CHAPTER 56

Configuring Wireshark

Note Wireshark is only supported on Supervisor Engine 7-E, Supervisor Engine 7L-E, and Catalyst 4500X-32.

Beginning with Cisco IOS Release XE 3.3.0SG in the IP Base and Enterprise Services feature sets, the Catalyst 4500 series switch supports Wireshark, a packet analyzer program, formerly known as Ethereal, which supports multiple protocols and presents information in a text-based user interface.

This chapter includes these sections:

Note For complete syntax and usage information for the switch commands used in this chapter, first look at the *Cisco Catalyst 4500 Series Switch Command Reference* and related publications at this location:

http://www.cisco.com/en/US/products/hw/switches/ps4324/index.html

If the command is not found in the Catalyst 4500 Series Switch Command Reference, it will be found in the larger Cisco IOS library. Refer to the *Cisco IOS Command Reference* and related publications at this location:

http://www.cisco.com/en/US/products/ps6350/index.html

About Wireshark

To understand what happens inside a network requires the ability to capture and analyze traffic. Prior to Cisco IOS Release XE 3.3.0SG, the Catalyst 4500 series switch offered only two features to address this need: SPAN and **debug platform packet**. Both are limited. 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 switch and only works on packets that stem from the software process-forwarding path. Although it has limited local display capabilities, it has 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.

Note In Cisco IOS Release XE 3.3.0SG, global packet capture on Wireshark is not supported.

These sections describe some key concepts for Wireshark:

Capture Points

A capture point is the central policy definition of the Wireshark feature. The point describes all the characteristics associated with a given instance of Wireshark: what 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 may also be manually or automatically deactivated or stopped.

Multiple capture points may be defined and activated simultaneously.

Attachment Points: Interfaces and traffic directions

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

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

Note Wireshark on the Catalyst 4500 series switch does not use the syntax of the capture filter.

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

Core System Filter

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

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 would limit the ability of network administrators to monitor and analyze traffic. To address this situation, Wireshark supports explicit specification of core system filter match criteria from the EXEC mode CLI. The disadvantage is that the match criteria that you can specify is a limited subset of what class map supports, such as MAC, IP source and destination addresses, ether-type, IP protocol, and TCP/UDP source and destination ports.

If you prefer to use configuration mode, you can define ACLs or have class maps refer capture points to them. Explicit and ACL-based match criteria are used internally to construct class maps and policy maps. These implicitly constructed class maps are not reflected in the switch running-config and are not NVGEN'd.

Note The configuration of ACL and class-map are part of the system and not aspects of the Wireshark feature

Capture Filter

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

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

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

Display Filter

With the display filter, you can direct Wireshark to further narrow the set of packets to display when decoding and displaying from a .pcap file. Because the syntax of the display filter is identical to the capture filter, the display filter is superfluous if a capture filter is also defined.

For more details on the syntax of capture and display filters, go to

http://wiki.wireshark.org/DisplayFilters

Actions

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

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

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

Storing 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 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 packet are discarded to accommodate the new packet. Although the buffer can also be cleared when needed, this mode is mainly used for debugging network traffic.

Storing Captured Packets to a .pcap File

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

- Catalyst 4500 series switch on-board flash storage (bootflash:)
- external flash disk (slot:)
- USB drive (usb0:)

Note Do not attempt to use Wireshark with any other devices.

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 3.3.0SG, 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.

Decoding and Displaying Packets

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

Displaying Live Traffic

Wireshark receives copies of packets from the Catalyst 4500 series switch core system. Wireshark applies its capture and display filters to discard uninteresting packets, and then decodes and displays the remaining packets.

Displaying from .pcap File

Wireshark can decode and display packets from a previously stored .pcap file and direct the display filter to selectively displayed packets. A capture filter is not applicable in this situation.

Storing and Displaying Packets

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

Activating and Deactivating Wireshark Capture Points

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 sanity 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 generates an error.

The capture and display filters are specified as needed.

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

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

Feature Interactions

This section describes how Wireshark features function in the Catalyst 4500 series switch environment:

- Layer 2 security features—Packets that are dropped by Layer 2 security features (such as port security, MAC address filtering, and spanning tree) are not captured by Wireshark. This differs from the behavior of SPAN.
- Classification-based security features—Packets that are dropped by input classification-based security features (such as ACLs and IPSG) are not caught by Wireshark capture points that are connected to attachment points at the same layer. In contrast, packets that are dropped by output classification-based security features are caught by Wireshark capture points that are connected to attachment points at the same layer. The logical model is that the Wireshark attachment point occurs after the security feature lookup on the input side, and symmetrically before the security feature lookup on the output side.

  Wireshark capture policies connected to Layer 2 attachment points in the input direction capture packets dropped by Layer 3 classification-based security features. Symmetrically, Wireshark capture policies attached to Layer 3 attachment points in the output direction capture packets dropped by Layer 2 classification-based security features.
- Routed ports and Layer 3 port channels—When a routed port or Layer 3 port channel is used as a Wireshark attachment point, the The policy that is applied to capture the packets is treated as attached at Layer 3. Wireshark only captures packets that are being routed by the interface.
- VLANs—When a VLAN is used as a Wireshark attachment point, packets are captured in both input and output directions. A packet that is bridged in the VLAN generates two copies, one on input and one on output.
- Private VLANs—Secondary PVLANs are disallowed as Wireshark attachment points. Using a primary PVLAN as a Wireshark attachment point enables capture of packets in the primary PVLAN and all associated secondary PVLANs. The entire PV domain becomes the attachment point.

- Redirection features—In the input direction, features traffic redirected by Layer 3 (such as PBR and WCCP), are logically later than Layer 3 Wireshark attachment points. Wireshark captures these packets even though they might later be redirected out another Layer 3 interface. Symmetrically, output features redirected by Layer 3 (such as egress WCCP) are logically prior to Layer 3 Wireshark attachment points, and Wireshark will not capture them.
- Classification copy features—Features that generate copies of packets from the role-based and Security lookup types are compatible with Wireshark. Multiple copies of these packets are generated.
- 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.

There are four classification results for input and output classification. In the input direction, they are ordered role-based, security, QoS, and forwarding override. In the output direction they are ordered forwarding override, role-based, security, and QoS.

On the input side, the Wireshark capture feature is placed in the forwarding override result type, prioritized above the other FO features (such as multicast local source capture, PBR and ingress WCCP). The packets captured by Wireshark are before any redirection by PBR or WCCP. Because security ACLs are applied ahead of FO-related features, packets that are dropped by security ACLs are not captured by Wireshark.

On the output side, the Wireshark capture feature is placed in the forwarding override result type, prioritized below the other FO features (such as egress WCCP). Wireshark captures packets only if the other egress FO features do not apply.

Configuring Wireshark

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.

The following sections describe how to configure Wireshark:

Default Wireshark Configuration

Table 56-1 shows the default Wireshark configuration.

Table 56-1 Default Wireshark Configuration

Feature	Default Setting
Duration	No limit
Packets	No limit
Packet-length	No limit (full packet)

Table 56-1 Default Wireshark Configuration

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

Wireshark Configuration Guidelines

When configuring Wireshark, ensure the following:

- Traffic is active on the interfaces the Wireshark policy is applied on.
- Filter rules match the traffic.
- Mandatory parameters are configured.

Defining, Modifying, or Deleting a Capture Point

Although listed in sequence, the steps to specify values for the options 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 respecifying the same option, in the following order:

Step 1 Define the name that identifies the capture point.

Step 2 Specify the attachment point with which the capture point is associated.

Multiple attachment points can be specified. Range support is also available both for adding and removing attachment points.

Step 3 Define the core system filter, defined either explicitly, through ACL or through a class map.

Step 4 Specify the session limit (in seconds or packets captured).

Step 5 Specify the packet segment length to be retained by Wireshark.

Step 6 Specify the file association, if the capture point intends to capture packets rather than merely display them.

Step 7 Specify the size of the memory buffer used by Wireshark to handle traffic bursts.

To filter the capture point, use the following commands:

Command	Purpose
[**no**] **monitor capture mycap match** {**any** | **mac** *mac-match-string* | **ipv4** *ipv4-match-string* | **ipv6** *ipv6-match-string*}	Defines an explicitly in-line core filter. To remove the filter, use the **no** form of this command.

Command	Purpose
[**no**] **monitor capture mycap match** *mac* {*src-mac-addr src-mac-mask* | **any** | **host** *src-mac-addr*} | {*dest-mac-addr dest-mac-mask* | **any** | **host** *dest-mac-addr*}	Specifies use of a filter for MAC. To remove the filter, use the **no** form of this command.
[**no**] **monitor capture mycap match** {**ipv4** | **ipv6**} [*src-prefix/length* | **any** | **host** *src-ip-addr*] [*dest-prefix/length* | **any** | **host** *dest-ip-addr*] [**no**] **monitor capture mycap match** {**ipv4** | **ipv6**} **proto** {**tcp** | **udp**} [*src-prefix/length* | **any** | **host** *src-ip-addr*] [**eq** | **gt** | **lt** | **neq** <*0-65535*>] [*dest-prefix/length* | **any** | **host** *dest-ip-addr*] [**eq** | **gt** | **lt** | **neq** <*0-65535*>]	Specifies a filter for IPv4/IPv6, use one of the formats. To remove the filters, use the **no** form of this command.

To define a capture point, use the following commands:

Command	Purpose
monitor capture *name* [{**interface** *name* | **vlan** *num* | **control-plane**} {**in** | **out** | **both**}	Specifies one or more attachment points with direction. To remove the attachment point, use the **no** form of this command.
monitor capture *name* **[[file location** *filename* [**buffer-size** <*1-100*>] [**ring** <2-10>] [**size** <*1-100*>]] | [**buffer** [**circular**] **size** <*1-100*>]]	Specifies the capture destination. To remove the details, use the **no** form of this command.
[no] monitor capture *name* **limit** {**duration** *seconds*] [**packet-length** *size*] [**packets** *num*]	Specifies capture limits. To remove the limits, use the **no** form of this command.

To clear the buffer contents, use the following command

Command	Purpose
monitor capture [clear | export *filename***]**	Clears capture buffer contents or stores the packets to a file.

To start and stop a capture point, use the following command:

Command	Purpose
monitor capture *name* **start** [**capture-filter** *filter-string*] [**display** [**display-filter** *filter-string*]] [**brief** | **detailed** | **dump** | **stop**]	To start or stop a capture point, use the **monitor capture** command.

Examples

Associating/disassociating a capture file

```
Switch# monitor capture point mycap file location bootdisk:mycap.pcap

Switch# no monitor capture mycap file
```

Specifying a memory buffer size for packet burst handling

```
Switch# monitor capture mycap buffer-size 1000000
```

Defining an explicit core system filter to match both IPv4 and IPv6 TCP traffic

```
Switch# monitor capture mycap match any protocol tcp
```

Defining a core system filter using an existing ACL or class-map

```
Switch# monitor capture mycap match access-list myacl

Switch# monitor capture mycap match class-map mycm
```

Activating and Deactivating a Capture Point

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**.

To activate or deactivate a capture point, perform these tasks:

Command	Purpose
monitor capture *name* **start** [**capture-filter** *filter-string*] [**display** [**display-filter** *filter-string*]] [**brief** | **detailed** | **dump**]	Activates a capture point.
monitor capture *name* **stop**	Deactivates a capture point.

Example:
`Switch# monitor capture mycap start capture-filter "net 10.1.1.0 0.0.0.255 and port 80"`
 	
`Switch# monitor capture mycap start display display-filter "net 10.1.1.0 0.0.0.255 and port 80"`	

Guidelines and Restrictions

- When packet capture is enabled in the input direction, the matching packets undergo software-based lookup in the CPU for the first 15 seconds. During this time, CPU usage is high and capture rate is low.
- When packet capture is enabled in the output direction, packets are not captured in the first 15 seconds.
- Packets captured in the output direction of an interface might not reflect the changes made by switch rewrite (includes TTL, VLAN tag, CoS, checksum, and MAC addresses).
- Capturing at a physical port that belongs to another logical port may not be supported. For example, capturing at EtherChannel member ports is not supported.
- Limiting circular file storage by file size is not supported.

- Wireshark cannot capture IPv6 packets if the capture point's class-map filter is attempting to match one of the following:
  - Extension headers followed by Hop-by-hop header (as per CSCtt16385)
  - DSCP values (as per CSCtx75765)

Best Practices

Consider the following best practices:

- 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).
- Limit packet capture with parameters of the **capture point** command (like packet number and capture duration).
- 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, 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.
- Do not leave a capture session enabled and unattended for a long period of time, because unanticipated bursts of traffic could increase the CPU usage.
- 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.
- Limit the number of Wireshark instances to two or less to avoid CPU or memory resource drain.

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

- Whenever an ACL is installed or modified on a switch in the ingress direction, for the first 15 seconds, the software ignores packet classification details sent by the hardware. Instead, it uses software-based classification for the packets received by CPU. So, during this period, the system can only capture fewer packets (compared to the time after the first 15 seconds) and CPU usage will be high.

Note In the egress direction, packets are not captured for the first 15 seconds.

- 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 use the default buffer size, packets may be dropped. Increase buffer size and avoid packet loss.
  - Writing to flash disk is a CPU-intensive operation, so the capture rate may not be sufficient.
  - 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 has ended, a session transitions to the process phase, in which it stops accepting packets and starts processing packets in the buffer. With the second approach (lock-step mode), a higher capture throughput can be achieved.
  - The streaming capture mode supports approximately 1500 pps; lock-step mode supports roughly 45 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. 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.
- Do not launch a capture session with ring files or capture buffer and leave it unattended for a long time. This may lead to performance or system health issues because of high CPU or memory usage.
- 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.

Notes Specific to the Wireshark CLI

- 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; it is necessary to remove a parameter.
- Wireshark allows you to specify one or more attachment points. To add more than one attachment point, re-enter the command with the new attachment point. To remove an attachment point, use the **no** form. You can specify an interface range as an attachment point.
- You cannot modify any parameters of a capture point while a session is active. To modify any parameter, stop the session, make the changes, and restart the session. Because an access list is generic to a switch and unrelated to the Wireshark process, it is alterable during a Wireshark session.
- 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 issue 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.
- 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.

Monitoring Wireshark

The commands in the following table are used to monitor Wireshark.

Table 56-2 Wireshark Monitoring Commands

Command	Purpose
show monitor capture point *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 file *name* [**display-filter** *filter-string*] [**brief** | **detailed** | **dump**]	Activates Wireshark using an existing .pcap file as the source for packets. If no display filter is specified, then all the packets in the file are displayed. The default display mode is **brief**.

Configuration Examples for Wireshark

Displaying a Brief Output from a .pcap File

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

```
Switch# show monitor capture file bootflash: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
```

Displaying Detailed Output from a .pcap File

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

```
Switch# show monitor capture file bootflash: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
```

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
```

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
```

Usage Examples for Wireshark

Example 1: Simple Capture and Display

Let us say we want to monitor traffic in the Layer 3 interface Gigabit 3/1:

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

```
Switch# monitor capture mycap interface gi 3/1 in match ipv4 any any
Switch# monitor capture mycap limit duration 60 packets 100
```

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

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

```
Switch# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet3/1 in
   monitor capture mycap match ipv4  any any
```

```
   monitor capture mycap limit packets 100 duration 60
Switch# show monitor capture mycap
Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet3/1, Direction: in
  Status : Inactive
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  File Details:
   File not associated
  Buffer Details:
   Buffer Type: LINEAR (default)
  Limit Details:
   Number of Packets to capture: 100
   Packet Capture duration: 60
```

Step 3 Start the capture process and display the results.

```
Switch# 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:

```
Switch# no monitor capture mycap
```

Example 2: 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.

```
Switch# monitor capture mycap interface gi 3/1 in match ipv4 any any
Switch# monitor capture mycap limit duration 60 packets 100
Switch# monitor cap mycap file location bootflash:mycap.pcap
```

Step 2 Confirm that the capture point has been correctly defined.

```
Switch# show monitor capture mycap parameter
   monitor capture mycap interface GigabitEthernet3/1 in
   monitor capture mycap match ipv4  any any
   monitor capture mycap file location bootflash:mycap.pcap
   monitor capture mycap limit packets 100 duration 60
Switch# show monitor capture mycap
  Target Type:
   Interface: GigabitEthernet3/1, Direction: in
  Status : Inactive
  Filter Details:
   IPv4
```

```
    Source IP:  any
    Destination IP:  any
   Protocol: any
  File Details:
   Associated file name: bootflash:mycap.pcap
  Buffer Details:
   Buffer Type: LINEAR (default)
  Limit Details:
   Number of Packets to capture: 100
   Packet Capture duration: 60
```

Step 3 Launch packet capture.

```
Switch# monitor capture mycap start
```

Step 4 After sufficient time has passed, stop the capture.

```
Switch# monitor capture mycap stop
```

Note Alternatively, you can 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.

```
Switch# show monitor capture file bootflash: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.

```
Switch# no monitor capture mycap
```

Example 3: Using Buffer Capture

This example shows how to use buffer capture:

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

```
Switch# monitor capture mycap interface gi 3/1 in
Switch# monitor capture mycap match ipv4 any any
Switch# monitor capture mycap buffer circular size 1
Switch# monitor capture mycap start
```

Step 2 Determine whether the capture is active.

```
Switch# show monitor capture mycap

Status Information for Capture mycap
  Target Type:
   Interface: GigabitEthernet3/1, Direction: in
```

```
  Status : Active
  Filter Details:
   IPv4
    Source IP:  any
    Destination IP:  any
   Protocol: any
  File Details:
   File not associated
  Buffer Details:
   Buffer Type: CIRCULAR
   Buffer Size (in MB): 1
  Limit Details:
   limit not set
```

Step 3 Display the packets in the buffer.

```
Switch# 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 now been buffered.

Step 4 Display the packets in other display modes.

```
Switch# 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)
…
Switch# 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 5 Clear the buffer once, wait for 10 seconds, then stop the traffic:

```
Switch# monitor capture mycap clear
```

Wait for 10 seconds and stop the traffic again.

Confirm that the same set of packets are displayed after this time gap.

```
Switch# 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

[Wait for about 10 secs]

Switch# 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

[Wait for about 10 secs]

Switch# 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 6 Clear the packets from the buffer.

```
Switch# monitor capture mycap clear
```

Step 7 Confirm that the buffer is now empty.

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

Wait about 10 seconds.

Step 8 Display the buffer contents.

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

Step 9 Restart the traffic, wait about 10 seconds, then display buffer contents.

```
Switch# 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
```

Step 10 Store the buffer contents to the mycap1.pcap file in the internal bootflash: storage device.

```
Switch# monitor capture mycap export bootflash:mycap1.pcap
Exported Successfully
```

Step 11 Ensure that the file has been created and that it contains the packets.

```
Switch# dir bootflash:mycap1.pcap
Directory of bootflash:/mycap1.pcap

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

831541248 bytes total (831340544 bytes free)
Switch# show monitor capture file bootflash: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 12 Stop the packet capture and display the buffer contents.

```
Switch# monitor capture mycap stop
Switch# 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 13 Clear the buffer and then try to display packets from the buffer.

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

Step 14 Delete the capture point.

```
Switch# no monitor capture mycap
```

Example 4: Capture Sessions

The following examples show how to start or stop a capture session in various modes:

```
Switch# monitor capture mycap int gi 3/1 in match ipv4 any any
Switch# monitor capture mycap file location bootflash:mycap.pcap
Switch# monitor capture mycap limit packets 100 duration 60

Switch# monitor capture mycap start
Switch#
Switch# monitor capture mycap stop
Switch# monitor capture mycap start capture-filter "udp.port == 20001"
Switch# monitor capture mycap stop
Switch# monitor capture mycap start capture-filter "udp.port == 20001" display
A file by the same capture file name already exists, overwrite?[confirm]

  0.000000     10.1.1.9 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.10 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
```

```
  0.000000    10.1.1.11 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.12 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.13 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.14 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.15 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.16 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.17 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.18 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.19 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.20 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.21 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.22 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.23 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.24 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.25 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.26 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.27 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.28 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.29 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.30 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

Switch# monitor capture mycap start capture-filter "udp.port == 20001" display
display-filter "udp.port == 20002"
%Display-filter cannot be specified when capture is associated to a file. Ignoring
display  filter%
A file by the same capture file name already exists, overwrite?[confirm]

  0.000000    10.1.1.96 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.97 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.98 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000    10.1.1.99 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.100 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.101 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.102 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.103 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.104 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.105 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.106 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.107 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.108 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  0.000000   10.1.1.109 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

Switch# monitor capture mycap start capture-filter "udp.port == 20001" display
display-filter "udp.port == 20002" detailed
%Display-filter cannot be specified when capture is associated to a file. Ignoring
display  filter%
A file by the same capture file name already exists, overwrite?[confirm]

Frame 1: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits)
    Arrival Time: Dec 31, 1969 17:00:00.000000000 PDT
    Epoch Time: 0.000000000 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)

Switch# monitor capture mycap start capture-filter "udp.port == 20001" display dump
A file by the same capture file name already exists, overwrite?[confirm]

  0.000000     10.1.1.6 -> 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 f6 0a 01 01 06 14 01   ......@.Y.......
0020  01 02 4e 21 4e 22 00 da 6e b1 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 ac 69 6e fd   .............in.

  0.000000     10.1.1.7 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

Switch# monitor capture mycap start display display-filter "udp.port == 20002"
%Display-filter cannot be specified when capture is associated to a file. Ignoring
display  filter%
A file by the same capture file name already exists, overwrite?[confirm]

  0.000000    10.1.1.41 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000    10.1.1.42 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000    10.1.1.43 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000    10.1.1.44 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000    10.1.1.45 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000    10.1.1.46 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.998993    10.1.1.47 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.998993    10.1.1.48 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.998993    10.1.1.49 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.998993    10.1.1.50 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.998993    10.1.1.51 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 10.998993    10.1.1.52 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

Switch# monitor capture mycap start display display-filter "udp.port == 20002" dump
%Display-filter cannot be specified when capture is associated to a file. Ignoring
display  filter%
A file by the same capture file name already exists, overwrite?[confirm]

  0.000000   10.1.1.117 -> 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 87 0a 01 01 75 14 01   ......@.Y....u..
0020  01 02 4e 21 4e 22 00 da 6e 42 00 01 02 03 04 05   ..N!N"..nB......
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 41 0c b4 5d   ............A..]

  1.000000   10.1.1.118 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002


Switch# no monitor capture mycap file

Switch# monitor capture mycap start display display-filter "udp.port == 20002" dump

  0.000000   10.1.1.160 -> 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 5c 0a 01 01 a0 14 01   ......@.Y\......
0020  01 02 4e 21 4e 22 00 da 6e 17 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 9f 20 8a e5   ............. ..

  1.000000   10.1.1.161 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

Switch# monitor capture mycap start display display-filter "udp.port == 20002"

  0.000000   10.1.1.173 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  1.000000   10.1.1.174 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  2.000000   10.1.1.175 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  3.000000   10.1.1.176 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  4.000000   10.1.1.177 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  5.000000   10.1.1.178 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  6.000000   10.1.1.179 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  7.000000   10.1.1.180 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  8.000000   10.1.1.181 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
  9.000000   10.1.1.182 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 10.000000   10.1.1.183 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 11.000000   10.1.1.184 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002
 12.000000   10.1.1.185 -> 20.1.1.2     UDP Source port: 20001  Destination port: 20002

Switch# monitor capture mycap start display detailed

Frame 1: 256 bytes on wire (2048 bits), 256 bytes captured (2048 bits)
    Arrival Time: Apr 12, 2012 11:46:54.245974000 PDT
    Epoch Time: 1334256414.245974000 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)

Switch#
```