



Packet Trace

First Published: August 03, 2016

The Packet-Trace feature provides a detailed understanding of how data packets are processed by the Cisco IOS XE platform, and thus helps customers to diagnose issues and troubleshoot them more efficiently. This module provides information about how to use the Packet-Trace feature.

- [Information About Packet Trace, on page 1](#)
- [Usage Guidelines for Configuring Packet Trace, on page 2](#)
- [Configuring Packet Trace, on page 2](#)
- [Configuring Packet Tracer with UDF Offset , on page 5](#)
- [Displaying Packet-Trace Information, on page 7](#)
- [Removing Packet-Trace Data, on page 8](#)
- [Configuration Examples for Packet Trace , on page 8](#)
- [Additional References, on page 16](#)
- [Feature Information for Packet Trace, on page 16](#)

Information About Packet Trace

The Packet-Trace feature provides three levels of inspection for packets: accounting, summary, and path data. Each level provides a detailed view of packet processing at the cost of some packet processing capability. However, Packet Trace limits inspection to packets that match the debug platform condition statements, and is a viable option even under heavy-traffic situations in customer environments.

The following table explains the three levels of inspection provided by packet trace.

Table 1: Packet-Trace Level

Packet-Trace Level	Description
Accounting	Packet-Trace accounting provides a count of packets that enter and leave the network processor. Packet-Trace accounting is a lightweight performance activity, and runs continuously until it is disabled.

Packet-Trace Level	Description
Summary	At the summary level of packet trace, data is collected for a finite number of packets. Packet-Trace summary tracks the input and output interfaces, the final packet state, and punt, drop, or inject packets, if any. Collecting summary data adds to additional performance compared to normal packet processing, and can help to isolate a troublesome interface.
Path data	<p>The packet-trace path data level provides the greatest level of detail in packet trace. Data is collected for a finite number of packets. Packet-Trace path data captures data, including a conditional debugging ID that is useful to correlate with feature debugs, a timestamp, and also feature-specific path-trace data.</p> <p>Path data also has two optional capabilities: packet copy and Feature Invocation Array (FIA) trace. The packet-copy option enables you to copy input and output packets at various layers of the packet (layer 2, layer 3 or layer 4). The FIA- trace option tracks every feature entry invoked during packet processing and helps you to know what is happening during packet processing.</p> <p>Note Collecting path data consumes more packet-processing resources, and the optional capabilities incrementally affect packet performance. Therefore, path-data level should be used in limited capacity or in situations where packet performance change is acceptable.</p>

Usage Guidelines for Configuring Packet Trace

Consider the following best practices while configuring the Packet-Trace feature:

- Use of ingress conditions when using the Packet-Trace feature is recommended for a more comprehensive view of packets.
- Packet-trace configuration requires data-plane memory. On systems where data-plane memory is constrained, carefully consider how you will select the packet-trace values. A close approximation of the amount of memory consumed by packet trace is provided by the following equation:

memory required = (statistics overhead) + number of packets * (summary size + data size + packet copy size).

When the Packet-Trace feature is enabled, a small, fixed amount of memory is allocated for statistics. Similarly, when per-packet data is captured, a small, fixed amount of memory is required for each packet for summary data. However, as shown by the equation, you can significantly influence the amount of memory consumed by the number of packets you select to trace, and whether you collect path data and copies of packets.

Configuring Packet Trace

Perform the following steps to configure the Packet-Trace feature.



Note The amount of memory consumed by the Packet-Trace feature is affected by the packet-trace configuration. You should carefully select the size of per-packet path data and copy buffers and the number of packets to be traced in order to avoid interrupting normal services. You can check the current data-plane DRAM memory consumption by using the **show platform hardware qfp active infrastructure exmem statistics** command.

SUMMARY STEPS

1. **enable**
2. **debug platform packet-trace packet** *pkt-num* [**fia-trace** | **summary-only**] [**circular**] [**data-size** *data-size*]
3. **debug platform packet-trace** {**punt** | **inject**|**copy**|**drop**|**packet**|**statistics**}
4. **debug platform condition** [**ipv4** | **ipv6**] [**interface** *interface*][**access-list** *access-list -name* | *ipv4-address / subnet-mask* | *ipv6-address / subnet-mask*] [**ingress** | **egress** |**both**]
5. **debug platform condition start**
6. **debug platform condition stop**
7. **show platform packet-trace** {**configuration** | **statistics** | **summary** | **packet** {**all** | *pkt-num*}}
8. **clear platform condition all**
9. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	<p>enable</p> <p>Example:</p> <pre>Router> enable</pre>	Enables the privileged EXEC mode. Enter your password if prompted.
Step 2	<p>debug platform packet-trace packet <i>pkt-num</i> [fia-trace summary-only] [circular] [data-size <i>data-size</i>]</p> <p>Example:</p> <pre>Router# debug platform packet-trace packets 2048 summary-only</pre>	<p>Collects summary data for a specified number of packets. Captures feature path data by default, and optionally performs FIA trace.</p> <p><i>pkt-num</i>—Specifies the maximum number of packets maintained at a given time.</p> <p>fia-trace—Provides detailed level of data capture, including summary data, feature-specific data. Also displays each feature entry visited during packet processing.</p> <p>summary-only—Enables the capture of summary data with minimal details.</p> <p>circular—Saves the data of the most recently traced packets.</p> <p><i>data-size</i>—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</p>

	Command or Action	Purpose
		can increase the size of the data buffers if necessary. The default value is 2048.
Step 3	debug platform packet-trace {punt inject copy drop packet statistics} Example: <pre>Router# debug platform packet-trace punt</pre>	Enables tracing of punted packets from data to control plane.
Step 4	debug platform condition [ipv4 ipv6] [interface interface][access-list access-list -name ipv4-address / subnet-mask ipv6-address / subnet-mask] [ingress egress both] Example: <pre>Router# debug platform condition interface g0/0/0 ingress</pre>	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.
Step 5	debug platform condition start Example: <pre>Router# debug platform condition start</pre>	Enables the specified matching criteria and starts packet tracing.
Step 6	debug platform condition stop Example: <pre>Router# debug platform condition start</pre>	Deactivates the condition and stops packet tracing.
Step 7	show platform packet-trace {configuration statistics summary packet {all pkt-num}} Example: <pre>Router# show platform packet-trace 14</pre>	Displays packet-trace data according to the specified option. See {start cross reference} Table 21-1 {end cross reference} for detailed information about the show command options.
Step 8	clear platform condition all Example: <pre>Router(config)# clear platform condition all</pre>	Removes the configurations provided by the debug platform condition and debug platform packet-trace commands.
Step 9	exit Example: <pre>Router# exit</pre>	Exits the privileged EXEC mode.

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

Procedure

	Command or Action	Purpose
Step 1	enable Example: <pre>Device> enable</pre>	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: <pre>Device# configure terminal</pre>	Enters global configuration mode.
Step 3	udf <i>udf name</i> header { inner outer } { 13 14 } offset <i>offset-in-bytes</i> length <i>length-in-bytes</i> Example: <pre>Router(config)# udf TEST_UDF_NAME_1 header inner 13 64 1 Router(config)# udf TEST_UDF_NAME_2 header inner 14 77 2</pre>	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. The length keyword specifies, in bytes, the length from the offset. The range is from 1 to 2.

	Command or Action	Purpose
	<pre>Router(config)# udf TEST_UDF_NAME_3 header outer 13 65 1 Router(config)# udf TEST_UDF_NAME_4 header outer 14 67 1</pre>	
Step 4	<p>udf <i>udf name</i> {header packet-start} <i>offset-base offset length</i></p> <p>Example:</p> <pre>Router(config)# udf TEST_UDF_NAME_5 packet-start 120 1</pre>	<ul style="list-style-type: none"> • 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	<p>ip access-list extended {<i>acl-name</i> [<i>acl-num</i>]}</p> <p>Example:</p> <pre>Router(config)# ip access-list extended acl2</pre>	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	<p>ip access-list extended { deny permit } udf udf-name value mask</p> <p>Example:</p> <pre>Router(config-acl)# permit ip any any udf TEST_UDF_NAME_5 0xD3 0xFF</pre>	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	<p>debug platform condition [ipv4 ipv6] [interface interface] [access-list access-list -name <i>ipv4-address / subnet-mask</i> <i>ipv6-address / subnet-mask</i>] [ingress egress both]</p> <p>Example:</p> <pre>Router# debug platform condition interface gi0/0/0 ipv4 access-list acl2 both</pre>	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.
Step 8	<p>debug platform condition start</p> <p>Example:</p> <pre>Router# debug platform condition start</pre>	Enables the specified matching criteria and starts packet tracing.

	Command or Action	Purpose
Step 9	<p>debug platform packet-trace packet <i>pkt-num</i> [fia-trace summary-only] [circular] [data-size <i>data-size</i>]</p> <p>Example:</p> <pre>Router# debug platform packet-trace packet 1024 fia-trace data-size 2048</pre>	<p>Collects summary data for a specified number of packets. Captures feature path data by default, and optionally performs FIA trace.</p> <p><i>pkt-num</i>—Specifies the maximum number of packets maintained at a given time.</p> <p>fia-trace—Provides detailed level of data capture, including summary data, feature-specific data. Also displays each feature entry visited during packet processing.</p> <p>summary-only—Enables the capture of summary data with minimal details.</p> <p>circular—Saves the data of the most recently traced packets.</p> <p><i>data-size</i>—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.</p>
Step 10	<p>debug platform packet-trace {punt inject copy drop packet statistics}</p> <p>Example:</p> <pre>Router# debug platform packet-trace punt</pre>	Enables tracing of punted packets from data to control plane.
Step 11	<p>debug platform condition stop</p> <p>Example:</p> <pre>Router# debug platform condition start</pre>	Deactivates the condition and stops packet tracing.
Step 12	<p>exit</p> <p>Example:</p> <pre>Router# exit</pre>	Exits the privileged EXEC mode.

Displaying Packet-Trace Information

Use these **show** commands to display packet-trace information.

Table 2: show Commands

Command	Description
show platform packet-trace configuration	Displays packet trace configuration, including any defaults.

Command	Description
show platform packet-trace statistics	Displays accounting data for all the traced packets.
show platform packet-trace summary	Displays summary data for the number of packets specified.
show platform packet-trace {all <i>pkt-num</i>} [decode]	Displays the path data for all the packets or the packet specified. The decode option attempts to decode the binary packet into a more human- readable form.

Removing Packet-Trace Data

Use these commands to clear packet-trace data.

Table 3: clear Commands

Command	Description
clear platform packet-trace statistics	Clears the collected packet-trace data and statistics.
clear platform packet-trace configuration	Clears the packet-trace configuration and the statistics.

Configuration Examples for Packet Trace

This section provides the following configuration examples:

Example: Configuring Packet Trace

This example describes how to configure packet trace and display the results. In this example, incoming packets to Gigabit Ethernet interface 0/0/1 are traced, and FIA-trace data is captured for the first 128 packets. Also, the input packets are copied. The **show platform packet-trace packet 0** command displays the summary data and each feature entry visited during packet processing for packet 0.

```

Router>
enable
Router# debug platform packet-trace packet 128 fia-trace
Router# debug platform packet-trace punt
Router# debug platform condition interface g0/0/1 ingress
Router# debug platform condition start
Router#! ping to UUT
Router# debug platform condition stop
Router# show platform packet-trace packet 0
Packet: 0          CBUG ID: 9
Summary
  Input       : GigabitEthernet0/0/1
  Output      : GigabitEthernet0/0/0
  State       : FWD
  Timestamp
    Start     : 1819281992118 ns (05/17/2014 06:42:01.207240 UTC)
    Stop      : 1819282095121 ns (05/17/2014 06:42:01.207343 UTC)
  Path Trace

```



```

Feature: IPV4
  Source      : 198.51.100.2
  Destination : 198.51.100.2
  Protocol    : 1 (ICMP)
Feature: FIA_TRACE
  Entry       : 0x8059dbe8 - DEBUG_COND_INPUT_PKT
  Timestamp   : 3685243309297
Feature: FIA_TRACE
  Entry       : 0x82011a00 - IPV4_INPUT_DST_LOOKUP_CONSUME
  Timestamp   : 3685243311450
Feature: FIA_TRACE
  Entry       : 0x82000170 - IPV4_INPUT_FOR_US_MARTIAN
  Timestamp   : 3685243312427
Feature: FIA_TRACE
  Entry       : 0x82004b68 - IPV4_OUTPUT_LOOKUP_PROCESS
  Timestamp   : 3685243313230
Feature: FIA_TRACE
  Entry       : 0x8034f210 - IPV4_INPUT_IPOPTIONS_PROCESS
  Timestamp   : 3685243315033
Feature: FIA_TRACE
  Entry       : 0x82013200 - IPV4_OUTPUT_GOTO_OUTPUT_FEATURE
  Timestamp   : 3685243315787
Feature: FIA_TRACE
  Entry       : 0x80321450 - IPV4_VFR_REFRAG
  Timestamp   : 3685243316980
Feature: FIA_TRACE
  Entry       : 0x82014700 - IPV6_INPUT_L2_REWRITE
  Timestamp   : 3685243317713
Feature: FIA_TRACE
  Entry       : 0x82000080 - IPV4_OUTPUT_FRAG
  Timestamp   : 3685243319223
Feature: FIA_TRACE
  Entry       : 0x8200e500 - IPV4_OUTPUT_DROP_POLICY
  Timestamp   : 3685243319950
Feature: FIA_TRACE
  Entry       : 0x8059aff4 - PACTRAC_OUTPUT_STATS
  Timestamp   : 3685243323603
Feature: FIA_TRACE
  Entry       : 0x82016100 - MARMOT_SPA_D_TRANSMIT_PKT
  Timestamp   : 3685243326183

```

```

Router# clear platform condition all
Router# exit

```

Linux Forwarding Transport Service (LFTS) is a transport mechanism to forward packets punted from the CPP into applications other than IOSd. This example displays the LFTS-based intercepted packet destined for binos application.

```

Router# show platform packet-trace packet 10
Packet: 10      CBUG ID: 52
Summary
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  State  : PUNT 55 (For-us control)
  Timestamp
    Start : 597718358383 ns (06/06/2016 09:00:13.643341 UTC)
    Stop  : 597718409650 ns (06/06/2016 09:00:13.643392 UTC)
Path Trace
  Feature: IPV4
  Input  : GigabitEthernet0/0/0
  Output : <unknown>
  Source : 10.64.68.2
  Destination : 224.0.0.102
  Protocol : 17 (UDP)

```

```

    SrcPort : 1985
    DstPort : 1985
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : <unknown>
  Entry  : 0x8a0177bc - DEBUG_COND_INPUT_PKT
  Lapsed time : 426 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : <unknown>
  Entry  : 0x8a017788 - IPV4_INPUT_DST_LOOKUP_CONSUME
  Lapsed time : 386 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : <unknown>
  Entry  : 0x8a01778c - IPV4_INPUT_FOR_US_MARTIAN
  Lapsed time : 13653 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017730 - IPV4_INPUT_LOOKUP_PROCESS_EXT
  Lapsed time : 2360 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017be0 - IPV4_INPUT_IPOPTIONS_PROCESS_EXT
  Lapsed time : 66 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017bfc - IPV4_INPUT_GOTO_OUTPUT_FEATURE_EXT
  Lapsed time : 680 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017d60 - IPV4_INTERNAL_ARL_SANITY_EXT
  Lapsed time : 320 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017a40 - IPV4_VFR_REFRAG_EXT
  Lapsed time : 106 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017d2c - IPV4_OUTPUT_DROP_POLICY_EXT
  Lapsed time : 1173 ns
Feature: FIA_TRACE
  Input  : GigabitEthernet0/0/0
  Output : internal0/0/rp:1
  Entry  : 0x8a017940 - INTERNAL_TRANSMIT_PKT_EXT
  Lapsed time : 20173 ns
LFTS Path Flow: Packet: 10    CBUG ID: 52
Feature: LFTS
  Pkt Direction: IN
  Punt Cause   : 55
  subCause    : 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
```

Example: Using Packet Trace

```

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#
```

Additional References

Standards

Standard	Title
None	—

MIBs

MIB	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at this URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
None	—

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Packet Trace

Table 21-4 lists the features in this module and provides links to specific configuration information.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature

set, or platform. To access Cisco Feature Navigator, go to <http://www.cisco.com/go/cfn>. An account on Cisco.com is not required.



Note [Table 21-4](#) lists only the software releases that support a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 4: Feature Information for Packet Trace

Feature Name	Releases	Feature Information
Packet Trace	Cisco IOS XE 3.10S	<p>The Packet Trace feature provides information about how data packets are processed by the Cisco IOS XE software.</p> <p>In Cisco IOS XE Release 3.10S, this feature was introduced on the Cisco ASR 1000 Series Aggregation Services Routers.</p> <p>The following commands were introduced or modified:</p> <ul style="list-style-type: none"> • debug platform packet-trace packet <i>pkt-num</i> [fia-trace summary-only] [data-size <i>data-size</i>] [circular] • debug platform packet-trace copy packet {input output both} [size <i>num-bytes</i>] [L2 L3 L4] • show platform packet-trace {configuration statistics summary packet {all <i>pkt-num</i>}}
	Cisco IOS XE 3.11S	<p>In Cisco IOS XE Release 3.11S, this feature was enhanced to include the following features:</p> <ul style="list-style-type: none"> • Matched versus traced statistics. • Trace stop timestamp in addition to trace start timestamp. <p>The following commands were introduced or modified:</p> <ul style="list-style-type: none"> • debug platform packet-trace drop [code <i>drop-num</i>] • show platform packet-trace packet {all <i>pkt-num</i>} [decode]
	Cisco IOS XE Denali 16.3.1	<p>In Cisco IOS XE Denali 16.3.1, this feature was enhanced to include Layer3 packet tracing along with IOSd.</p> <p>The following commands were introduced or modified: debug platform packet-trace punt.</p>
	Cisco IOS XE Amsterdam 17.3.1	<p>The output of the show platform packet-trace command now includes additional trace information for packets either originated from IOSd or destined to IOSd or other BinOS processes.</p>

