  76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85   vwxyz{|}~.......
00b0  86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95   ................
00c0  96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5   ................
00d0  a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5   ................
00e0  b6 b7 b8 b9 ba bb bc bd be bf c0 c1 c2 c3 c4 c5   ................
00f0  c6 c7 c8 c9 ca cb cc cd ce cf d0 d1 03 3e d0 33   ............>.3
```

Step 5a: Clear the buffer by entering:

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

Step 5b: Wait for 10 seconds.

Step 5c: Stop the traffic by entering:

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

Step 6: Confirm that the same set of packets are displayed after this time gap by entering:

```
Device# show monitor capture mycap buffer brief

  0.000000     10.1.1.2 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000     10.1.1.3 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000     10.1.1.4 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000     10.1.1.5 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000     10.1.1.6 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000     10.1.1.7 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000     10.1.1.8 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000     10.1.1.9 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.10 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.11 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 7: Wait for 10 seconds, then confirm that the same set of packets are displayed after this time gap by entering:

```
Device# show monitor capture mycap buffer brief

  0.000000     10.1.1.2 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000     10.1.1.3 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000     10.1.1.4 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000     10.1.1.5 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000     10.1.1.6 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000     10.1.1.7 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000     10.1.1.8 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000     10.1.1.9 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.10 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.11 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 8: Repeat Step 7.

Step 9: Clear the buffer by entering:

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

Step 10: Confirm that the buffer is now empty by entering:

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

Step 11: Wait about 10 seconds, then display the buffer contents by entering:

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

Step 12: Restart the traffic, wait for 10 seconds, then display the buffer contents by entering:

```
Device# monitor capture mycap start
wait for 10 seconds...
Device# show monitor capture mycap buffer brief

  0.000000     10.1.1.2 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000     10.1.1.3 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000     10.1.1.4 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000     10.1.1.5 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000     10.1.1.6 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000     10.1.1.7 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000     10.1.1.8 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000     10.1.1.9 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.10 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.11 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 13: Store the buffer contents to the mycap1.pcap file in the internal flash: storage device by entering:

```
Device# monitor capture mycap export flash:mycap1.pcap
Exported Successfully
```

Step 14: Check that the file has been created and that it contains the packets by entering:

```
Device# dir flash:mycap1.pcap
Directory of flash:/mycap1.pcap

14758  -rw-       20152  Apr 15 2012 16:00:28 -07:00  mycap1.pcap

831541248 bytes total (831340544 bytes free)
Device# show monitor capture file flash:mycap1.pcap brief
  1   0.000000     10.1.1.2 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  2   1.000000     10.1.1.3 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  3   2.000000     10.1.1.4 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  4   3.000000     10.1.1.5 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  5   4.000000     10.1.1.6 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  6   5.000000     10.1.1.7 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  7   6.000000     10.1.1.8 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  8   7.000000     10.1.1.9 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

  9   8.000000    10.1.1.10 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 10   9.000000    10.1.1.11 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 11  10.000000    10.1.1.12 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 12  11.000000    10.1.1.13 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 13  12.000000    10.1.1.14 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 14  13.000000    10.1.1.15 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 15  14.000000    10.1.1.16 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

 16  15.000000    10.1.1.17 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 15: Stop the packet capture and display the buffer contents by entering:

```
Device# monitor capture mycap stop
Device# show monitor capture mycap buffer brief

  0.000000     10.1.1.2 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000     10.1.1.3 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000     10.1.1.4 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000     10.1.1.5 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000     10.1.1.6 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000     10.1.1.7 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000     10.1.1.8 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000     10.1.1.9 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.10 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.11 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 10.000000    10.1.1.12 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 11.000000    10.1.1.13 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 16: Clear the buffer and then try to display packets from the buffer by entering:

```
Device# monitor capture mycap clear
Device# show monitor capture mycap buffer brief
```

Step 17: Delete the capture point by entering:

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

Example: Capture Sessions

```
Device# monitor capture mycap start display display-filter "stp"
  0.000000 20:37:06:cf:08:b6 -> 01:80:c2:00:00:00 STP Conf. Root = 32768/100/20:37:06:ce:f0:80
  Cost = 0  Port = 0x8136
  2.000992 20:37:06:cf:08:b6 -> 01:80:c2:00:00:00 STP Conf. Root = 32768/100/20:37:06:ce:f0:80
  Cost = 0  Port = 0x8136
  2.981996 20:37:06:cf:08:b6 -> 20:37:06:cf:08:b6 LOOP Reply
  4.000992 20:37:06:cf:08:b6 -> 01:80:c2:00:00:00 STP Conf. Root = 32768/100/20:37:06:ce:f0:80
  Cost = 0  Port = 0x8136
  6.000000 20:37:06:cf:08:b6 -> 01:80:c2:00:00:00 STP Conf. Root = 32768/100/20:37:06:ce:f0:80
  Cost = 0  Port = 0x8136
  7.998001 20:37:06:cf:08:b6 -> 01:80:c2:00:00:00 STP Conf. Root = 32768/100/20:37:06:ce:f0:80
  Cost = 0  Port = 0x8136
  9.998001 20:37:06:cf:08:b6 -> 01:80:c2:00:00:00 STP Conf. Root = 32768/100/20:37:06:ce:f0:80
  Cost = 0  Port = 0x8136
Capture test is not active Failed to Initiate Wireshark
Device# show monitor capture mycap parameter
   monitor capture mycap control-plane both
   monitor capture mycap match any
   monitor capture mycap file location flash:mycap1.1 buffer-size 90
   monitor capture mycap limit duration 10

Device# monitor capture mycap start display display-filter "udp.port == 20002"
A file by the same capture file name already exists, overwrite?[confirm] [ENTER]
after a minute or so...
Capture mycap is not active Failed to Initiate Wireshark
```

```
*Oct 13 15:00:44.649: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
*Oct 13 15:00:46.657: %BUFCAP-6-DISABLE_ASYNC: Capture Point mycap disabled. Rea
son : Wireshark Session Ended

Device# monitor capture mycap start display display-filter "udp.port == 20002" dump
A file by the same capture file name already exists, overwrite?[confirm]
after a minute or so...
Capture mycap is not active Failed to Initiate Wireshark
*Oct 13 15:00:44.649: %BUFCAP-6-ENABLE: Capture Point mycap enabled.
*Oct 13 15:00:46.657: %BUFCAP-6-DISABLE_ASYNC: Capture Point mycap disabled. Rea
son : Wireshark Session Ended

Device# no monitor capture mycap file
Device# monitor capture mycap start display display-filter "udp.port == 20002" dump
 Please associate capture file/buffer
Unable to activate Capture.

Device# monitor capture mycap start display display-filter "udp.port == 20002"
 Please associate capture file/buffer
Unable to activate Capture.

Device# monitor capture mycap start display detailed
 Please associate capture file/buffer
Unable to activate Capture.
```

Example: Capture and Store in Lock-step Mode

This example captures live traffic and stores the packets in lock-step mode.

Note The capture rate might be slow for the first 15 seconds. If possible and necessary, start the traffic 15 seconds after the capture session starts.

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

```
Device# monitor capture mycap interface GigabitEthernet1/0/1 in
Device# monitor capture mycap match ipv4 any any
Device# monitor capture mycap limit duration 60 packets 100
Device# monitor capture mycap file location flash:mycap.pcap buffer-size 64
```

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

```
Device# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet1/0/1 in
   monitor capture mycap file location flash:mycap.pcap buffer-size 64
   monitor capture mycap limit packets 100 duration 60

Device# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet1/0/1, Direction: in
   Status : Inactive
  Filter Details:
   Filter not attached
  Buffer Details:
   Buffer Type: LINEAR (default)
  File Details:
   Associated file name: flash:mycap.pcap
   Size of buffer(in MB): 64
  Limit Details:
   Number of Packets to capture: 100
   Packet Capture duration: 60
   Packet Size to capture: 0 (no limit)
   Packets per second: 0 (no limit)
   Packet sampling rate: 0 (no sampling)
```

Step 3: Launch packet capture by entering:

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

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

Step 4: Display the packets by entering:

```
Device# show monitor capture file flash:mycap.pcap
  0.000000    10.1.1.30 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000    10.1.1.31 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000    10.1.1.32 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000    10.1.1.33 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000    10.1.1.34 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000    10.1.1.35 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000    10.1.1.36 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000    10.1.1.37 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.38 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.39 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 5: Delete the capture point by entering:

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

Example: Simple Capture and Store of Packets in Egress Direction

This example shows how to capture packets to a filter:

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

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

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

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

Device# show monitor capture mycap

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

Step 3: Launch packet capture by entering:

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

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

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

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

The mycap.pcap file now contains the captured packets.

Step 4: Display the packets by entering:

```
Device# show monitor capture file flash:mycap.pcap
  0.000000    10.1.1.30 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000    10.1.1.31 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000    10.1.1.32 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000    10.1.1.33 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000    10.1.1.34 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000    10.1.1.35 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000    10.1.1.36 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000    10.1.1.37 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000    10.1.1.38 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000    10.1.1.39 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

Step 5: Delete the capture point by entering:

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

Additional References

Related Documents

Related Topic	Document Title
General Packet Filtering	For general packet filtering, refer to: Display Filter Reference

Error Message Decoder

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

Standards and RFCs

Standard/RFC	Title

MIBs

MIB	MIBs Link
All supported MIBs for this release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Related Topics

Filters, on page 4

Feature History and Information for WireShark

Release	Modification
Cisco IOS XE 3.3SE	This feature was introduced.