Trace Management

The following sections are included in this chapter:

Tracing Overview

Tracing is a function that logs internal events. Trace files containing trace messages are automatically created and saved to the tracelogs directory on the hard disk: file system on the router, which stores tracing files in bootflash.

The contents of trace files are useful for the following purposes:

- Troubleshooting—Helps to locate and solve an issue with a router. The trace files can be accessed in diagnostic mode even if other system issues are occurring simultaneously.
- Debugging—Helps to obtain a detailed view of system actions and operations.

How Tracing Works

Tracing logs the contents of internal events on a router. Trace files containing all the trace output pertaining to a module are periodically created and updated and stored in the tracelog directory. Trace files can be erased from this directory to recover space on the file system without impacting system performance. The files can be copied to other destinations using file transfer functions (such as FTP and TFTP) and opened using a plain text editor.

Note Tracing cannot be disabled on a router.

Use the following commands to view trace information and set tracing levels:

- **show logging process module**—Shows the most recent trace information for a specific module. This command can be used in privileged EXEC and diagnostic modes. When used in diagnostic mode, this command can gather trace log information during a Cisco IOS XE failure.
- **set platform software trace**—Sets a tracing level that determines the types of messages that are stored in the output. For more information on tracing levels, see Tracing Levels, on page 4.

Configuring Packet Tracer with UDF Offset

Perform the following steps to configure the Packet-Trace UDF with offset:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **udf** *udf name* **header** {**inner** | **outer**} {**13**|**14**} **offset** *offset-in-bytes* **length** *length-in-bytes*
4. **udf** *udf name* {**header** | **packet-start**} *offset-base offset length*
5. **ip access-list extended** {*acl-name* |*acl-num*}
6. **ip access-list extended { deny | permit } udf udf-name value mask**
7. **debug platform condition [ipv4 | ipv6] [interface** *interface***] [access-list** *access-list -name* | *ipv4-address / subnet-mask* | *ipv6-address / subnet-mask***] [ingress | egress |both]**
8. **debug platform condition start**
9. **debug platform packet-trace packet** *pkt-num* **[fia-trace | summary-only] [circular] [data-size** *data-size***]**
10. **debug platform packet-trace {punt | inject|copy | drop |packet | statistics}**
11. **debug platform condition stop**
12. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**udf** *udf name* **header** {**inner** | **outer**} {**13**|**14**} **offset** *offset-in-bytes* **length** *length-in-bytes* **Example:** `Router(config)# udf TEST_UDF_NAME_1 header  inner 13 64 1`	Configures individual UDF definitions. You can specify the name of the UDF, the networking header from which offset, and the length of data to be extracted. The **inner** or **outer** keywords indicate the start of the offset from the unencapsulated Layer 3 or Layer 4 headers, or if there is an encapsulated packet, they indicate the start of offset from the inner L3/L4.

	Command or Action	Purpose

	`Router(config)# udf TEST_UDF_NAME_2 header  inner l4 77 2`
	`Router(config)# udf TEST_UDF_NAME_3 header outer l3 65 1`
	`Router(config)# udf TEST_UDF_NAME_4 header outer l4 67 1`
		The **length** keyword specifies, in bytes, the length from the offset. The range is from 1 to 2.
		.

Step 4	**udf** *udf name* {**header** | **packet-start**} *offset-base offset length* **Example:** `Router(config)# udf TEST_UDF_NAME_5 packet-start 120 1`	• header—Specifies the offset base configuration. • packet-start—Specifies the offset base from packet-start. packet-start" can vary depending on if packet-trace is for an inbound packet or outbound packet. If the packet-trace is for an inbound packet then the packet-start will be layer2. For outbound, he packet-start will be layer3. • offset—Specifies the number of bytes offset from the offset base. To match the first byte from the offset base (Layer 3/Layer 4 header), configure the offset as 0. • length—Specifies the number of bytes from the offset. Only 1 or 2 bytes are supported. To match additional bytes, you must define multiple UDFs.
Step 5	**ip access-list extended** {*acl-name* |*acl-num*} **Example:** `Router(config)# ip access-list extended acl2`	Enables extended ACL configuration mode. The CLI enters the extended ACL configuration mode in which all subsequent commands apply to the current extended access list. Extended ACLs control traffic by the comparison of the source and destination addresses of the IP packets to the addresses configured in the ACL.
Step 6	**ip access-list extended { deny | permit } udf udf-name value mask** **Example:** `Router(config-acl)# permit ip any any udf TEST_UDF_NAME_5 0xD3 0xFF`	Configures the ACL to match on UDFs along with the current access control entries (ACEs) . The bytes defined in ACL is 0xD3. Masks are used with IP addresses in IP ACLs to specify what should be permitted and denied.
Step 7	**debug platform condition [ipv4 | ipv6] [interface** *interface***] [access-list** *access-list -name* | *ipv4-address* / *subnet-mask* | *ipv6-address* / *subnet-mask***] [ingress | egress |both]** **Example:** `Router# debug platform condition interface gi0/0/0 ipv4 access-list acl2 both`	Specifies the matching criteria for tracing packets. Provides the ability to filter by protocol, IP address and subnet mask, access control list (ACL), interface, and direction.

	Command or Action	Purpose
Step 8	**debug platform condition start** **Example:** `Router# debug platform condition start`	Enables the specified matching criteria and starts packet tracing.
Step 9	**debug platform packet-trace packet** *pkt-num* **[fia-trace | summary-only] [circular] [data-size** *data-size***]** **Example:** `Router# debug platform packet-trace packet 1024 fia-trace data-size 2048`	Collects summary data for a specified number of packets. Captures feature path data by default, and optionally performs FIA trace. *pkt-num*—Specifies the maximum number of packets maintained at a given time. **fia-trace**—Provides detailed level of data capture, including summary data, feature-specific data. Also displays each feature entry visited during packet processing. **summary-only**—Enables the capture of summary data with minimal details. **circular**—Saves the data of the most recently traced packets. *data-size*—Specifies the size of data buffers for storing feature and FIA trace data for each packet in bytes. When very heavy packet processing is performed on packets, users can increase the size of the data buffers if necessary. The default value is 2048.
Step 10	**debug platform packet-trace {punt | inject|copy | drop |packet | statistics}** **Example:** `Router# debug platform packet-trace punt`	Enables tracing of punted packets from data to control plane.
Step 11	**debug platform condition stop** **Example:** `Router# debug platform condition start`	Deactivates the condition and stops packet tracing.
Step 12	**exit** **Example:** `Router# exit`	Exits the privileged EXEC mode.

Tracing Levels

Tracing levels determine how much information should be stored about a module in the trace buffer or file.

The following table shows all the tracing levels that are available and provides descriptions of what types of messages are displayed with each tracing level.

Table 1: Tracing Levels and Descriptions

Tracing Level	Level Number	Description
Emergency	0	The message is regarding an issue that makes the system unusable.
Alert	1	The message is regarding an action that must be taken immediately.
Critical	2	The message is regarding a critical condition. This is the default setting for every module on the router.
Error	3	The message is regarding a system error.
Warning	4	The message is regarding a system warning.
Notice	5	The message is regarding a significant issue, but the router is still working normally.
Informational	6	The message is useful for informational purposes only.
Debug	7	The message provides debug-level output.
Verbose	8	All possible tracing messages are sent.
Noise	—	All possible trace messages pertaining to a module are logged. The noise level is always equal to the highest possible tracing level. Even if a future enhancement to tracing introduces a higher tracing level than verbose level, the noise level will become equal to the level of the newly introduced tracing level.

If a tracing level is set, messages are collected from both lower tracing levels and from its own level.

For example, setting the tracing level to 3 (error) means that the trace file will contain output messages for levels: 0 (emergencies), 1 (alerts), 2 (critical), and 3 (error).

If you set the trace level to 4 (warning), it results in output messages for levels: 0 (emergencies), 1 (alerts), 2 (critical), 3 (error), and 4 (warning).

The default tracing level for every module on the router is 5 (notice).

A tracing level is not set in a configuration mode, which results in tracing-level settings being returned to default values after the router reloads.

Caution Setting the tracing level of a module to debug level or higher can have a negative impact on the performance.

Caution Setting high tracing levels on a large number of modules can severely degrade performance. If a high tracing level is required in a specific context, it is almost always preferable to set the tracing level of a single module to a higher level rather than setting multiple modules to high levels.

Viewing a Tracing Level

By default, all the modules on a router are set to 5 (notice). This setting is maintained unless changed by a user.

To see the tracing level for a module on a router, enter the **show logging process** command in privileged EXEC mode or diagnostic mode.

The following example shows how the **show logging process** command is used to view the tracing levels of the forwarding manager processes on an active RP:

```
Router# showlogging process forwarding-manager rp active
Module Name                     Trace Level
-----------------------------------------------
acl                             Notice
binos                           Notice
binos/brand                     Notice
bipc                            Notice
bsignal                         Notice
btrace                          Notice
cce                             Notice
cdllib                          Notice
cef                             Notice
chasfs                          Notice
chasutil                        Notice
erspan                          Notice
ess                             Notice
ether-channel                   Notice
evlib                           Notice
evutil                          Notice
file_alloc                      Notice
fman_rp                         Notice
fpm                             Notice
fw                              Notice
icmp                            Notice
interfaces                      Notice
iosd                            Notice
ipc                             Notice
ipclog                          Notice
iphc                            Notice
IPsec                           Notice
mgmte-acl                       Notice
mlp                             Notice
```

```
mqipc                            Notice
nat                              Notice
nbar                             Notice
netflow                          Notice
om                               Notice
peer                             Notice
qos                              Notice
route-map                        Notice
sbc                              Notice
services                         Notice
sw_wdog                          Notice
tdl_acl_config_type              Notice
tdl_acl_db_type                  Notice
tdl_cdlcore_message              Notice
tdl_cef_config_common_type       Notice
tdl_cef_config_type              Notice
tdl_dpidb_config_type            Notice
tdl_fman_rp_comm_type            Notice
tdl_fman_rp_message              Notice
tdl_fw_config_type               Notice
tdl_hapi_tdl_type                Notice
tdl_icmp_type                    Notice
tdl_ip_options_type              Notice
tdl_ipc_ack_type                 Notice
tdl_IPsec_db_type                Notice
tdl_mcp_comm_type                Notice
tdl_mlp_config_type              Notice
tdl_mlp_db_type                  Notice
tdl_om_type                      Notice
tdl_ui_message                   Notice
tdl_ui_type                      Notice
tdl_urpf_config_type             Notice
tdllib                           Notice
trans_avl                        Notice
uihandler                        Notice
uipeer                           Notice
uistatus                         Notice
urpf                             Notice
vista                            Notice
wccp                             Notice
```

Setting a Tracing Level

To set a tracing level for a module on a router, or for all the modules within a process on a router, enter the **set platform software trace** command in the privileged EXEC mode or diagnostic mode.

The following example shows the tracing level for the ACL module in the Forwarding Manager of the ESP processor in slot 0 set to `info`:

```
set platform software trace forwarding-manager F0 acl info
```

Viewing the Content of the Trace Buffer

To view the trace messages in the trace buffer or file, enter the **show logging process** command in privileged EXEC or diagnostic mode. In the following example, the trace messages for the Host Manager process in Route Processor slot 0 are viewed using the **show logging process command**:

```
Router# show logging process host-manager R0
08/23 12:09:14.408 [uipeer]: (info): Looking for a ui_req msg
08/23 12:09:14.408 [uipeer]: (info): Start of request handling for con 0x100a61c8
08/23 12:09:14.399 [uipeer]: (info): Accepted connection for 14 as 0x100a61c8
08/23 12:09:14.399 [uipeer]: (info): Received new connection 0x100a61c8 on descriptor 14
08/23 12:09:14.398 [uipeer]: (info): Accepting command connection on listen fd 7
08/23 11:53:57.440 [uipeer]: (info): Going to send a status update to the shell manager in
 slot 0
08/23 11:53:47.417 [uipeer]: (info): Going to send a status update to the shell manager in
 slot 0
```

Example: Using Packet Trace

This example provides a scenario in which packet trace is used to troubleshoot packet drops for a NAT configuration on a Cisco ASR 1006 Router. This example shows how you can effectively utilize the level of detail provided by the Packet-Trace feature to gather information about an issue, isolate the issue, and then find a solution.

In this scenario, you can detect that there are issues, but are not sure where to start troubleshooting. You should, therefore, consider accessing the Packet-Trace summary for a number of incoming packets.

```
Router# debug platform condition ingress
Router# debug platform packet-trace packet 2048 summary-only
Router# debug platform condition start
Router# debug platform condition stop
Router# show platform packet-trace summary
Pkt   Input             Output            State  Reason
0     Gi0/0/0           Gi0/0/0           DROP   402 (NoStatsUpdate)
1     internal0/0/rp:0  internal0/0/rp:0  PUNT   21  (RP<->QFP keepalive)
2     internal0/0/recycle:0  Gi0/0/0      FWD
```

The output shows that packets are dropped due to NAT configuration on Gigabit Ethernet interface 0/0/0, which enables you to understand that an issue is occurring on a specific interface. Using this information, you can limit which packets to trace, reduce the number of packets for data capture, and increase the level of inspection.

```
Router# debug platform packet-trace packet 256
Router# debug platform packet-trace punt
Router# debug platform condition interface Gi0/0/0
Router# debug platform condition start
Router# debug platform condition stop
Router# show platform packet-trace summary
Router# show platform packet-trace 15
Packet: 15          CBUG ID: 238
Summary
  Input     : GigabitEthernet0/0/0
  Output    : internal0/0/rp:1
  State     : PUNT 55  (For-us control)
  Timestamp
    Start   : 1166288346725 ns (06/06/2016 09:09:42.202734 UTC)
    Stop    : 1166288383210 ns (06/06/2016 09:09:42.202770 UTC)
Path Trace
  Feature: IPV4
    Input       : GigabitEthernet0/0/0
    Output      : <unknown>
    Source      : 10.64.68.3
    Destination : 224.0.0.102
    Protocol    : 17 (UDP)
```

```
      SrcPort   : 1985
      DstPort   : 1985
IOSd Path Flow: Packet: 15    CBUG ID: 238
  Feature: INFRA
    Pkt Direction: IN
    Packet Rcvd From CPP
  Feature: IP
    Pkt Direction: IN
    Source      : 10.64.68.122
    Destination : 10.64.68.255
  Feature: IP
    Pkt Direction: IN
    Packet Enqueued in IP layer
    Source      : 10.64.68.122
    Destination : 10.64.68.255
    Interface   : GigabitEthernet0/0/0
  Feature: UDP
    Pkt Direction: IN
    src         : 10.64.68.122(1053)
    dst         : 10.64.68.255(1947)
    length      : 48

Router#show platform packet-trace packet 10
Packet: 10          CBUG ID: 10
Summary
  Input     : GigabitEthernet0/0/0
  Output    : internal0/0/rp:0
  State     : PUNT 55  (For-us control)
  Timestamp
    Start   : 274777907351 ns (01/10/2020 10:56:47.918494 UTC)
    Stop    : 274777922664 ns (01/10/2020 10:56:47.918509 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet0/0/0
    Output      : <unknown>
    Source      : 10.78.106.2
    Destination : 224.0.0.102
    Protocol    : 17 (UDP)
      SrcPort   : 1985
      DstPort   : 1985

IOSd Path Flow: Packet: 10    CBUG ID: 10
  Feature: INFRA
    Pkt Direction: IN
Packet Rcvd From DATAPLANE
 Feature: IP
    Pkt Direction: IN
    Packet Enqueued in IP layer
    Source      : 10.78.106.2
    Destination : 224.0.0.102
    Interface   : GigabitEthernet0/0/0

  Feature: UDP
    Pkt Direction: IN DROP
    Pkt : DROPPED
    UDP: Discarding silently
    src        : 881 10.78.106.2(1985)
    dst         : 224.0.0.102(1985)
    length      : 60

Router#show platform packet-trace packet  12
Packet: 12          CBUG ID: 767
Summary
  Input     : GigabitEthernet3
```

```
  Output    : internal0/0/rp:0
  State     : PUNT 11  (For-us data)
  Timestamp
    Start   : 16120990774814 ns (01/20/2020 12:38:02.816435 UTC)
    Stop    : 16120990801840 ns (01/20/2020 12:38:02.816462 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet3
    Output      : <unknown>
    Source      : 12.1.1.1
    Destination : 12.1.1.2
    Protocol    : 6 (TCP)
      SrcPort   : 46593
      DstPort   : 23
IOSd Path Flow: Packet: 12    CBUG ID: 767
  Feature: INFRA
    Pkt Direction: IN
    Packet Rcvd From DATAPLANE

  Feature: IP
    Pkt Direction: IN
    Packet Enqueued in IP layer
    Source      : 12.1.1.1
    Destination : 12.1.1.2
    Interface   : GigabitEthernet3

  Feature: IP
    Pkt Direction: IN
    FORWARDEDTo transport layer
    Source        : 12.1.1.1
    Destination   : 12.1.1.2
    Interface     : GigabitEthernet3

  Feature: TCP
    Pkt Direction: IN
    tcp0: I NoTCB 12.1.1.1:46593 12.1.1.2:23 seq 1925377975 OPTS 4 SYN  WIN 4128
```

```
Router# show platform packet-trace summary
Pkt   Input                     Output                    State  Reason
0     INJ.2                     Gi1                       FWD
1     Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
2     INJ.2                     Gi1                       FWD
3     Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
4     INJ.2                     Gi1                       FWD
5     INJ.2                     Gi1                       FWD
6     Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
7     Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
8     Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
9     Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
10    INJ.2                     Gi1                       FWD
11    INJ.2                     Gi1                       FWD
12    INJ.2                     Gi1                       FWD
13    Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
14    Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
15    Gi1                       internal0/0/rp:0          PUNT   11  (For-us data)
16    INJ.2                     Gi1                       FWD
```

The following example displays the packet trace data statistics.

```
Router#show platform packet-trace statistics
Packets Summary
  Matched  3
  Traced   3
Packets Received
  Ingress  0
```

```
  Inject   0
Packets Processed
  Forward  0
  Punt     3
    Count       Code  Cause
    3           56    RP injected for-us control
  Drop     0
  Consume  0

          PKT_DIR_IN
             Dropped       Consumed       Forwarded
INFRA           0             0             0
TCP             0             0             0
UDP             0             0             0
IP              0             0             0
IPV6            0             0             0
ARP             0             0             0

         PKT_DIR_OUT
             Dropped       Consumed       Forwarded
INFRA           0             0             0
TCP             0             0             0
UDP             0             0             0
IP              0             0             0
IPV6            0             0             0
ARP             0             0             0
```

The following example displays packets that are injected and punted to the forwarding processor from the control plane.

```
Router#debug platform condition ipv4 10.118.74.53/32 both
 Router#Router#debug platform condition start
Router#debug platform packet-trace packet 200
Packet count rounded up from 200 to 256

Router#show platform packet-tracer packet 0
show plat pack pa 0
Packet: 0           CBUG ID: 674
Summary
  Input     : GigabitEthernet1
  Output    : internal0/0/rp:0
  State     : PUNT 11  (For-us data)
  Timestamp
    Start   : 17756544435656 ns (06/29/2020 18:19:17.326313 UTC)
    Stop    : 17756544469451 ns (06/29/2020 18:19:17.326346 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : GigabitEthernet1
    Output      : <unknown>
    Source      : 10.118.74.53
    Destination : 198.51.100.38
    Protocol    : 17 (UDP)
      SrcPort   : 2640
      DstPort   : 500

IOSd Path Flow: Packet: 0    CBUG ID: 674
  Feature: INFRA
  Pkt Direction: IN
    Packet Rcvd From DATAPLANE

  Feature: IP
  Pkt Direction: IN
    Packet Enqueued in IP layer
    Source      : 10.118.74.53
```

```
    Destination : 198.51.100.38
    Interface   : GigabitEthernet1

  Feature: IP
  Pkt Direction: IN
  FORWARDED To transport layer
    Source        : 10.118.74.53
    Destination   : 198.51.100.38
    Interface     : GigabitEthernet1

  Feature: UDP
  Pkt Direction: IN
  DROPPED
 UDP: Checksum error: dropping
 Source      : 10.118.74.53(2640)
 Destination : 198.51.100.38(500)

Router#show platform packet-tracer packet 2
Packet: 2           CBUG ID: 2

IOSd Path Flow:
  Feature: TCP
  Pkt Direction: OUTtcp0: O SYNRCVD 198.51.100.38:22 198.51.100.55:52774 seq 3052140910
OPTS 4 ACK 2346709419 SYN  WIN 4128

  Feature: TCP
  Pkt Direction: OUT
  FORWARDED
 TCP: Connection is in SYNRCVD state
 ACK         : 2346709419
 SEQ         : 3052140910
 Source      : 198.51.100.38(22)
 Destination : 198.51.100.55(52774)


  Feature: IP
  Pkt Direction: OUTRoute out the generated packet.srcaddr: 198.51.100.38, dstaddr:
198.51.100.55

  Feature: IP
  Pkt Direction: OUTInject and forward successful srcaddr: 198.51.100.38, dstaddr:
198.51.100.55

  Feature: TCP
  Pkt Direction: OUTtcp0: O SYNRCVD 198.51.100.38:22 198.51.100.55:52774 seq 3052140910
OPTS 4 ACK 2346709419 SYN  WIN 4128
Summary
  Input     : INJ.2
  Output    : GigabitEthernet1
  State     : FWD
  Timestamp
    Start   : 490928006866 ns (06/29/2020 13:31:30.807879 UTC)
    Stop    : 490928038567 ns (06/29/2020 13:31:30.807911 UTC)
Path Trace
  Feature: IPV4(Input)
    Input       : internal0/0/rp:0
    Output      : <unknown>
    Source      : 172.18.124.38
    Destination : 172.18.124.55
    Protocol    : 6 (TCP)
      SrcPort   : 22
      DstPort   : 52774
  Feature: IPSec
    Result    : IPSEC_RESULT_DENY
```

```
    Action    : SEND_CLEAR
    SA Handle : 0
    Peer Addr : 55.124.18.172
    Local Addr: 38.124.18.172

Router#
```