debug ip rtp header-compression through debug ipv6 icmp

• debug ip rtp header-compression through debug ipv6 icmp, on page 2

debug ip rtp header-compression through debug ipv6 icmp

debug ip rtp header-compression

To display events specific to Real-Time Transport Protocol (RTP) header compression, use the **debug ip rtp header-compression**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip rtp header-compression
no debug ip rtp header-compression

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug ip rtp header-compression**command:

```
Router# debug ip rtp header-compression
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 0, received sequence 0
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 1, received sequence 1
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 2, received sequence 2
RHC BRI0: rcv compressed rtp packet
RHC BRI0: context0: expected sequence 3, received sequence 3
```

The table below describes the significant fields shown in the display.

Table 1: debug ip rtp header-compression Field Descriptions

Field	Description
context0	Compression state for a connection 0.
expected sequence	RTP header compression link sequence (expected).
received sequence	RTP header compression link sequence (actually received).

Related Commands

Command	Description
debug ip rtp packets	Displays a detailed dump of packets specific to RTP header compression.

debug ip rtp packets

To display a detailed dump of packets specific to Real-Time Transport Protocol (RTP) header compression, use the **debug ip rtp packets**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip rtp packets
no debug ip rtp packets

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug ip rtp packets**command:

```
Router# debug ip rtp packets
RTP packet dump:
  IP:  source: 171.68.8.10, destination: 224.2.197.169, id: 0x249B, ttl: 9,
       TOS: 0 prot: 17,
  UDP: source port: 1034, destination port: 27404, checksum: 0xB429,len: 152
  RTP: version: 2, padding: 0, extension: 0, marker: 0,
       payload: 3, ssrc 2369713968,
       sequence: 2468, timestamp: 85187180, csrc count: 0
```

The table below describes the significant fields shown in the display.

Table 2: debug ip rtp packets Field Descriptions

Field	Description
id	IP identification.
ttl	IP time to live (TTL).
len	Total UDP length.

Related Commands

Command	Description
debug ip rtp header-compression	Displays events specific to RTP header compression.

debug ip scp

To troubleshoot secure copy (SCP) authentication problems, use the **debug ip scp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip scp
no debug ip scp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.

Release	Modification
12.0(21)S	This command was integrated into Cisco IOS Release 12.0(21)S.
12.2(22)S	This command was integrated into Cisco IOS Release 12.2(22)S.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(18)SXD	This command was integrated into Cisco IOS Release 12.2(18)SXD.

Examples

The following example is output from the **debug ip scp** command. In this example, a copy of the file scptest.cfg from a UNIX host running configuration of the router was successful.

```
Router# debug ip scp
4d06h:SCP:[22 -> 10.11.29.252:1018] send <OK>
4d06h:SCP:[22 <- 10.11.29.252:1018] recv C0644 20 scptest.cfg
4d06h:SCP:[22 -> 10.11.29.252:1018] send <OK>
4d06h:SCP:[22 <- 10.11.29.252:1018] recv 20 bytes
4d06h:SCP:[22 <- 10.11.29.252:1018] recv <OK>
4d06h:SCP:[22 -> 10.11.29.252:1018] send <OK>
4d06h:SCP:[22 <- 10.11.29.252:1018] recv <EOF>
```

The following example is also output from the **debug ip scp** command, but in this example, the user has privilege 0 and is therefore denied:

```
Router# debug ip scp
4d06h:SCP:[22 -> 10.11.29.252:1018] send Privilege denied.
```

Related Commands

Command	Description
ip scp server enable	Enables SCP server-side functionality.

debug ip sctp api

To provide diagnostic information about Stream Control Transmission Protocol (SCTP) application programming interfaces (APIs), use the **debug ip sctp api**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp api
no debug ip sctp api

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

In a live system, the debugging messages for performance, state, signal, and warnings are the most useful. These show any association or destination address failures and can be used to monitor the stability of any established associations.

Caution The **debug ip sctp api** command should not be used in a live system that has any significant amount of traffic running because it can generate a lot of traffic, which can cause associations to fail.

Examples

The following example shows SCTP calls to the API that are being executed and the parameters associated with these calls:

```
Router# debug ip sctp api
*Mar  1 00:31:14.211: SCTP: sctp_send: Assoc ID: 1
*Mar  1 00:31:14.211: SCTP:            stream num: 10
*Mar  1 00:31:14.211: SCTP:            bptr: 62EE332C, dptr: 4F7B598
*Mar  1 00:31:14.211: SCTP:            datalen: 100
*Mar  1 00:31:14.211: SCTP:            context: 1
*Mar  1 00:31:14.211: SCTP:            lifetime: 0
*Mar  1 00:31:14.211: SCTP:            unorder flag: FALSE
*Mar  1 00:31:14.211: SCTP:            bundle flag: TRUE
*Mar  1 00:31:14.211: SCTP: sctp_send successful return
*Mar  1 00:31:14.211: SCTP: sctp_receive: Assoc ID: 1
*Mar  1 00:31:14.215: SCTP:               max data len: 100
*Mar  1 00:31:14.215: SCTP: sctp_receive successful return
*Mar  1 00:31:14.215: SCTP: Process Send Request
*Mar  1 00:31:14.951: SCTP: sctp_receive: Assoc ID: 0
*Mar  1 00:31:14.951: SCTP:               max data len: 100
*Mar  1 00:31:14.951: SCTP: sctp_receive successful return
.
.
.
```

The table below describes the significant fields shown in the display.

Table 3: debug ip sctp api Field Descriptions

Field	Description
Assoc ID	Association identifier.
stream num	SCTP stream number.
bptr, dptr	Address of the buffer that contains the data, and address of the start of the data.
datalen	Length of the data that the application is sending (the datagram).
context	A value that is meaningful to the application. Returned with the datagram if the datagram ever needs to be retrieved.
lifetime	Not used.
unorder flag	Specifies that the datagram should be sent as unordered data.
bundle flag	Indicates whether the application wants the datagram to be delayed slightly, trying to bundle it with other data being sent.

Field	Description
max data len	Maximum length of data that can be received--the size of the receive buffer.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp congestion

To provide diagnostic information about Stream Control Transmission Protocol (SCTP) congestion parameters, use the **debug ip sctp congestion**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp congestion
no debug ip sctp congestion

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

In a live system, the debugging messages for performance, state, signal, and warnings are the most useful. These show any association or destination address failures and can be used to monitor the stability of any established associations.

Debug commands other than those for performance, state, signal, and warnings can generate a great deal of output and therefore can cause associations to fail. These commands should be used only in test environments or when there are very low amounts of traffic.

Examples

The following example shows parameters used to calculate SCTP congestion:

```
Router# debug ip sctp congestion
SCTP: Assoc 0: Slow start 10.6.0.4, cwnd 3000
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7800
SCTP: Assoc 0: Free chunks, local rwnd 9000
SCTP: Assoc 0: Data chunks rcvd, local rwnd 8200
SCTP: Assoc 0: Add Sack, local a_rwnd 8200
SCTP: Assoc 0: Free chunks, local rwnd 9000
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7800
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7000
SCTP: Assoc 0: Add Sack, local a_rwnd 7000
SCTP: Assoc 0: Free chunks, local rwnd 9000
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 14000, cwnd 19500, outstand 0
SCTP: Assoc 0: Bundled 12 chunks, remote rwnd 12800, outstand 1200
SCTP: Assoc 0: Bundling data, next chunk dataLen (100) > remaining mtu size
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 12800, cwnd 19500, outstand 1200
SCTP: Assoc 0: Bundled 12 chunks, remote rwnd 11600, outstand 2400
SCTP: Assoc 0: Bundling data, next chunk dataLen (100) > remaining mtu size
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 11600, cwnd 19500, outstand 2400
SCTP: Assoc 0: Bundled 12 chunks, remote rwnd 10400, outstand 3600
SCTP: Assoc 0: Bundling data, next chunk dataLen (100) > remaining mtu size
SCTP: Assoc 0: Bundle for 10.5.0.4, rem rwnd 10400, cwnd 19500, outstand 3600
SCTP: Assoc 0: Bundled 4 chunks, remote rwnd 10000, outstand 4000
SCTP: Assoc 0: No additional chunks waiting.
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7800
SCTP: Assoc 0: Data chunks rcvd, local rwnd 7000
SCTP: Assoc 0: Add Sack, local a_rwnd 7000
SCTP: Assoc 0: Chunk A22F3B45 ack'd, dest 10.5.0.4, outstanding 3900
SCTP: Assoc 0: Chunk A22F3B46 ack'd, dest 10.5.0.4, outstanding 3800
SCTP: Assoc 0: Chunk A22F3B47 ack'd, dest 10.5.0.4, outstanding 3700
SCTP: Assoc 0: Chunk A22F3B48 ack'd, dest 10.5.0.4, outstanding 3600
SCTP: Assoc 0: Chunk A22F3B49 ack'd, dest 10.5.0.4, outstanding 3500
SCTP: Assoc 0: Chunk A22F3B4A ack'd, dest 10.5.0.4, outstanding 3400
SCTP: Assoc 0: Chunk A22F3B4B ack'd, dest 10.5.0.4, outstanding 3300
SCTP: Assoc 0: Chunk A22F3B4C ack'd, dest 10.5.0.4, outstanding 3200
SCTP: Assoc 0: Chunk A22F3B4D ack'd, dest 10.5.0.4, outstanding 3100
SCTP: Assoc 0: Chunk A22F3B4E ack'd, dest 10.5.0.4, outstanding 3000
SCTP: Assoc 0: Chunk A22F3B4F ack'd, dest 10.5.0.4, outstanding 2900
SCTP: Assoc 0: Chunk A22F3B50 ack'd, dest 10.5.0.4, outstanding 2800
SCTP: Assoc 0: Chunk A22F3B51 ack'd, dest 10.5.0.4, outstanding 2700
SCTP: Assoc 0: Chunk A22F3B52 ack'd, dest 10.5.0.4, outstanding 2600
SCTP: Assoc 0: Chunk A22F3B53 ack'd, dest 10.5.0.4, outstanding 2500
SCTP: Assoc 0: Chunk A22F3B54 ack'd, dest 10.5.0.4, outstanding 2400
SCTP: Assoc 0: Chunk A22F3B55 ack'd, dest 10.5.0.4, outstanding 2300
SCTP: Assoc 0: Chunk A22F3B56 ack'd, dest 10.5.0.4, outstanding 2200
```

The table below describes the significant fields shown in the display.

Table 4: debug ip sctp congestion Field Descriptions

Field	Description
cwnd	Congestion window values for destination address.
rwnd, a_rwnd	Receiver window values as defined in RFC 2960.
outstanding	Number of bytes outstanding.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp init

To show datagrams and other information related to the initializing of new Stream Control Transmission Protocol (SCTP) associations, use the **debug ip sctp init**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp init
no debug ip sctp init

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(4)T	This command was introduced.

Release	Modification
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

All initialization chunks are shown, including the INIT, INIT_ACK, COOKIE_ECHO, and COOKIE_ACK chunks. This debug command can be used to see the chunks associated with any initialization sequence but does not display data chunks sent once the association is established. Therefore, it is safe to use in a live system that has traffic flowing when you have trouble with associations failing and being reestablished.

Examples

The following example shows initialization chunks for SCTP associations:

```
Router# debug ip sctp init
*Mar  1 00:53:07.279: SCTP Test: Attempting to open assoc to remote port 8787...assoc ID
is 0
*Mar  1 00:53:07.279: SCTP: Process Assoc Request
*Mar  1 00:53:07.279: SCTP: Assoc 0: dest addr list:
*Mar  1 00:53:07.279: SCTP:                 addr 10.5.0.4
*Mar  1 00:53:07.279: SCTP:                 addr 10.6.0.4
*Mar  1 00:53:07.279:
...
*Mar  1 00:53:13.279: SCTP: Assoc 0: Send Init
*Mar  1 00:53:13.279: SCTP:        INIT_CHUNK, len 42
*Mar  1 00:53:13.279: SCTP:        Initiate Tag: B4A10C4D, Initial TSN: B4A10C4D, rwnd 9000
*Mar  1 00:53:13.279: SCTP:        Streams Inbound: 13, Outbound: 13
*Mar  1 00:53:13.279: SCTP:        IP Addr: 10.1.0.2
*Mar  1 00:53:13.279: SCTP:        IP Addr: 10.2.0.2
*Mar  1 00:53:13.279: SCTP:        Supported addr types: 5
*Mar  1 00:53:13.307: SCTP: Process Init
*Mar  1 00:53:13.307: SCTP:        INIT_CHUNK, len 42
*Mar  1 00:53:13.307: SCTP:        Initiate Tag: 3C2D8327, Initial TSN: 3C2D8327, rwnd 18000
*Mar  1 00:53:13.307: SCTP:        Streams Inbound: 13, Outbound: 13
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.5.0.4
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.6.0.4
*Mar  1 00:53:13.307: SCTP:        Supported addr types: 5
*Mar  1 00:53:13.307: SCTP: Assoc 0: Send InitAck
*Mar  1 00:53:13.307: SCTP:        INIT_ACK_CHUNK, len 124
*Mar  1 00:53:13.307: SCTP:        Initiate Tag: B4A10C4D, Initial TSN: B4A10C4D, rwnd 9000
*Mar  1 00:53:13.307: SCTP:        Streams Inbound: 13, Outbound: 13
*Mar  1 00:53:13.307: SCTP:        Responder cookie len 88
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.1.0.2
*Mar  1 00:53:13.307: SCTP:        IP Addr: 10.2.0.2
*Mar  1 00:53:13.311: SCTP: Assoc 0: Process Cookie
*Mar  1 00:53:13.311: SCTP:        COOKIE_ECHO_CHUNK, len 88
*Mar  1 00:53:13.311: SCTP: Assoc 0: dest addr list:
*Mar  1 00:53:13.311: SCTP:                 addr 10.5.0.4
*Mar  1 00:53:13.311: SCTP:                 addr 10.6.0.4
*Mar  1 00:53:13.311:
*Mar  1 00:53:13.311: SCTP: Instance 0 dest addr list:
*Mar  1 00:53:13.311: SCTP:                 addr 10.5.0.4
*Mar  1 00:53:13.311: SCTP:                 addr 10.6.0.4
*Mar  1 00:53:13.311:
*Mar  1 00:53:13.311: SCTP: Assoc 0: Send CookieAck
*Mar  1 00:53:13.311: SCTP:        COOKIE_ACK_CHUNK
```

The table below describes the significant fields shown in the display.

Table 5: debug ip sctp init Field Descriptions

Field	Description
Initiate Tag	Initiation chunk identifier.
Initial TSN	Initial transmission sequence number.
rwnd	Receiver window values.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp multihome

To show the source and destination of datagrams in order to monitor the use of the multihome addresses for Stream Control Transmission Protocol (SCTP), use the **debug ip sctp multihome**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp multihome
no debug ip sctp multihome

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

More than one IP address parameter can be included in an initialization (INIT) chunk when the INIT sender is multihomed. Datagrams should be sent to the primary destination addresses unless the network is experiencing problems, in which case the datagrams should be sent to secondary addresses.

Caution The **debug ip sctp multihome** command generates one debug line for each datagram sent or received. It should be used with extreme caution in a live network.

Examples

The following example shows source and destination for multihomed addresses:

```
Router# debug ip sctp multihome
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 476
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Assoc 0: Send Data to dest 10.5.0.4
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 476
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 476
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Assoc 0: Send Data to dest 10.5.0.4
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 476
SCTP: Rcvd s=10.6.0.4  8787, d=10.2.0.2  8787, len 44
SCTP: Sent:  Assoc 0: s=10.2.0.2  8787, d=10.6.0.4  8787, len 44
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 1404
SCTP: Rcvd s=10.5.0.4  8787, d=10.1.0.2  8787, len 476
```

The table below describes the significant fields shown in the display.

Table 6: debug ip sctp multihome Field Descriptions

Field	Description
s	Source address and port.
d	Destination address and port.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp performance

To display the average number of Stream Control Transmission Protocol (SCTP) chunks and datagrams being sent and received per second, use the **debug ip sctp performance**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp performance
no debug ip sctp performance

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

In a live system, the debugging messages for performance, state, signal, and warnings are the most useful. These show any association or destination address failures and can be used to monitor the stability of any established associations.

Once enabled, the **debug ip sctp performance** command displays the average number of chunks and datagrams being sent and received per second once every 10 seconds. Note that the averages are cumulative since the last time the statistics were cleared using the **clear ip sctp statistics** command and may not accurately reflect the number of datagrams and chunks currently being sent and received at that particular moment.

Examples

The following example shows a low rate of traffic:

```
Router# debug ip sctp performance

SCTP Sent: SCTP Dgrams 5, Chunks 28, Data Chunks 29, ULP Dgrams 29
SCTP Rcvd: SCTP Dgrams 7, Chunks 28, Data Chunks 29, ULP Dgrams 29
Chunks Discarded: 0,  Retransmitted 0
SCTP Sent: SCTP Dgrams 6, Chunks 29, Data Chunks 30, ULP Dgrams 30
SCTP Rcvd: SCTP Dgrams 7, Chunks 29, Data Chunks 30, ULP Dgrams 30
Chunks Discarded: 0,  Retransmitted 0
```

The table below describes the significant fields shown in the display.

Table 7: debug ip sctp performance Field Descriptions

Field	Description
SCTP Dgrams	Datagram sent to or received from the network.
Chunks	Includes data chunks and control chunks sent or received.
Data Chunks	Data chunks sent or received.
ULP Dgrams	Upper-layer protocol (ULP) datagrams, which are datagrams sent to or received from the ULP or application.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp rcvchunks

To provide diagnostic information about chunks received with Stream Control Transmission Protocol (SCTP), use the **debug ip sctp rcvchunks**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp rcvchunks
no debug ip sctp rcvchunks

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp rcvchunks** command shows the following information about received chunks:

- Whether the chunk is for a new datagram or is part of a datagram that is being reassembled
- Whether the datagram is complete after receiving this chunk
- If the datagram is complete, whether the datagram is in sequence within the specified stream and can be delivered to the upper-layer protocol (ULP)
- The selective acknowledgments (SACKs) that are returned to the remote SCTP peer
- The cumulative transmission sequence number (Cum TSN) that was acknowledged and the number of fragments included
- Whether the datagram is received by the ULP

Caution The **debug ip sctp rcvchunks** command generates multiple debug lines for each chunk received. It should be used with extreme caution in a live network.

Examples

In the following example, a segmented datagram is received in two chunks for stream 0 and sequence number 0. The length of the first chunk is 1452 bytes, and the second is 1 byte. The first chunk indicates that it is for a new datagram, but the second chunk indicates that it is part of an existing datagram that is already being reassembled. When the first chunk is processed, it is noted to be in sequence, but is not complete and so cannot be delivered yet. When the second chunk is received, the datagram is both in sequence and complete. The application receives the datagram, and a SACK is shown to acknowledge that both chunks were received with no missing chunks indicated (that is, with no fragments).

```
Router# debug ip sctp rcvchunks
SCTP: Assoc 0: New chunk (0/0/1452/2C33D822) for new dgram (0)
```

```
SCTP: Assoc 0: dgram (0) is in seq
SCTP: Assoc 0: Add Sack Chunk, CumTSN=2C33D822, numFrags=0
SCTP: Assoc 0: New chunk (0/0/1/2C33D823) for existing dgram (0)
SCTP: Assoc 0: dgram (0) is complete
SCTP: Assoc 0: ApplRecv chunk 0/0/1452/2C33D822
SCTP: Assoc 0: ApplRecv chunk 0/0/1/2C33D823
SCTP: Assoc 0: Add Sack Chunk, CumTSN=2C33D823, numFrags=0
```

The table below describes the significant fields shown in the display.

Table 8: debug ip sctp rcvchunks Field Descriptions

Field	Description
0 / 0 / 1452 / 2C33D822	Stream number / datagram sequence number / chunk length, in bytes / chunk transmission sequence number.
Sack Chunk	Selective acknowledgment chunk.
ApplRecv	Application has received the chunk.
CumTSN	Cumulative transmission sequence number that is being acknowledged.
numFrags	Number of fragments, or missing chunks.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp rto

To show adjustments that are made to the retransmission timeout (RTO) value when using Stream Control Transmission Protocol (SCTP), use the **debug ip sctp rto**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp rto
no debug ip sctp rto

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp rto** command shows adjustments that are made to the retransmission timeout value (shown as retrans in the command output) because of either retransmission of data chunks or unacknowledged heartbeats.

Caution The **debug ip sctp rto** command can generate a great deal of output. It should be used with extreme caution in a live network.

Examples

In the following example, there is only one destination address available. Each time the chunk needs to be retransmitted, the RTO value is doubled.

```
Router# debug ip sctp rto
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 2000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 4000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 8000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 16000 ms
SCTP: Assoc 0: destaddr 10.5.0.4, retrans timeout on chunk 942BAC55
SCTP: Assoc 0: destaddr 10.5.0.4, rto backoff 32000 ms
```

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.

Command	Description
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp segments

To show short diagnostics for every datagram that is sent or received with Stream Control Transmission Protocol (SCTP), use the **debug ip sctp segments**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp segments
no debug ip sctp segments

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp segments** command provides the short form of the output about datagrams. For the verbose form, use the **debug ip sctp segmentv** command.

Caution The **debug ip sctp segments** command generates several lines of output for each datagram sent or received. It should be used with extreme caution in a live network.

Examples

The following output shows an example in which an association is established, a few heartbeats are sent, the remote endpoint fails, and the association is restarted.

```
Router# debug ip sctp segments
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 56
SCTP:        INIT_CHUNK, Tag: 3C72A02A, TSN: 3C72A02A
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 56
SCTP:        INIT_CHUNK, Tag: 13E5AD6C, TSN: 13E5AD6C
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 136
SCTP:        INIT_ACK_CHUNK, Tag: 3C72A02A, TSN: 3C72A02A
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 100
```

```
SCTP:        COOKIE_ECHO_CHUNK, len 88
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 16
SCTP:        COOKIE_ACK_CHUNK
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 52
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 52
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 52
SCTP:        HEARTBEAT_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 56
SCTP:        INIT_CHUNK, Tag: 4F2D8235, TSN: 4F2D8235
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 136
SCTP:        INIT_ACK_CHUNK, Tag: 7DD7E424, TSN: 7DD7E424
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 100
SCTP:        COOKIE_ECHO_CHUNK, len 88
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 16
SCTP:        COOKIE_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 144
SCTP:        SACK_CHUNK, TSN ack: 7DD7E423, rwnd 18000, num frags 0
SCTP:        DATA_CHUNK, 4/0/100/4F2D8235
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 4F2D8235, rwnd 8900, num frags 0
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 128
SCTP:        DATA_CHUNK, 4/0/100/7DD7E424
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 7DD7E424, rwnd 17900, num frags 0
SCTP: Recv:  Assoc 0: s=10.6.0.4  8787, d=10.2.0.2  8787, len 44
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.2.0.2  8787, d=10.6.0.4  8787, len 44
SCTP:        HEARTBEAT_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 128
SCTP:        DATA_CHUNK, 7/0/100/4F2D8236
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 144
SCTP:        SACK_CHUNK, TSN ack: 4F2D8236, rwnd 9000, num frags 0
SCTP:        DATA_CHUNK, 7/0/100/7DD7E425
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 7DD7E424, rwnd 18000, num frags 0
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28
SCTP:        SACK_CHUNK, TSN ack: 7DD7E425, rwnd 17900, num frags 0
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 128
SCTP:        DATA_CHUNK, 4/1/100/4F2D8237
```

The table below describes the significant fields shown in the display.

Table 9: debug ip sctp segments Field Descriptions

Field	Description
s	Source address and port.
d	Destination address and port.
len	Length of chunk, in bytes.
Tag	The identifier for an initialization chunk.
TSN	Transmission sequence number.
rwnd	Receiver window value.
num frags	Number of fragments received.

Field	Description
7 / 0 / 100 / 4F2D8236	(Data chunks) Stream number / datagram sequence number / chunk length, in bytes / chunk transmission sequence number.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
debug ip sctp segmentv	Shows every datagram that is sent or received and the chunks that are contained in each. This is the verbose form of the output, and it shows detailed information for each chunk type.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp segmentv

To show verbose diagnostics for every datagram that is sent or received with Stream Control Transmission Protocol (SCTP), use the **debug ip sctp segmentv**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp segmentv
no debug ip sctp segmentv

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp segmentv** command provides the verbose form of the output for datagrams. For the simple form, use the **debug ip sctp segments** command.

Caution The **debug ip sctp segmentv** command generates multiple lines of output for each datagram sent and received. It should be used with extreme caution in a live network.

Examples

The following output shows an example in which an association is established, a few heartbeats are sent, the remote endpoint fails, and the association is restarted:

```
Router# debug ip sctp segmentv
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 56, ver tag 0
SCTP:        INIT_CHUNK, len 42
SCTP:        Initiate Tag: B131ED6A, Initial TSN: B131ED6A, rwnd 9000
SCTP:        Streams Inbound: 13, Outbound: 13
SCTP:        IP Addr: 10.1.0.2
SCTP:        IP Addr: 10.2.0.2
SCTP:        Supported addr types: 5
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 56, ver tag 0
SCTP:        INIT_CHUNK, len 42
SCTP:        Initiate Tag: 5516B2F3, Initial TSN: 5516B2F3, rwnd 18000
SCTP:        Streams Inbound: 13, Outbound: 13
SCTP:        IP Addr: 10.5.0.4
SCTP:        IP Addr: 10.6.0.4
SCTP:        Supported addr types: 5
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 136, ver tag 5516B2F3
SCTP:        INIT_ACK_CHUNK, len 124
SCTP:        Initiate Tag: B131ED6A, Initial TSN: B131ED6A, rwnd 9000
SCTP:        Streams Inbound: 13, Outbound: 13
SCTP:        Responder cookie len 88
SCTP:        IP Addr: 10.1.0.2
SCTP:        IP Addr: 10.2.0.2
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 100, ver tag B131ED6A
SCTP:        COOKIE_ECHO_CHUNK, len 88
SCTP: Sent:  Assoc NULL: s=10.1.0.2  8787, d=10.5.0.4  8787, len 16, ver tag 5516B2F3
SCTP:        COOKIE_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 144, ver tag B131ED6A
SCTP:        SACK_CHUNK, len 16
SCTP:        TSN ack: (0xB131ED69)
SCTP:        Rcv win credit: 18000
SCTP:        Num frags: 0
SCTP:        DATA_CHUNK, flags 3, chunkLen 116
SCTP:        DATA_CHUNK, 0/0/100/5516B2F3
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 28, ver tag 5516B2F3
SCTP:        SACK_CHUNK, len 16
SCTP:        TSN ack: (0x5516B2F3)
SCTP:        Rcv win credit: 8900
SCTP:        Num frags: 0
SCTP: Sent:  Assoc 0: s=10.1.0.2  8787, d=10.5.0.4  8787, len 128, ver tag 5516B2F3
SCTP:        DATA_CHUNK, flags 3, chunkLen 116
SCTP:        DATA_CHUNK, 0/0/100/B131ED6A
SCTP: Recv:  Assoc 0: s=10.6.0.4  8787, d=10.2.0.2  8787, len 44, ver tag B131ED6A
SCTP:        HEARTBEAT_CHUNK
SCTP: Sent:  Assoc 0: s=10.2.0.2  8787, d=10.6.0.4  8787, len 44, ver tag 5516B2F3
```

```
SCTP:        HEARTBEAT_ACK_CHUNK
SCTP: Recv:  Assoc 0: s=10.5.0.4  8787, d=10.1.0.2  8787, len 28, ver tag B131ED6A
SCTP:        SACK_CHUNK, len 16
```

The table below describes the significant fields shown in the display.

Table 10: debug ip sctp segmentv Field Descriptions

Field	Description
s	Source address and port.
d	Destination address and port.
len	Length of chunk, in bytes.
ver tag	Verification identifier.
Tag	The identifier for an initialization chunk.
TSN	Transmission sequence number.
rwnd	Receive window value.
Rcv win credit	Receive window value. Same as rwnd.
Num frags	Number of fragments received.
0/0/100/5516B2F3	(Data chunks) Stream number / datagram sequence number / chunk length, in bytes / chunk transmission sequence number.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
debug ip sctp segments	Shows short diagnostics for every datagram that is sent or received with SCTP.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.

Command	Description
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp signal

To show signals that are sent from Stream Control Transmission Protocol (SCTP) to the application or upper-layer protocol (ULP), use the **debug ip sctp signal**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp signal
no debug ip sctp signal

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp signal** command can be used to see if the current associations are stable or not. Because it generates output only on state transitions, it is safe to use in a live environment. It still should be used with caution, however, depending on the number of associations being handled by the system and the stability of the network.

The **debug ip sctp state** command is often used at the same time as the **debug ip sctp signal** command. Using the two commands together gives good insight into the stability of associations.

Examples

In the following example, a new association is requested and established. The peer then restarts the association and notes that the association failed and is being reestablished. The local peer then indicates that the association has failed because it has tried to retransmit the specified chunk more than the maximum number of times without success. As a result, the association fails (because of communication loss) and is terminated. The ULP requests that the association be attempted again, and this attempt succeeds. A shutdown is then received from the remote peer, and the local peer enters the shutdown acknowledge sent state, which is followed by the association being terminated. Again, another association attempt is made and succeeds.

```
Router# debug ip sctp signal
Router# debug ip sctp state
<new assoc attempt>
00:20:08: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:20:15: SCTP: Assoc 0:  state COOKIE_WAIT -> ESTABLISHED
00:20:15: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
00:21:03: SCTP: Assoc 0: Restart rcvd from peer
00:21:03: SCTP: Assoc 0: Sent ASSOC_RESTART signal
00:21:04: SCTP: Assoc 0: chunk 62EA7F40 retransmitted more than max times, failing assoc
```

```
00:21:04: SCTP: Assoc 0: Sent ASSOC_FAILED signal, reason: SCTP_COMM_LOST
00:21:04: SCTP: Assoc 0: Sent ASSOC_TERMINATE signal
00:21:04: SCTP: Assoc 0:  state ESTABLISHED -> CLOSED
<new assoc attempt>
00:21:04: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:21:04: SCTP: Assoc 0:  state COOKIE_WAIT -> COOKIE_ECHOED
00:21:04: SCTP: Assoc 0:  state COOKIE_ECHOED -> ESTABLISHED
00:21:04: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
00:21:04: SCTP: Assoc 0: Sent TERMINATE_PENDING signal
00:21:04: SCTP: Assoc 0:  state ESTABLISHED -> SHUTDOWN_ACKSENT
00:21:04: SCTP: Assoc 0: Sent ASSOC_TERMINATE signal
00:21:04: SCTP: Assoc 0:  state SHUTDOWN_ACKSENT -> CLOSED
<new assoc attempt>
00:21:04: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:21:04: SCTP: Assoc 0:  state COOKIE_WAIT -> COOKIE_ECHOED
00:21:04: SCTP: Assoc 0:  state COOKIE_ECHOED -> ESTABLISHED
00:21:04: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
```

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
debug ip sctp state	Shows SCTP state transitions.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp sndchunks

To show information about chunks that are being sent to remote Stream Control Transmission Protocol (SCTP) peers, use the **debug ip sctp sndchunks**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp sndchunks
no debug ip sctp sndchunks

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp sndchunks** command provides the following information:

- Application send requests from the local SCTP peer
- Chunks being bundled and sent to the remote peer
- Processing of the selective acknowledgments (SACKs) from the remote peer, indicating which chunks were successfully received
- Chunks that are marked for retransmission

Caution The **debug ip sctp sndchunks** command generates large amounts of data if there is any significant amount of traffic flowing. It should be used with extreme caution in live networks.

Examples

The following example shows output for the **debug ip sctp sndchunks** command for a case in which data chunks are being sent, with some of them marked for retransmission:

```
Router# debug ip sctp sndchunks
SCTP: Assoc 0: ApplSend, chunk: 0/10412/100/A23134F8 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10443/100/A23134F9 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10448/100/A231355C to 10.5.0.4
SCTP: Assoc 0: Set oldest chunk for dest 10.5.0.4 to TSN A23134F8
SCTP: Assoc 0: Bundling data, added 0/10412/100/A23134F8, outstanding 100
SCTP: Assoc 0: Bundling data, added 5/10443/100/A23134F9, outstanding 200
SCTP: Assoc 0: Bundling data, added 4/10545/100/A23134FA, outstanding 300
SCTP: Assoc 0: Bundling data, added 10/10371/100/A23134FB, outstanding 400
SCTP: Assoc 0: Bundling data, added 11/10382/100/A23134FC, outstanding 500
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A231350F, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313510
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A2313527, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313528
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A231353F, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313540
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A2313557, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A2313558
SCTP: Assoc 0: ApplSend, chunk: 10/10385/100/A23135BE to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 8/10230/100/A23135BF to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10459/100/A23135C0 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 4/10558/100/A23135C1 to 10.5.0.4
SCTP: Assoc 0: Set oldest chunk for dest 10.5.0.4 to TSN A231355D
SCTP: Assoc 0: Bundling data, added 5/10449/100/A231355D, outstanding 100
SCTP: Assoc 0: Bundling data, added 3/10490/100/A231355E, outstanding 200
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A23135A4, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A23135A5
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A23135BC, numFrags=0
SCTP: Assoc 0: Reset oldest chunk on addr 10.5.0.4 to A23135BD
```

```
SCTP: Assoc 0: Process Sack Chunk, CumTSN=A23135C1, numFrags=0
SCTP: Assoc 0: ApplSend, chunk: 5/10460/100/A23135C2 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 5/10461/100/A23135C3 to 10.5.0.4
SCTP: Assoc 0: ApplSend, chunk: 11/10403/100/A2313626 to 10.5.0.4
SCTP: Assoc 0: Set oldest chunk for dest 10.5.0.4 to TSN A23135C2
SCTP: Assoc 0: Bundling data, added 5/10460/100/A23135C2, outstanding 100
SCTP: Assoc 0: Bundling data, added 5/10461/100/A23135C3, outstanding 200
SCTP: Assoc 0: Bundling data, added 5/10462/100/A23135C4, outstanding 300
SCTP: Assoc 0: Bundling data, added 4/10559/100/A23135C5, outstanding 400
SCTP: Assoc 0: Bundling data, added 4/10560/100/A23135C6, outstanding 500
SCTP: Assoc 0: Bundled 12 chunk(s) in next dgram to 10.5.0.4
SCTP: Assoc 0: Bundling data, added 1/10418/100/A2313622, outstanding 9700
SCTP: Assoc 0: Bundling data, added 3/10502/100/A2313623, outstanding 9800
SCTP: Assoc 0: Bundling data, added 7/10482/100/A2313624, outstanding 9900
SCTP: Assoc 0: Bundling data, added 3/10503/100/A2313625, outstanding 10000
SCTP: Assoc 0: Bundling data, added 11/10403/100/A2313626, outstanding 10100
SCTP: Assoc 0: Bundled 5 chunk(s) in next dgram to 10.5.0.4
SCTP: Assoc 0: Mark chunk A23135C2 for retrans
SCTP: Assoc 0: Mark chunk A23135C3 for retrans
SCTP: Assoc 0: Mark chunk A23135C4 for retrans
SCTP: Assoc 0: Mark chunk A23135C5 for retrans
SCTP: Assoc 0: Mark chunk A23135C6 for retrans
SCTP: Assoc 0: Mark chunk A23135C7 for retrans
SCTP: Assoc 0: Mark chunk A23135C8 for retrans
SCTP: Assoc 0: Mark chunk A23135C9 for retrans
SCTP: Assoc 0: Mark chunk A23135CA for retrans
SCTP: Assoc 0: Bundled 6 chunk(s) in next dgram to 10.6.0.4
SCTP: Assoc 0: Mark chunk A23135C2 for retrans
SCTP: Assoc 0: Mark chunk A23135C3 for retrans
SCTP: Assoc 0: Mark chunk A23135C4 for retrans
```

The table below describes the significant fields shown in the display.

Table 11: debug ip sctp sndchunks Field Descriptions

Field	Description
0 / 10412 / 100 / A23134F8	Stream number / datagram sequence number / chunk length, in bytes / chunk transmission sequence number.
outstanding	Number of bytes outstanding to the specified destination address.
CumTSN	Cumulative transmission sequence number (TSN).
numFrags	Number of fragments sent.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.

Command	Description
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp state

To show state transitions in the Stream Control Transmission Protocol (SCTP), use the **debug ip sctp state**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp state
no debug ip sctp state

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

The **debug ip sctp state** command can be used to see if the current associations are stable or not. Because it generates output only on state transitions, it is safe to use in a live environment. It still should be used with caution, however, depending on the number of associations being handled by the system and the stability of the network.

The **debug ip sctp state** command is often used at the same time as the **debug ip sctp signal** command. Using the two commands together gives good insight into the stability of associations.

Examples

In the following example, a new association is requested and established. The peer then restarts the association and notes that the association failed and is being reestablished. The local peer then indicates that the association has failed because it has tried to retransmit the specified chunk more than the maximum number of times without success. As a result, the association fails (because of communication loss) and is terminated. The upper-layer protocol (ULP) requests that the association be attempted again, and this attempt succeeds. A shutdown is then received from the remote peer, and the local peer enters the shutdown acknowledge sent state, which is followed by the association being terminated. Again, another association attempt is made and succeeds.

```
Router# debug ip sctp signal
Router# debug ip sctp state
<new assoc attempt>
00:20:08: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:20:15: SCTP: Assoc 0:  state COOKIE_WAIT -> ESTABLISHED
00:20:15: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
00:21:03: SCTP: Assoc 0: Restart rcvd from peer
00:21:03: SCTP: Assoc 0: Sent ASSOC_RESTART signal
00:21:04: SCTP: Assoc 0: chunk 62EA7F40 retransmitted more than max times, failing assoc
00:21:04: SCTP: Assoc 0: Sent ASSOC_FAILED signal, reason: SCTP_COMM_LOST
00:21:04: SCTP: Assoc 0: Sent ASSOC_TERMINATE signal
00:21:04: SCTP: Assoc 0:  state ESTABLISHED -> CLOSED
<new assoc attempt>
00:21:04: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:21:04: SCTP: Assoc 0:  state COOKIE_WAIT -> COOKIE_ECHOED
00:21:04: SCTP: Assoc 0:  state COOKIE_ECHOED -> ESTABLISHED
00:21:04: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
00:21:04: SCTP: Assoc 0: Sent TERMINATE_PENDING signal
00:21:04: SCTP: Assoc 0:  state ESTABLISHED -> SHUTDOWN_ACKSENT
00:21:04: SCTP: Assoc 0: Sent ASSOC_TERMINATE signal
00:21:04: SCTP: Assoc 0:  state SHUTDOWN_ACKSENT -> CLOSED
<new assoc attempt>
00:21:04: SCTP: Assoc 0:  state CLOSED -> COOKIE_WAIT
00:21:04: SCTP: Assoc 0:  state COOKIE_WAIT -> COOKIE_ECHOED
00:21:04: SCTP: Assoc 0:  state COOKIE_ECHOED -> ESTABLISHED
00:21:04: SCTP: Assoc 0: Sent ASSOC_UP signal for CONFIGD_ASSOC
```

The table below describes the significant fields shown in the display.

Table 12: debug ip sctp state Field Descriptions

Field	Description
CLOSED -> COOKIE_WAIT	SCTP endpoint sends initialization chunk and moves to the COOKIE_WAIT state to wait for acknowledgment and a state cookie from the remote endpoint.
COOKIE_WAIT -> COOKIE_ECHOED	SCTP endpoint returns the state cookie to the remote endpoint and enters COOKIE_ECHOED state.
COOKIE_ECHOED -> ESTABLISHED	SCTP endpoint enters ESTABLISHED state after receiving acknowledgment that the state cookie has been received by the remote endpoint.
ESTABLISHED -> SHUTDOWN_ACKSENT	SCTP endpoint enters SHUTDOWN_ACKSENT state after receiving a shutdown message and sending a shutdown acknowledgment to the remote endpoint.
SHUTDOWN_ACKSENT -> CLOSED	SCTP endpoint enters CLOSED state.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
debug ip sctp signal	Shows signals that are sent from SCTP to the application or ULP.

Command	Description
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp timer

To provide information about Stream Control Transmission Protocol (SCTP) timers that are started, stopped, and triggering, use the **debug ip sctp timer**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp timer
no debug ip sctp timer

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

Many SCTP timers should not be restarted after they have been started once. For these timers, the first call succeeds in starting the timer, and subsequent calls do nothing until the timer either expires or is stopped. For example, the retransmission timer is started when the first chunk is sent, but then is not started again for subsequent chunks when there is outstanding data.

Caution The **debug ip sctp timer** command generates a significant amount of output. It should be used with extreme caution in a live network.

Examples

The following example shows the starting and stopping of various SCTP timers:

```
Router# debug ip sctp timer
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Timer BUNDLE triggered
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Stopping RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Starting RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Stopping RETRANS timer for destaddr 10.5.0.4
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
SCTP: Assoc 0: Stopping CUMSACK timer
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Assoc 0: Starting CUMSACK timer
SCTP: Timer already started, not restarting
```

The table below describes the significant fields shown in the display.

Table 13: debug ip sctp timer Field Descriptions

Field	Description
CUMSACK	Cumulative selective acknowledgment.
RETRANS	Retransmission.

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.

Command	Description
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sctp warnings

To display diagnostic information about unusual situations in Stream Control Transmission Protocol (SCTP), use the **debug ip sctp warnings**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sctp warnings
no debug ip sctp warnings

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(4)T	This command was introduced.

Usage Guidelines

In a live system, the debugging messages for performance, state, signal, and warnings are the most useful. They show any association or destination address failures and can be used to monitor the stability of established associations.

The **debug ip sctp warnings** command displays information on any unusual situation that is encountered. These situations may or may not indicate problems, depending on the particulars of the situation.

Examples

The following example shows some events and conditions that are flagged as warnings:

```
Router# debug ip sctp warnings
SCTP: Assoc 0: No cookie in InitAck, discarding
SCTP: Assoc 0: Incoming INIT_ACK: inbound streams reqd 15, allowed 13
SCTP: Assoc 0: Incoming INIT_ACK request: outbound streams req'd 13, allowed 1
SCTP: Assoc 0: Remote verification tag in init ack is zero, discarding
SCTP: Remote verification tag in init is zero, discarding
SCTP: Assoc 0: Rwnd less than min allowed (1500) in incoming INITACK, rcvd 0
SCTP: Assoc 0: Rwnd less than min allowed (1500) in incoming INITACK, rcvd 1499
SCTP: Rwnd in INIT too small (0), discarding
SCTP: Rwnd in INIT too small (1499), discarding
SCTP: Unknown INIT param 16537 (0x4099), length 8
```

```
SCTP: Assoc 0: Unknown INITACK param 153 (0x99), length 8
SCTP: Assoc 0: No cookie in InitAck, discarding
SCTP: Assoc 0: No cookie in InitAck, discarding
SCTP: Processing INIT, invalid param len 0, discarding...
SCTP: Assoc 0: Processing INITACK, invalid param len 0, discarding...
```

Related Commands

Command	Description
clear ip sctp statistics	Empties the buffer that holds SCTP statistics.
debug ip sctp congestion	Shows a list of all current SCTP associations.
show ip sctp association parameters	Shows the parameters configured for the association defined by the association identifier.
show ip sctp association statistics	Shows the current statistics for the association defined by the association identifier.
show ip sctp errors	Shows error counts logged by SCTP.
show ip sctp instances	Shows all currently defined SCTP instances.
show ip sctp statistics	Shows overall statistics counts for SCTP.
show iua as	Shows information about the current condition of an application server.
show iua asp	Shows information about the current condition of an application server process.

debug ip sd

To display all session directory (SD) announcements received, use the **debug ip sd**command in privileged EXEC mode. To disable debugging output, use the **no**form of this command.

debug ip sd
no debug ip sd

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

This command shows session directory announcements for multicast IP. Use it to observe multicast activity.

Examples

The following is sample output from the **debug ip sd** command:

```
Router# debug ip sd
SD: Announcement from 172.16.58.81 on Serial0.1, 146 bytes
    s=*cisco: CBONE Audio
    i=cisco internal-only audio conference
    o=dino@dino-ss20.cisco.com
    c=224.0.255.1 16 2891478496 2892688096
    m=audio 31372 1700
SD: Announcement from 172.22.246.68 on Serial0.1, 147 bytes
    s=IMS: U.S. Senate
    i=U.S. Senate at http://town.hall.org/radio/live.html
    o=carl@also.radio.com
    c=224.2.252.231 95 0 0
    m=audio 36572 2642
    a=fmt:gsm
```

The table below describes the significant fields shown in the display.

Table 14: debug ip sd Field Descriptions

Field	Description
SD	Session directory event.
Announcement from	Address sending the SD announcement.
on Serial0.1	Interface receiving the announcement.
146 bytes	Size of the announcement event.
s=	Session name being advertised.
i=	Information providing a descriptive name for the session.
o=	Origin of the session, either an IP address or a name.
c=	Connect description showing address and number of hops.
m=	Media description that includes media type, port number, and ID.

Related Commands

Command	Description
debug ip dvmrp	Displays information on DVMRP packets received and sent.
debug ip igmp	Displays IGMP packets received and sent, and IGMP host-related events.
debug ip mbgp dampening	Logs route flap dampening activity related to MBGP.
debug ip mrouting	Displays changes to the IP multicast routing table.
debug ip pim	Displays PIM packets received and sent, and PIM-related events.

debug ip sdee

To enable debugging messages for Security Device Event Exchange (SDEE) notification events, use the **debug ip sdee** command in privileged EXEC mode. To disable SDEE debugging messages, use the **no** form of this command.

debug ip sdee [**alerts**] [**detail**] [**messages**] [**requests**] [**subscriptions**]
no debug ip sdee [**alerts**] [**detail**] [**messages**] [**requests**] [**subscriptions**]

Syntax Description

alerts	Displays new alerts that are reported to SDEE from IPS.
detail	Displays detailed SDEE messages.
messages	Displays error and status messages that are reported to SDEE from IPS.
requests	Displays SDEE client requests.
subscriptions	Displays SDEE client subscription requests.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Examples

The following is sample SDEE debug output. In this example, you can see which messages correspond to SDEE alerts, requests, and subscriptions.

```
Router# debug ip sdee alerts requests subscriptions
5d00h:SDEE:got request from client at 10.0.0.2
5d00h:SDEE:reported 13 events for client at 10.0.0.2
5d00h:SDEE:GET request for client 10.0.0.2 subscription IDS1720:0
5d00h:SDEE:reported 50 events for client 10.0.0.2 subscription IDS1720:0
5d00h: SDEE alert:sigid 2004 name ICMP Echo Req from 10.0.0.2 time 1021174067
5d00h: SDEE alert:sigid 2004 name ICMP Echo Req from 10.0.0.2 time 1021174071
5d00h: SDEE alert:sigid 2004 name ICMP Echo Req from 10.0.0.2 time 1021174072
5d00h: SDEE alert:sigid 2004 name ICMP Echo Req from 10.0.0.2 time 1021175127
5d00h:SDEE:missed events for IDS1720:0
```

Related Commands

Command	Description
ip ips notify	Specifies the method of event notification.
ip sdee events	Sets the maximum number of SDEE events that can be stored in the event buffer.
ip sdee subscriptions	Sets the maximum number of SDEE subscriptions that can be open simultaneously.

debug ip security

To display IP security option processing, use the **debug ip security** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip security
no debug ip security

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

The **debug ip security** command displays information for both basic and extended IP security options. For interfaces where **ip security** is configured, each IP packet processed for that interface results in debugging output regardless of whether the packet contains IP security options. IP packets processed for other interfaces that also contain IP security information also trigger debugging output. Some additional IP security debugging information is also controlled by the **debug ip packet** command in privileged EXEC mode.

Caution Because the **debug ip security** command generates a substantial amount of output for every IP packet processed, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Examples

The following is sample output from the **debug ip security** command:

```
Router# debug ip security
IP Security: src 172.24.72.52 dst 172.24.72.53, number of BSO 1
     idb: NULL
     pak: insert (0xFF) 0x0
IP Security: BSO postroute: SECINSERT changed to secret (0x5A) 0x10
IP Security: src 172.24.72.53 dst 172.24.72.52, number of BSO 1
     idb: secret (0x6) 0x10 to secret (0x6) 0x10, no implicit
          def secret (0x6) 0x10
     pak: secret (0x5A) 0x10
IP Security: checking BSO 0x10 against [0x10 0x10]
IP Security: classified BSO as secret (0x5A) 0x10
```

The table below describes significant fields shown in the display.

Table 15: debug ip security Field Descriptions

Field	Description
number of BSO	Indicates the number of basic security options found in the packet.
idb	Provides information on the security configuration for the incoming interface.
pak	Provides information on the security classification of the incoming packet.
src	Indicates the source IP address.
dst	Indicates the destination IP address.

The following line indicates that the packet was locally generated, and it has been classified with the internally significant security level “insert” (0xff) and authority information of 0x0:

```
idb: NULL
pak: insert (0xff) 0x0
```

The following line indicates that the packet was received via an interface with dedicated IP security configured. Specifically, the interface is configured at security level “secret” and with authority information of 0x0. The packet itself was classified at level “secret” (0x5a) and authority information of 0x10.

```
idb: secret (0x6) 0x10 to secret (0x6) 0x10, no implicit
     def secret (0x6) 0x10
pak: secret (0x5A) 0x10
```

debug ip sla error

To enable debugging output of Cisco IOS IP Service Level Agreements (SLAs) operation run-time errors, use the **debug ip sla error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sla error [{*operation-number* | **ep-api** | **event-publisher**}]
no debug ip sla error [{*operation-number* | **ep-api** | **event-publisher**}]

Syntax Description

operation-number	(Optional) Identification number of the operation for which debugging output is to be enabled.
ep-api	(Optional) Enables IP SLAs Event Publisher application programming interface (API) debug messages.
event-publisher	(Optional) Enables IP SLAs Event Publisher debug messages.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(4)T	This command was introduced. This command replaces the **debug ip sla monitor error**command.
12.0(32)SY	This command was integrated into Cisco IOS Release 12.0(32)SY.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB. This command replaces the **debug rtr error**command.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB. This command replaces the **debug ip sla monitor error** command.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI. This command replaces the **debug ip sla monitor error** command.
12.4(22)T	This command was modified. The **ep-api** and **event-publisher** keywords were added.
12.2(33)SRE	This command was modified. The **ep-api** and **event-publisher** keywords were added.

Usage Guidelines

The **debug ip sla error** *operation-number*command displays run-time errors. When an operation number other than 0 is specified, all run-time errors for that operation are displayed when the operation is active. When the operation number is 0, all run-time errors relating to the IP SLAs scheduler process are displayed. When no operation number is specified, all run-time errors for all active operations configured on the router are displayed.

Note Use the **debug ip sla error**command before using the **debug ip sla trace** command because the **debug ip sla error** command generates a lesser amount of debugging output.

The **debug ip sla error** command is supported in IPv4 networks. This command can also be used to enable debugging output for an IP SLAs operation that supports IPv6 addresses.

Examples

The following is sample output from the **debug ip sla error** command. The output indicates failure because the target is not there or because the responder is not enabled on the target.

```
Router# debug ip sla error
May  5 05:00:35.483: control message failure:1
May  5 05:01:35.003: control message failure:1
May  5 05:02:34.527: control message failure:1
May  5 05:03:34.039: control message failure:1
May  5 05:04:33.563: control message failure:1
May  5 05:05:33.099: control message failure:1
May  5 05:06:32.596: control message failure:1
May  5 05:07:32.119: control message failure:1
May  5 05:08:31.643: control message failure:1
May  5 05:09:31.167: control message failure:1
May  5 05:10:30.683: control message failure:1
```

Related Commands

Command	Description
debug ip sla trace	Traces the execution of an IP SLAs operation.

debug ip sla ethernet-monitor

To enable debugging output for a Cisco IOS IP Service Level Agreements (SLAs) Ethernet operation, use the **debug ip sla ethernet-monitor** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sla ethernet-monitor [*operation-number*]
no debug ip sla ethernet-monitor [*operation-number*]

Syntax Description

operation-number	(Optional) Number of the Ethernet operation for which the debugging output will be displayed.

Command Default

Debugging activity for a Cisco IOS IP SLAs Ethernet operation does not occur.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRB	This command was introduced.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.4(20)T	This command was integrated into Cisco IOS Release 12.4(20)T.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

Examples

The following is sample output from the **debug ip sla ethernet-monitor** command:

```
Router# debug ip sla ethernet-monitor
00:00:15: IP SLAs Auto Ethernet(0):vlan = 2, domain = DOMAIN_OPERATOR_L3_1, mpid = 6322
                                 from CFM
00:00:15: IP SLAs Auto Ethernet(0):saaHandleEventFromCFM::Received Event from CFM
00:00:15: IP SLAs Auto Ethernet(0):Event::ECFM_SAA_EV_MEP_ADD
00:00:15: IP SLAs Auto Ethernet(0):1 auto-probes found for domain = DOMAIN_OPERATOR_L3_1
and vlan = 2
00:00:15: IP SLAs Auto Ethernet(0):autoProbe probe_id = 1
00:00:15: IP SLAs Auto Ethernet(0):0 Probes already running in auto-probe = 1
00:00:15: IP SLAs Auto Ethernet(1):starting probe with freq = 20 sec
00:00:15: IP SLAs Auto Ethernet(1):starting probe 100001
```

Related Commands

Command	Description
ip sla	Begins configuration for an IP SLAs operation and enters IP SLA configuration mode.
ip sla ethernet-monitor	Begins configuration for an IP SLAs auto Ethernet operation and enters IP SLA Ethernet monitor configuration mode.

debug ip sla monitor error

Note Effective with Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the **debug ip sla monitor error**command is replaced by the **debug ip sla error**command. See the **debug ip sla error**command for more information.

To enable debugging output of Cisco IOS IP Service Level Agreements (SLAs) operation run-time errors, use the **debug ip sla monitor error**command in privileged EXEC mode. To disable debugging output, use the **no**form of this command.

debug ip sla monitor error [*operation-number*]
no debug ip sla monitor error [*operation-number*]

Syntax Description

operation-number	(Optional) Identification number of the operation for which debugging output is to be enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced. This command replaces the **debug rtr error**command.
12.4(4)T	This command was replaced by the **debug ip sla error**command.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.2(33)SB	This command was replaced by the **debug ip sla error** command.
12.2(33)SXI	This command was replaced by the **debug ip sla error** command.

Usage Guidelines

The **debug ip sla monitor error**command displays run-time errors. When an operation number other than 0 is specified, all run-time errors for that operation are displayed when the operation is active. When the operation number is 0, all run-time errors relating to the IP SLAs scheduler process are displayed. When no operation number is specified, all run-time errors for all active operations configured on the router are displayed.

Note Use the **debug ip sla monitor error**command before using the **debug ip sla monitor trace** command because the **debug ip sla monitor error** command generates a lesser amount of debugging output.

Examples

The following is sample output from the **debug ip sla monitor error** command. The output indicates failure because the target is not there or because the responder is not enabled on the target. All debugging output for IP SLAs (including the output from the **debug ip sla monitor trace** command) has the format shown in the table below.

```
Router# debug ip sla monitor error
May  5 05:00:35.483: control message failure:1
May  5 05:01:35.003: control message failure:1
May  5 05:02:34.527: control message failure:1
May  5 05:03:34.039: control message failure:1
May  5 05:04:33.563: control message failure:1
May  5 05:05:33.099: control message failure:1
May  5 05:06:32.596: control message failure:1
May  5 05:07:32.119: control message failure:1
May  5 05:08:31.643: control message failure:1
May  5 05:09:31.167: control message failure:1
May  5 05:10:30.683: control message failure:1
```

The table below describes the significant fields shown in the display.

Table 16: debug ip sla monitor error Field Descriptions

Field	Description
IP SLA Monitor 1	Number of the operation generating the message.

Field	Description
Error Return Code	Message identifier indicating the error type (or error itself).
LU0 IP SLA Monitor Probe 1	Name of the process generating the message.
in echoTarget on call luReceive LuApiReturnCode of InvalidHandle - invalid host name or API handle	Supplemental messages that pertain to the message identifier.

Related Commands

Command	Description
debug ip sla monitor trace	Traces the execution of an IP SLAs operation.

debug ip sla monitor mpls-lsp-monitor

Note Effective with Cisco IOS Release 12.2(33)SB, the **debug ip sla monitor mpls-lsp-monitor**command is replaced by the **debug ip sla mpls-lsp-monitor**command. See the **debug ip sla mpls-lsp-monitor**command for more information.

To enable debugging output for the IP Service Level Agreements (SLAs) label switched path (LSP) Health Monitor, use the **debug ip sla monitor mpls-lsp-monitor**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sla monitor mpls-lsp-monitor [*operation-number*]
no debug ip sla monitor mpls-lsp-monitor [*operation-number*]

Syntax Description

operation-number	(Optional) Number of the LSP Health Monitor operation for which the debugging output will be displayed.

Command Default Debugging is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(31)SB2	This command was introduced.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.2(33)SB	This command was replaced by the **debug ip sla mpls-lsp-monitor** command.

Examples

The following is sample output from the **debug ip sla monitor mpls-lsp-monitor** command:

```
Router# debug ip sla monitor mpls-lsp-monitor
IP SLA Monitor MPLSLM debugging for all entries is on
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding vrf red into tree entry 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding Probe 100005
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding ProbeID 100005 to tree entry 10.10.10.8 (1)
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding vrf blue into tree entry 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Duplicate in AddQ 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding vrf green into tree entry 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Duplicate in AddQ 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Added Probe(s) 100005 will be scheduled after 26
secs over schedule period 60
```

Related Commands

Command	Description
auto ip sla mpls-lsp-monitor	Begins configuration for an IP SLAs LSP Health Monitor operation and enters auto IP SLA MPLS configuration mode.

debug ip sla trace

To trace the execution of a Cisco IOS IP Service Level Agreements (SLAs) operation, use the **debug ip sla trace**command in privileged EXEC mode. To disable trace debugging output, use the **no**form of this command.

debug ip sla trace [{*operation-number* | **ep-api** | **event-publisher**}]
no debug ip sla trace [{*operation-number* | **ep-api** | **event-publisher**}]

Syntax Description

operation-number	(Optional) Identification number of the operation for which debugging output is to be enabled.
ep-api	(Optional) Enables IP SLAs Event Publisher API debugging output.
event-publisher	(Optional) Enables IP SLAs Event Publisher debugging output.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(4)T	This command was introduced. This command replaces the **debug ip sla monitor trace**command.
12.0(32)SY	This command was integrated into Cisco IOS Release 12.0(32)SY.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB. This command replaces the **debug rtr trace**command.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB. This command replaces the **debug ip sla monitor trace** command.

Release	Modification
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI. This command replaces the **debug ip sla monitor trace** command.
12.4(22)T	This command was modified. The **ap-api** and **event-publisher** keywords were added.
12.2(33)SRE	This command was modified. The **ep-api** and **event-publisher** keywords were added.

Usage Guidelines

The **debug ip sla trace** *operation-number* command traces the execution of an IP SLAs operation. When an operation number other than 0 is specified, execution for that operation is traced. When the operation number is 0, the IP SLAs scheduler process is traced. When no operation number is specified, all active operations are traced.

The **debug ip sla trace** command also enables the **debug ip sla error** command for the specified operation. However, the **no debug ip sla trace** command does not disable the **debug ip sla error** command. You must manually disable the command by using the **no debug ip sla error** command.

All debugging output (including **debug ip sla error** command output) has the format shown in the **debug ip sla error** command output example.

Note The **debug ip sla trace**command can generate a large number of debug messages. First use the **debug ip sla error** command, and then use the **debug ip sla trace** on a per-operation basis.

Examples

The following is sample output from the **debug ip sla trace** command. In this example, an operation is traced through a single operation attempt: the setup of a connection to the target, and the attempt at an echo to calculate UDP packet response time.

```
Router# debug ip sla trace
May  5 05:25:08.584:rtt hash insert :3.0.0.3 3383
May  5 05:25:08.584:    source=3.0.0.3(3383)  dest-ip=5.0.0.1(9)
May  5 05:25:08.588:sending control msg:
May  5 05:25:08.588: Ver:1 ID:51 Len:52
May  5 05:25:08.592:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:25:08.607:receiving reply
May  5 05:25:08.607: Ver:1 ID:51 Len:8
May  5 05:25:08.623:    local delta:8
May  5 05:25:08.627:    delta from responder:1
May  5 05:25:08.627:    received <16> bytes and      responseTime = 3 (ms)
May  5 05:25:08.631:rtt hash remove:3.0.0.3 3383IP SLA Monitor 1:Starting An Echo Operation
 - IP SLA Monitor Probe 1
May  5 05:26:08.104:rtt hash insert :3.0.0.3 2974
May  5 05:26:08.104:    source=3.0.0.3(2974)  dest-ip=5.0.0.1(9)
May  5 05:26:08.108:sending control msg:
May  5 05:26:08.108: Ver:1 ID:52 Len:52
May  5 05:26:08.112:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:26:08.127:receiving reply
May  5 05:26:08.127: Ver:1 ID:52 Len:8
May  5 05:26:08.143:    local delta:8
May  5 05:26:08.147:    delta from responder:1
May  5 05:26:08.147:    received <16> bytes and      responseTime = 3 (ms)
May  5 05:26:08.151:rtt hash remove:3.0.0.3 2974IP SLA Monitor 1:Starting An Echo Operation
 - IP SLA Monitor Probe 1
```

Related Commands

Command	Description
debug ip sla error	Enables debugging output of IP SLAs operation run-time errors.

debug ip sla mpls-lsp-monitor

Note Effective with Cisco IOS Release 15.1(1)S, the **debug ip sla mpls-lsp-monitor** command was replaced by the **debug ip sla trace mpls-lsp-monitor** command. See the **debug ip sla trace mpls-lsp-monitor** command for more information.

To enable debugging output for the IP Service Level Agreements (SLAs) label switched path (LSP) Health Monitor, use the **debug ip sla mpls-lsp-monitor**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sla mpls-lsp-monitor [*operation-number*]
no debug ip sla mpls-lsp-monitor [*operation-number*]

Syntax Description

operation-number	(Optional) Number of the LSP Health Monitor operation for which the debugging output will be displayed.

Command Default

Debugging is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(6)T	This command was introduced.
12.0(32)SY	This command was integrated into Cisco IOS Release 12.0(32)SY.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB. This command replaces the **debug rtr mpls-lsp-monitor**command.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB. This command replaces the **debug ip sla monitor mpls-lsp-monitor** command.
15.1(1)S	This command was replaced by the **debug ip sla trace mpls-lsp-monitor** command.

Examples

The following is sample output from the **debug ip sla mpls-lsp-monitor** command:

```
Router# debug ip sla mpls-lsp-monitor
IP SLAs MPLSLM debugging for all entries is on
*Aug 19 19:59: IP SLAs MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLAs MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLAs MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLAs MPLSLM(1):Adding vrf red into tree entry 10.10.10.8
*Aug 19 19:59: IP SLAs MPLSLM(1):Adding Probe 100005
```

```
*Aug 19 19:59: IP SLAs MPLSLM(1):Adding ProbeID 100005 to tree entry 10.10.10.8 (1)
*Aug 19 19:59: IP SLAs MPLSLM(1):Adding vrf blue into tree entry 10.10.10.8
*Aug 19 19:59: IP SLAs MPLSLM(1):Duplicate in AddQ 10.10.10.8
*Aug 19 19:59: IP SLAs MPLSLM(1):Adding vrf green into tree entry 10.10.10.8
*Aug 19 19:59: IP SLAs MPLSLM(1):Duplicate in AddQ 10.10.10.8
*Aug 19 19:59: IP SLAs MPLSLM(1):Added Probe(s) 100005 will be scheduled after 26 secs over
 schedule period 60
```

Related Commands

Command	Description
debug ip sla trace mpls-lsp-monitor	Traces the execution of an IP SLAs LSP Health Monitor operation.

debug ip sla trace

To trace the execution of a Cisco IOS IP Service Level Agreements (SLAs) operation, use the **debug ip sla trace**command in privileged EXEC mode. To disable trace debugging output, use the **no**form of this command.

debug ip sla trace [{*operation-number* | **ep-api** | **event-publisher**}]
no debug ip sla trace [{*operation-number* | **ep-api** | **event-publisher**}]

Syntax Description

operation-number	(Optional) Identification number of the operation for which debugging output is to be enabled.
ep-api	(Optional) Enables IP SLAs Event Publisher API debugging output.
event-publisher	(Optional) Enables IP SLAs Event Publisher debugging output.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(4)T	This command was introduced. This command replaces the **debug ip sla monitor trace**command.
12.0(32)SY	This command was integrated into Cisco IOS Release 12.0(32)SY.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB. This command replaces the **debug rtr trace**command.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB. This command replaces the **debug ip sla monitor trace** command.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI. This command replaces the **debug ip sla monitor trace** command.
12.4(22)T	This command was modified. The **ap-api** and **event-publisher** keywords were added.
12.2(33)SRE	This command was modified. The **ep-api** and **event-publisher** keywords were added.

Usage Guidelines

The **debug ip sla trace** *operation-number* command traces the execution of an IP SLAs operation. When an operation number other than 0 is specified, execution for that operation is traced. When the operation number is 0, the IP SLAs scheduler process is traced. When no operation number is specified, all active operations are traced.

The **debug ip sla trace** command also enables the **debug ip sla error** command for the specified operation. However, the **no debug ip sla trace** command does not disable the **debug ip sla error** command. You must manually disable the command by using the **no debug ip sla error** command.

All debugging output (including **debug ip sla error** command output) has the format shown in the **debug ip sla error** command output example.

Note The **debug ip sla trace**command can generate a large number of debug messages. First use the **debug ip sla error** command, and then use the **debug ip sla trace** on a per-operation basis.

Examples

The following is sample output from the **debug ip sla trace** command. In this example, an operation is traced through a single operation attempt: the setup of a connection to the target, and the attempt at an echo to calculate UDP packet response time.

```
Router# debug ip sla trace
May  5 05:25:08.584:rtt hash insert :3.0.0.3 3383
May  5 05:25:08.584:    source=3.0.0.3(3383)  dest-ip=5.0.0.1(9)
May  5 05:25:08.588:sending control msg:
May  5 05:25:08.588: Ver:1 ID:51 Len:52
May  5 05:25:08.592:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:25:08.607:receiving reply
May  5 05:25:08.607: Ver:1 ID:51 Len:8
May  5 05:25:08.623:    local delta:8
May  5 05:25:08.627:    delta from responder:1
May  5 05:25:08.627:    received <16> bytes and      responseTime = 3 (ms)
May  5 05:25:08.631:rtt hash remove:3.0.0.3 3383IP SLA Monitor 1:Starting An Echo Operation
 - IP SLA Monitor Probe 1
May  5 05:26:08.104:rtt hash insert :3.0.0.3 2974
May  5 05:26:08.104:    source=3.0.0.3(2974)  dest-ip=5.0.0.1(9)
May  5 05:26:08.108:sending control msg:
May  5 05:26:08.108: Ver:1 ID:52 Len:52
May  5 05:26:08.112:cmd:command:RTT_CMD_UDP_PORT_ENABLE, ip:5.0.0.1, port:9, duration:5000
May  5 05:26:08.127:receiving reply
May  5 05:26:08.127: Ver:1 ID:52 Len:8
May  5 05:26:08.143:    local delta:8
May  5 05:26:08.147:    delta from responder:1
May  5 05:26:08.147:    received <16> bytes and      responseTime = 3 (ms)
May  5 05:26:08.151:rtt hash remove:3.0.0.3 2974IP SLA Monitor 1:Starting An Echo Operation
 - IP SLA Monitor Probe 1
```

Related Commands

Command	Description
debug ip sla error	Enables debugging output of IP SLAs operation run-time errors.

debug ip sla trace mpls-lsp-monitor

To trace the execution of an IP Service Level Agreements (SLAs) label switched path (LSP) Health Monitor operation, use the **debug ip sla trace mpls-lsp-monitor**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sla trace mpls-lsp-monitor [*operation-number*]
no debug ip sla mpls-lsp-monitor

Syntax Description

operation-number	(Optional) Number of the LSP Health Monitor operation for which the debugging output will be displayed. The range is 0 to 2147483647.

Command Default

Trace debugging of IP SLAs LSP Health Monitor operations is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(1)S	This command was introduced. This command replaces the **debug ip sla mpls-lsp-monitor** command.

Usage Guidelines

For Cisco IP SLAs Engine 3.0 in Cisco IOS Release 15.1(1)S, this command replaces the **debug ip sla mpls-lsp-monitor** command.

To determine the IP SLAs engine version, IP SLAs Engine 2.0 or 3.0, running on your Cisco router, use the **show ip application** command in privileged EXEC mode, as shown in the following example:

```
Router# show ip sla application
 IP Service Level Agreements
Version: Round Trip Time MIB 2.2.0, Infrastructure Engine-III
```

The **debug ip sla trace mpls-lsp-monitor** command traces the execution of IP SLAs LSP Health Monitor operations. When an operation number other than 0 is specified, execution for that operation is traced. When the operation number is 0, the IP SLAs scheduler process is traced. When no operation number is specified, all active LSP Health Monitor operations are traced.

This command also enables the **debug ip sla error** command for the specified operation. However, the **no debug ip sla trace mpls-lsp-monitor** command does not disable the **debug ip sla error** command. You must manually disable the command by using the **no debug ip sla error**command.

The **debug ip sla trace mpls-lsp-monitor** command can generate a large number of debug messages. To help reduce the number of debug messages, first use the **debug ip sla error** command and then use the **debug ip sla trace mpls-lsp-monitor** command on a per-operation basis.

Examples

The following is sample output from the **debug ip sla trace mpls-lsp-monitor** command:

```
Router# debug ip sla trace mpls-lsp-monitor
IP SLA Monitor MPLSLM debugging for all entries is on
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Next hop 10.10.10.8 added in AddQ
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Next hop 10.10.10.8 added in AddQ
```

```
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding vrf red into tree entry 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding Probe 100005
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding ProbeID 100005 to tree entry 10.10.10.8 (1)
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding vrf blue into tree entry 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Duplicate in AddQ 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Adding vrf green into tree entry 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Duplicate in AddQ 10.10.10.8
*Aug 19 19:59: IP SLA Monitor MPLSLM(1):Added Probe(s) 100005 will be scheduled after 26
secs over schedule period 60
```

Related Commands

Command	Description
debug ip sla error	Enables debugging output of Cisco IOS IP SLAs operation run-time errors.
debug ip sla mpls-lsp-monitor	Enables debugging output for Cisco IOS IP SLAs LSP Health Monitor operations in IP SLAs Engine 2.0.
show ip application	Displays global information about Cisco IOS IP SLAs.

debug ip sla trace twamp

To enable debugging output of Cisco IOS IP Service Level Agreements (SLAs) operation for Two-Way Active Measurement Protocol (TWAMP), use the **debug ip sla trace twamp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip sla trace twamp{**connection** [**source-ip** *ip-address*] | **control**{**reflector** | **server**} | **session** [**source-ip** *ip-address*]}
no debug ip sla trace twamp{**connection** [**source-ip** *ip-address*] | **control**{**reflector** | **server**} | **session** [**source-ip** *ip-address*]}

Syntax Description

connection	Displays communication messages between an IP SLAs TWAMP client and server.
source-ip *ip-address*	(Optional) Debug IP Performance Metrics (IPPM) TWAMP connections for the specified source. Specify the source using the IP address of the client device.
control	Displays communication messages between the IP SLAs TWAMP server and reflector.
reflector	Displays communication messages sent by an IP SLAs TWAMP reflector to the TWAMP server.
server	Displays communication messages sent by an IP SLAs TWAMP server to the TWAMP reflector.
session	Displays communication messages between an IP SLAs TWAMP sender and reflector.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(52)SE	This command was introduced.

Usage Guidelines

Use the **debug ip sla trace twamp** command to display communication messages between the client and server during a TWAMP session.

Note Use the **debug ip sla error twamp connection** command before using the **debug ip sla trace twamp connection** command because the **debug ip sla error twamp connection** command generates less debugging output.

Related Commands

Command	Description
debug ip sla error twamp	Displays exceptions during communication between the IP SLAs TWAMP client and server.

debug ip slb

To display debugging messages for the Cisco IOS Server Load Balancing (SLB) feature, use the **debug ip slb**command in user EXEC or privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip slb {**all** | **asn** [**msid**] | **conns** [*acl-number*] | **dfp** | **firewallfarm** | **fragments** | **gtp** | **icmp** | **kal-ap** | **natpool** | **probe** | **reals** | **replication** | **route** | **sessions** [{**asn** | **gtp** | **ipmobile** | **radius**}] | **sticky gtp imsi** | **vservers**}
no debug ip slb {**all** | **asn** [**msid**] | **conns** [*acl-number*] | **dfp** | **firewallfarm** | **fragments** | **gtp** | **icmp** | **kal-ap** | **natpool** | **probe** | **reals** | **replication** | **route** | **sessions** [{**asn** | **gtp** | **ipmobile** | **radius**}] | **sticky gtp imsi** | **vservers**}

Syntax Description

all	Displays all debugging messages for Cisco IOS SLB.
asn	Displays debugging messages related to Access Service Network (ASN) load balancing.
msid	(Optional) Displays debugging messages related to the ASN Mobile Station ID (MSID) sticky database.

conns *acl-number*	Displays debugging messages for all connections being handled by IOS SLB, including Wireless Session Protocol (WSP) events and states. The optional *acl-number* argument references an IP access control list (ACL). This argument limits the information displayed based on the client IP address, real server IP address, or virtual server IP address: • For simple ACLs, IOS SLB checks the client IP address. • For extended ACLs, IOS SLB checks the client real and virtual IP addresses. For more information about ACLs, refer to the “Configuring IP Services” chapter of the *Cisco IOS IP Configuration Guide* , Release 12.2.
dfp	Displays debugging messages for Dynamic Feedback Protocol (DFP). • To display debugging messages for the DFP agent subsystem, use the **debug ip dfp agent** command. • To display debugging messages for the general packet radio service (GPRS) DFP weight calculation, use the **debug gprs dfp**command.
firewallfarm	Displays debugging messages related to firewall load balancing.
fragments	Displays debugging messages related to the IOS SLB fragment database.
gtp	Displays all GPRS Tunneling Protocol (GTP)-related packet handler, gateway GPRS support node (GGSN), serving GPRS support node (SGSN), and Network Service Access Point Identifier (NSAPI) debugging messages for IOS SLB.
icmp	Displays all Internet Control Message Protocol debugging messages for IOS SLB.
kal-ap	Displays all KeepAlive Application Protocol (KAL-AP) debugging messages for IOS SLB.
natpool	Displays debugging messages related to the IOS SLB client Network Address Translation (NAT) pool.
probe	Displays debugging messages related to probes.
reals	Displays debugging messages for all real servers defined to IOS SLB.
replication	Displays debugging messages related to IOS SLB stateful backup virtual server.
route	Displays debugging messages for all routing handled by the IOS SLB RADIUS framed-IP sticky database.

sessions [**asn**| **gtp** | **ipmobile** | **radius**	Displays debugging messages for all sessions being handled by IOS SLB. • The optional **asn**keyword enables users to limit the information displayed to only ASN sessions. • The optional **gtp** keyword enables users to limit the information displayed to only GTP sessions. • The optional **ipmobile**keyword enables users to limit the information displayed to only Mobile IP sessions. • The optional **radius** keyword enables users to limit the information displayed to only RADIUS sessions.
sticky gtp imsi	Displays all debugging messages related to the IOS SLB GTP International Mobile Subscriber ID (IMSI) sticky database.
vservers	Displays debugging messages for all virtual servers defined to IOS SLB.

Command Modes

User EXEC or privileged EXEC (#)

Command History

Release	Modification
12.0(7)XE	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.
12.2	This command was integrated into Cisco IOS Release 12.2.
12.1(2)E	The **natpool** and **replication** keywords were added.
12.1(3a)E	The **firewallfarm** keyword was added.
12.1(7)E	The **vservers** keyword was added.
12.1(9)E	The **sessions** keyword was added.
12.1(11b)E	The **route** keyword, the *acl-number* argument, and the **radius** option on the **sessions** keyword were added.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.1(13)E3	The **gtp** keyword and the **gtp** option on the **sessions** keyword were added.
12.2(14)ZA2	The **ipmobile** keyword was added.
12.2(18)SXE	The **sticky gtp imsi**keywords were added.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.2(33)SRC	The **kal-ap**keyword was added.
12.2(33)SRC1	The **asn**keyword and the **asn**option on the **sessions** keyword were added.

Release	Modification
12.2(33)SRE	The **msid**option on the **asn** keyword was added.

Usage Guidelines

This command displays debugging messages for IOS SLB.

See the following caution before using debug commands:

Caution Because debugging output is assigned high priority in the CPU process, it can render the system unusable. For this reason, use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, it is best to use debug commands during periods of lower network flows and fewer users. Debugging during these periods reduces the effect these commands have on other users on the system.

Examples

The following example configures a debugging session to check all IP IOS SLB parameters:

```
Router# debug ip slb all
SLB All debugging is on
Router#
```

The following example stops all debugging:

```
Router# no debug all
All possible debugging has been turned off
Router#
```

The following example configures debugging to check IP IOS SLB replication used with stateful backup and displays the output from the send or transmit virtual server:

```
Router# debug ip slb replication
*Mar  2 08:02:38.019:  SLB Replicate: (send) update vs: VS1 update_count 42
```

The following example shows Cisco IOS SLB DFP debug output:

```
Router# debug ip slb dfp
SLB DFP debugging is on
router#
022048 SLB DFP Queue to main queue - type 2 for Agent 161.44.2.3458229
022048 SLB DFP              select_rc = -1   readset = 0
022048 SLB DFP       Sleeping...
022049 SLB DFP              readset = 0
022049 SLB DFP              select_rc = -1   readset = 0
022049 SLB DFP Processing Q event for Agent 161.44.2.3458229 - OPEN
022049 SLB DFP Queue to conn_proc_q - type 2 for Agent 161.44.2.3458229
022049 SLB DFP              readset = 0
022049 SLB DFP Set SLB_DFP_SIDE_QUEUE
022049 SLB DFP Processing Conn Q event for Agent 161.44.2.3458229 - OPEN
022049 SLB DFP Open to Agent 161.44.2.3458229 succeeded, socket = 0
022049 SLB DFP Agent 161.44.2.3458229 start connect
022049 SLB DFP Connect to Agent 161.44.2.3458229 successful - socket 0
022049 SLB DFP Queue to main queue - type 6 for Agent 161.44.2.3458229
022049 SLB DFP Processing Conn Q unknown MAJOR 80
022049 SLB DFP Reset SLB_DFP_SIDE_QUEUE
022049 SLB DFP              select_rc = -1   readset = 0
022049 SLB DFP       Sleeping...
```

```
022050 SLB DFP              readset = 1
022050 SLB DFP              select_rc = 1   readset = 1
022050 SLB DFP Agent 161.44.2.3458229 fd = 0 readset = 1
022050 SLB DFP Message length 44 from Agent 161.44.2.3458229
022050 SLB DFP Agent 161.44.2.3458229 setting Host 17.17.17.17, Bind ID 1 Weight 1
022050 SLB DFP Agent 161.44.2.3458229 setting Host 34.34.34.34, Bind ID 2 Weight 2
022050 SLB DFP Agent 161.44.2.3458229 setting Host 51.51.51.51, Bind ID 3 Weight 3
022050 SLB DFP Processing Q event for Agent 161.44.2.3458229 - WAKEUP
022050 SLB DFP              readset = 1
022050 SLB DFP              select_rc = 1   readset = 1
022050 SLB DFP Agent 161.44.2.3458229 fd = 0 readset = 1
022050 SLB DFP Message length 64 from Agent 161.44.2.3458229
022050 SLB DFP Agent 161.44.2.3458229 setting Host 17.17.17.17, Bind ID 1 Weight 1
022050 SLB DFP Agent 161.44.2.3458229 setting Host 68.68.68.68, Bind ID 4 Weight 4
022050 SLB DFP Agent 161.44.2.3458229 setting Host 85.85.85.85, Bind ID 5 Weight 5
022050 SLB DFP Agent 161.44.2.3458229 setting Host 17.17.17.17, Bind ID 111 Weight 111
022050 SLB DFP              readset = 1
022115 SLB DFP Queue to main queue - type 5 for Agent 161.44.2.3458229
022115 SLB DFP              select_rc = -1   readset = 0
022115 SLB DFP       Sleeping...
022116 SLB DFP              readset = 1
022116 SLB DFP              select_rc = -1   readset = 0
022116 SLB DFP Processing Q event for Agent 161.44.2.3458229 - DELETE
022116 SLB DFP Queue to conn_proc_q - type 5 for Agent 161.44.2.3458229
022116 SLB DFP              readset = 1
022116 SLB DFP Set SLB_DFP_SIDE_QUEUE
022116 SLB DFP Processing Conn Q event for Agent 161.44.2.3458229 - DELETE
022116 SLB DFP Connection to Agent 161.44.2.3458229 closed
022116 SLB DFP Agent 161.44.2.3458229 deleted
022116 SLB DFP Processing Conn Q unknown MAJOR 80
022116 SLB DFP Reset SLB_DFP_SIDE_QUEUE
022116 SLB DFP Set SLB_DFP_SIDE_QUEUE
022116 SLB DFP Reset SLB_DFP_SIDE_QUEUE
```

debug ip snat

To display information about IP packets translated by the IP stateful network address translation (SNAT) feature, use the **debug ip snat** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip snat [**detailed**]
no debug ip snat [**detailed**]

Syntax Description

detailed	(Optional) Displays debug information in a detailed format.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T	This command was introduced.

Usage Guidelines

The **SNAT** feature allows two or more network address translators to function as a translation group. One member of the translation group handles traffic requiring translation of IP address information. It informs the backup translator of active flows as they occur. The backup translator can then use information from the active translator to prepare duplicate translation table entries enabling the backup translator to become the active translator in the event of a critical failure. Traffic continues to flow without interruption because the same network address translations are used and the state of those translations has been previously defined.

Caution Because the **debug ip snat** command generates a significant amount of output, use it only when traffic on the IP network is low, so other activity on the system is not adversely affected.

Examples

The following is sample output from the **debug ip snat** command:

```
Router# debug ip snat detailed
2w6d:SNAT:Establish TCP peers for PRIMARY
2w6d:SNAT (Send):Enqueuing SYNC Message for Router-Id 100
2w6d:SNAT(write2net):192.168.123.2 <---> 192.168.123.3  send message
2w6d:SNAT(write2net):ver 2, id 100, opcode 1, len 68
2w6d:SNAT (Send):Enqueuing DUMP-REQUEST Message for Router-Id 100
2w6d:SNAT(write2net):192.168.123.2 <---> 192.168.123.3  send message
2w6d:SNAT(write2net):ver 2, id 100, opcode 6, len 68
2w6d:SNAT (readfromnet):Enqueuing SYNC Message msg to readQ
2w6d:SNAT (Receive):Processed SYNC Message from Router-Id:0 for Router-Id:200's entry/entries
2w6d:SNAT (readfromnet):Enqueuing DUMP-REQUEST Message msg to readQ
try/entries
2w6d:SNAT(sense):Send SYNC message
2w6d:SNAT (Send):Enqueuing SYNC Message for Router-Id 100
2w6d:SNAT(write2net):192.168.123.2 <---> 192.168.123.3  send message
2w6d:SNAT(write2net):ver 2, id 100, opcode 1, len 68
2w6d:SNAT (readfromnet):Enqueuing SYNC Message msg to readQ
2w6d:SNAT (Receive):Processed SYNC Message from Router-Id:200 for Router-Id:200's
entry/entries
```

The table below describes the significant fields shown in the display.

Table 17: debug ip snat Field Descriptions

Field	Description
SNAT:	Indicates that the packet is being translated by the SNAT feature.
DUMP-REQUEST Message	Requests for entries after the SNAT router is active.

debug ip socket

To display all state change information for all sockets, use the **debug ip socket** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip socket
no debug ip socket

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Use this command to collect information on the socket interface. To get more complete information on a socket/TCP port pair, use this command in conjunction with the **debug ip tcp transactions** command.

Because the socket debugging information is state-change oriented, you will not see the debugging message on a per-packet basis. However, if the connections normally have very short lives (few packet exchanges during the life cycle of a connection), then socket debugging could become expensive because of the state changes involved during connection setup and teardown.

Examples

The following is sample output from the **debug ip socket** output from a server process:

```
Router# debug ip socket
Added socket 0x60B86228 to process 40
SOCKET: set TCP property TCP_PID, socket 0x60B86228, TCB 0x60B85E38
Accepted new socket fd 1, TCB 0x60B85E38
Added socket 0x60B86798 to process 40
SOCKET: set TCP property TCP_PID, socket 0x60B86798, TCB 0x60B877C0
SOCKET: set TCP property TCP_BIT_NOTIFY, socket 0x60B86798, TCB 0x60B877C0
SOCKET: created new socket to TCP, fd 2, TCB 0x60B877C0
SOCKET: bound socket fd 2 to TCB 0x60B877C0
SOCKET: set TCP property TCP_WINDOW_SIZE, socket 0x60B86798, TCB 0x60B877C0
SOCKET: listen on socket fd 2, TCB 0x60B877C0
SOCKET: closing socket 0x60B86228, TCB 0x60B85E38
SOCKET: socket event process: socket 0x60B86228, TCB new state --> FINWAIT1
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING
SOCKET: Removed socket 0x60B86228 from process 40 socket list
```

The following is sample output from the **debug ip socket** command from a client process:

```
Router# debug ip socket
Added socket 0x60B70220 to process 2
SOCKET: set TCP property TCP_PID, socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: set TCP property TCP_BIT_NOTIFY, socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: created new socket to TCP, fd 0, TCB 0x60B6CFDC
SOCKET: socket event process: socket 0x60B70220, TCB new state --> SYNSENT
socket state: SS_ISCONNECTING
SOCKET: socket event process: socket 0x60B70220, TCB new state --> ESTAB
socket state: SS_ISCONNECTING
SOCKET: closing socket 0x60B70220, TCB 0x60B6CFDC
SOCKET: socket event process: socket 0x60B70220, TCB new state --> FINWAIT1
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING
SOCKET: Removed socket 0x60B70220 from process 2 socket list
```

The table below describes the significant fields shown in the display.

Table 18: debug ip socket Field Descriptions

Field	Description
Added socket 0x60B86228 process 40	New socket is opened for process 40.
SOCKET	Indicates that this is a SOCKET transaction.
set TCP property TCP_PID	Sets the process ID to the TCP associated with the socket.

Field	Description
socket 0x60B86228, TCB 0x60B85E38	Address for the socket/TCP pair.
set TCP property TCP_BIT_NOTIFY	Sets the method for how the socket wants to be notified for an event.
created new socket to TCP, fd 2	Opened a new socket referenced by file descriptor 2 to TCP.
bound socket fd 2 to TCB	Bound the socket referenced by file descriptor 2 to TCP.
listen on socket fd 2	Indicates which file descriptor the application is listening to.
closing socket	Indicates that the socket is being closed.
socket event process	Processed a state change event occurred in the transport layer.
TCB new state --> FINWAIT1	TCP state machine changed to FINWAIT1. (See the **debug ip tcp transaction** command for more information on TCP state machines.)
socket state: SS_ISCONNECTED SS_CANTSENDMORE SS_ISDISCONNECTING	New SOCKET state flags after the transport event processing. This socket is still connected, but disconnecting is in progress, and it will not send more data to peer. Possible SOCKET state flags follow: • SS_NOFDREF No file descriptor reference for this socket. • SS_ISCONNECTING Socket connecting is in progress. • SS_ISBOUND Socket is bound to TCP. • SS_ISCONNECTED Socket is connected to peer. • SS_ISDISCONNECTING Socket disconnecting is in progress. • SS_CANTSENDMORE Can’t send more data to peer. • SS_CANTRCVMORE Can’t receive more data from peer. • SS_ISDISCONNECTED Socket is disconnected. Connection is fully closed.

Field	Description
Removed socket 0x60B86228 from process 40 socket list	Connection is closed, and the socket is removed from the process socket list.

Related Commands

Command	Description
debug ip tcp transactions	Displays information on significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip ssh

To display debugging messages for Secure Shell (SSH), use the **debug ip ssh** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip ssh [{**detail** | **packet**}]
no debug ip ssh

Syntax Description

detail	(Optional) Specifies SSH protocol, channel requests and information state changes.
packet	(Optional) Specifies information regarding the SSH packet.

Command Default

Debugging for SSH is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(5)S	This command was introduced.
12.1(1)T	This command was integrated into Cisco IOS Release 12.1T.
12.4(20)T	The detail and packet keywords were added.
Cisco IOS XE Release 2.4	This command was implemented on the Cisco ASR 1000 series routers.

Usage Guidelines

Use the **debug ip ssh** command to ensure normal operation of the SSH server.

Examples

The following example shows the SSH debugging output:

```
Router# debug ip ssh
00:53:46: SSH0: starting SSH control process
00:53:46: SSH0: Exchanging versions - SSH-1.5-Cisco-1.25
00:53:46: SSH0: client version is - SSH-1.5-1.2.25
00:53:46: SSH0: SSH_SMSG_PUBLIC_KEY message sent
00:53:46: SSH0: SSH_CMSG_SESSION_KEY message received
00:53:47: SSH0: keys exchanged and encryption on
00:53:47: SSH0: authentication request for userid guest
```

```
00:53:47: SSH0: authentication successful for jcisco
00:53:47: SSH0: starting exec shell
```

The following example shows the SSH detail output:

```
Router# debug ip ssh detail
00:04:22: SSH0: starting SSH control process
00:04:22: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
00:04:22: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
00:04:22: SSH2 0: SSH2_MSG_KEXINIT sent
00:04:22: SSH2 0: SSH2_MSG_KEXINIT received
00:04:22: SSH2:kex: client->server enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2:kex: server->client enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally
```

The following example shows the SSH packet output:

```
Router# debug ip ssh packet
00:05:43: SSH2 0: send:packet of  length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 144 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: signature length 143
00:05:43: SSH2 0: send:packet of  length 448 (length also includes padlen of 7)
00:05:43: SSH2 0: send:packet of  length 16 (length also includes padlen of 10)
00:05:43: SSH2 0: newkeys: mode 1
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: total packet length of 16 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 8 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 10 bytes
00:05:43: SSH2 0: newkeys: mode 0
00:05:43: SSH2 0: ssh_receive: 52 bytes received
00:05:43: SSH2 0: input: total packet length of 32 bytes
```

```
00:05:43: SSH2 0: partial packet length(block size)16 bytes,needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3 :ok
```

debug ip subscriber

To enable Intelligent Services Gateway (ISG) IP subscriber session debugging, use the **debug ip subscriber** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip subscriber {**all** | **error** | **event** | **fsm** | **packet**}
no debug ip subscriber {**all** | **error** | **event** | **fsm** | **packet**}

Syntax Description

all	Displays all debugging messages related to IP subscriber sessions.
error	Displays debugging messages about IP subscriber session errors.
event	Displays debugging messages about IP subscriber session events.
fsm	Displays debugging messages related to session state changes for IP subscriber sessions.
packet	Displays debugging messages related to IP subscriber session packets.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(31)SB2	This command was introduced.
12.2(33)SRC	This command was integrated into Cisco IOS Release 12.2(33)SRC.
Cisco IOS XE Release 2.2	This command was integrated into Cisco IOS XE Release 2.2.

Examples

The following example show sample output for the **debug ip subscriber** command:

```
Router# debug ip subscriber packet
Packet debugs:
1d07h: IPSUB_DP: [Et0/0:I:CEF:0000.0000.0002] Rx driver forwarded packet via les, return
code = 0
1d07h: IPSUB_DP: [Et0/0:I:PROC:0000.0000.0002] Packet classified, results = 0x18
1d07h: IPSUB_DP: [ms1:I:PROC:0000.0000.0002] Rx driver forwarded the packet
1d07h: IPSUB_DP: [ms1:I:PROC:0000.0000.0002] Packet classified, results = 0x42
1d07h: IPSUB_DP: [ms1:O:PROC:RED:50.0.0.3] Packet classified, results = 0x14
Router#
1d07h: IPSUB_DP: [ms1:O:PROC:RED:50.0.0.3] Subscriber features executed, return code = 0
1d07h: IPSUB_DP: [ms1:O:PROC:RED:50.0.0.3] Tx driver forwarding the packet
1d07h: IPSUB_DP: [Et0/0:O:PROC:RED:50.0.0.3] Packet classified, results = 0x14
```

Related Commands

Command	Description
show ip subscriber	Displays information about ISG IP subscriber sessions.

debug ip subscriber redundancy

To enable Intelligent Service Gateway (ISG) IP subscriber session debugging on a Cisco 7600 router, use the **debug ip subscriber** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug ip subscriber redundancy
no debug ip subscriber redundancy

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SRC	This command was introduced.

Examples

The following example shows that the **debug ip subscriber redundancy**command is turned on:

```
Router# debug ip subscriber redundancy
 IP subscriber redundancy debugging is on.
```

Related Commands

Command	Description
clear ip subscriber interface	Disconnects and removes all ISG IP subscriber sessions associated with a specific interface on a Cisco 7600 router.
clear ip subscriber slot	Disconnects and removes all ISG IP subscriber sessions associated with a specific hardware slot on a Cisco 7600 router.
show ip subscriber interface	Displays information about an ISG IP subscriber interface on a Cisco 7600 router.
show ip subscriber redundancy	Displays information about ISG IP subscriber sessions on a Cisco 7600 router.
show debugging	Displays information about the types of debugging that are enabled for your router.

debug ip tcp congestion

To display information about TCP congestion events, use the **debug ip tcp congestion** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip tcp congestion
no debug ip tcp congestion

Syntax Description

This command has no arguments or keywords.

Command Default

Information from the New Reno congestion control algorithm is displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(2)T	This command was introduced.

Usage Guidelines

The **debug ip tcp congestion** command can be used to debug a performance problem on a TCP/IP network that you have isolated above the data-link layer. It also displays information related to variation in TCP's send window, congestion window, and congestion threshold window.

Examples

The following is sample output from the **debug ip tcp congestion** command:

```
Router# debug ip tcp congestion

*May 20 22:49:49.091: Setting New Reno as congestion control algorithm
*May 22 05:21:47.281: Advance cwnd by 12
*May 22 05:21:47.281: TCP85FD0C10: sndcwnd: 1472
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1475
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1478
*May 22 05:21:47.285: Advance cwnd by 9
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1487
.
.
.
*May 20 22:50:32.559: [New Reno] sndcwnd: 8388480 ssthresh: 65535 snd_mark: 232322
*May 20 22:50:32.559: 10.168.10.10:42416 <---> 10.168.30.11:49100 congestion window changes
*May 20 22:50:32.559: cwnd from 8388480 to 2514841, ssthresh from 65535 to 2514841
```

For IOS TCP, New Reno is the default congestion control algorithm. However, an application can also use Binary Increase Congestion Control (BIC) as the congestion algorithm. The following is sample output from the **debug ip tcp congestion** command using the BIC congestion algorithm:

```
Router# debug ip tcp congestion

*May 22 05:21:42.281: Setting BIC as congestion control algorithm
*May 22 05:21:47.281: Advance cwnd by 12
*May 22 05:21:47.281: TCP85FD0C10: sndcwnd: 1472
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1475
*May 22 05:21:47.285: Advance cwnd by 3
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1478
*May 22 05:21:47.285: Advance cwnd by 9
*May 22 05:21:47.285: TCP85FD0C10: sndcwnd: 1487
.
.
.
.
.
.
*May 20 22:50:32.559: [BIC] sndcwnd: 8388480 ssthresh: 65535 bic_last_max_cwnd: 0 last_cwnd:
 8388480
*May 20 22:50:32.559: 10.168.10.10:42416 <---> 10.168.30.11:49100 congestion window changes
```

```
*May 20 22:50:32.559: cwnd from 8388480 to 2514841, ssthresh from 65535 to 2514841
*May 20 22:50:32.559: bic_last_max_cwnd changes from 0 to 8388480
```

The table below describes the significant fields shown in the display.

Table 19: debug ip tcp congestion Field Descriptions

Field	Description
Setting New Reno as congestion control algorithm	TCP is using New Reno as the congestion control algorithm.
TCP85FD0C10	TCP's control block identifier.
Advance cwnd	Increase in TCP’s congestion window.
sndcwnd	TCP's send congestion window.
[New Reno]	Values reflected are those of TCP’s New Reno congestion control.
ssthresh:	TCP’s slow start threshold.
snd_mark	New value of one of New Reno’s parameters.
10.168.10.10:42416:	Local address and port number for the TCP connection.
10.168.30.11.49100:	Foreign address and port number for the TCP connection.
congestion window changes	Change in TCP’s send congestion window.

Related Commands

Command	Description
ip tcp window-size	Alters the TCP window size.

debug ip tcp driver

To display information on TCP driver events; for example, connections opening or closing, or packets being dropped because of full queues, use the **debug ip tcp driver** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip tcp driver
no debug ip tcp driver

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

The TCP driver is the process that the router software uses to send packet data over a TCP connection. Remote source-route bridging (RSRB), serial tunneling (STUN), and X.25 switching currently use the TCP driver.

Using the **debug ip tcp driver** command together with the **debug ip tcp driver-pak**command provides the most verbose debugging output concerning TCP driver activity.

Examples

The following is sample output from the **debug ip tcp driver** command:

```
Router# debug ip tcp driver
TCPDRV359CD8: Active open 172.21.80.26:0 --> 172.21.80.25:1996 OK, lport 36628
TCPDRV359CD8: enable tcp timeouts
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 Abort
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 DoClose tcp abort
```

The table below describes the significant fields shown in the display.

Table 20: debug ip tcp driver Field Descriptions

Field	Description
TCPDRV359CD8:	Unique identifier for this instance of TCP driver activity.
Active open 172.21.80.26	Indication that the router at IP address 172.21.80.26 has initiated a connection to another router.
:0	TCP port number the initiator of the connection uses to indicate that any port number can be used to set up a connection.
--> 172.21.80.25	IP address of the remote router to which the connection has been initiated.
:1996	TCP port number that the initiator of the connection is requesting that the remote router use for the connection. (1996 is a private TCP port number reserved in this implementation for RSRB.)
OK,	Indication that the connection has been established. If the connection has not been established, this field and the following field do not appear in this line of output.
lport 36628	TCP port number that has actually been assigned for the initiator to use for this connection.

The following line indicates that the TCP driver user (RSRB, in this case) will allow TCP to drop the connection if excessive retransmissions occur:

```
TCPDRV359CD8: enable tcp timeouts
```

The following line indicates that the TCP driver user (in this case, RSRB) at IP address 172.21.80.26 (and using TCP port number 36628) is requesting that the connection to IP address 172.21.80.25 using TCP port number 1996 be aborted:

```
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 Abort
```

The following line indicates that this connection was in fact closed because of an abnormal termination:

```
TCPDRV359CD8: 172.21.80.26:36628 --> 172.21.80.25:1996 DoClose tcp abort
```

debug ip tcp driver-pak

To display information on every operation that the TCP driver performs, use the **debug ip tcp driver-pak** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip tcp driver-pak
no debug ip tcp driver-pak

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

This command turns on a verbose debugging by logging at least one debugging message for every packet sent or received on the TCP driver connection.

The TCP driver is the process that the router software uses to send packet data over a TCP connection. Remote source-rate bridging (RSRB), serial tunneling (STUN), and X.25 switching currently use the TCP driver.

To observe the context within which certain **debug ip tcp driver-pak** messages occur, turn on this command in conjunction with the **debug ip tcp driver** command.

Caution Because the **debug ip tcp driver-pak** command generates so many messages, use it only on lightly loaded systems. This command not only places a substantial load on the system processor, it also may change the symptoms of any unexpected behavior that occurs.

Examples

The following is sample output from the **debug ip tcp driver-pak** command:

```
Router# debug ip tcp driver-pak
TCPDRV359CD8: send 2E8CD8 (len 26) queued
TCPDRV359CD8: output pak 2E8CD8 (len 26) (26)
TCPDRV359CD8: readf 42 bytes (Thresh 16)
TCPDRV359CD8: readf 26 bytes (Thresh 16)
TCPDRV359CD8: readf 10 bytes (Thresh 10)
TCPDRV359CD8: send 327E40 (len 4502) queued
TCPDRV359CD8: output pak 327E40 (len 4502) (4502)
```

The table below describes the significant fields shown in the display.

Table 21: debug ip tcp driver-pak Field Descriptions

Field	Description
TCPDRV359CD8	Unique identifier for this instance of TCP driver activity.
send	Indicates that this event involves the TCP driver sending data.
2E8CD8	Address in memory of the data the TCP driver is sending.
(len 26)	Length of the data (in bytes).

Field	Description
queued	Indicates that the TCP driver user process (in this case, RSRB) has transferred the data to the TCP driver to send.

The following line indicates that the TCP driver has sent the data that it had received from the TCP driver user, as shown in the previous line of output. The last field in the line (26) indicates that the 26 bytes of data were sent out as a single unit.

```
TCPDRV359CD8: output pak 2E8CD8 (len 26) (26)
```

The following line indicates that the TCP driver has received 42 bytes of data from the remote IP address. The TCP driver user (in this case, remote source-route bridging) has established an input threshold of 16 bytes for this connection. (The input threshold instructs the TCP driver to transfer data to the TCP driver user only when at least 16 bytes are present.)

```
TCPDRV359CD8: readf 42 bytes (Thresh 16)
```

debug ip tcp ecn

To turn on debugging of the TCP Explicit Congestion Notification (ECN) capability, use the **debug ip tcp ecn**command in privileged EXEC mode. To turn off the debugging, use the **no** form of this command.

debug ip tcp ecn
no debug ip tcp ecn

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(7)T	This command was introduced.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Examples

The following example shows the messages that verify that the end hosts are connected and configured for ECN:

```
Router# debug ip tcp ecn
!
TCP ECN debugging is on
!
Router# telnet 10.1.25.31

Trying 10.1.25.31 ...
!
01:43:19: 10.1.25.35:11000 <---> 10.1.25.31:23   out ECN-setup SYN
```

```
01:43:21: 10.1.25.35:11000 <---> 10.1.25.31:23   congestion window changes
01:43:21: cwnd from 1460 to 1460, ssthresh from 65535 to 2920
01:43:21: 10.1.25.35:11000 <---> 10.1.25.31:23   in non-ECN-setup SYN-ACK
```

Before a TCP connection can use ECN, a host sends an ECN-setup SYN (synchronization) packet to a remote end that contains an ECE and CWR bit set in the header. This indicates to the remote end that the sending TCP is ECN-capable, rather than an indication of congestion. The remote end sends an ECN-setup SYN-ACK (acknowledgment) packet to the sending host.

In the example above, the "out ECN-setup SYN" text means that a SYN packet with the ECE and CWR bit set was sent to the remote end. The "in non-ECN-setup SYN-ACK" text means that the remote end did not favorably acknowledge the ECN request and that therefore the session is ECN capable.

The following debug output shows that ECN capabilities are enabled at both ends. In response to the ECN-setup SYN, the other end favorably replied with an ECN-setup SYN-ACK message. This connection is now ECN capable for the rest of the session.

```
Router# telnet 10.10.10.10

Trying 10.10.10.10 ... Open
Password required, but none set
!
1d20h: 10.1.25.34:11003 <---> 10.1.25.35:23   out ECN-setup SYN
1d20h: 10.1.25.34:11003 <---> 10.1.25.35:23   in ECN-setup SYN-ACK
```

Use the **show tcp tcb** command to display the end-host connections.

Related Commands

Command	Description
ip tcp ecn	Enables TCP ECN.
show tcp tcb	Displays the status of local and remote end hosts.

debug ip tcp ha

To display TCP high availability (HA) events or debugging information for TCP stack interactions between the active Route Processor (RP) and the standby RP, use the **debug ip tcp ha** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip tcp ha {**events** | **transactions**} [**detail**]
no debug ip tcp ha {**events** | **transactions**} [**detail**]

Syntax Description

events	Displays TCP HA failures.
transactions	Displays failed TCP stack interactions between the active RP and standby RP.
detail	(Optional) Displays detailed debugging information about successful TCP HA operations and useful informational messages or about successful TCP stack interactions between the active and standby RP.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(28)SB	This command was introduced.
15.0(1)S	This command was integrated into Cisco IOS Release 15.0(1)S.
Cisco IOS XE 3.1S	This command was integrated into Cisco IOS XE Release 3.1S.

Usage Guidelines

The **debug ip tcp ha** command is used to display TCP stateful switchover (SSO) events or debugging information for TCP stack interactions between the active RP and the standby RP. This is command is useful for troubleshooting SSO-aware TCP connections.

Use the **debug ip tcp ha** command with the **transactions** keyword to display failed TCP stack interactions between the active RP and standby RP. This form of the command displays failed TCP HA messages, RF redundancy-related client-application transactions, IPC client-application transactions, and In-Service Software Upgrade (ISSU) transactions.

Use the **debug ip tcp ha** command with the **transactions** and **detail** keywords to display successful TCP stack interactions between the active and standby RP. This form of the command displays successful TCP HA messages, RF redundancy-related client-application transactions, IPC client-application transactions, and ISSU transactions.

Use the **debug ip tcp ha** command with the **events** keyword to display TCP HA failures. This form of the command displays TCP HA failed encode or decode messages, system resources failures (such as memory allocation failures in the context of TCP HA), failed state changes, and failures that occur when SSO is enabled or disabled.

Use the **debug ip tcp ha** command with the **events** and **detail** keywords to display successful TCP HA operations and useful informational messages. This form of the command displays successful TCP encode or decode messages, state changes, and operations that occur when SSO is enabled or disabled.

Examples

The following is sample output from the **debug ip tcp ha** command with the **transactions** and **detail** keywords. The following output shows packet flow from the active to the standby RP for an established TCP SSO connection:

```
*Feb 19 23:28:23.324: TCPHA: Sending pkt msg, conn_id = 39, seq no = 2727115707
*Feb 19 23:28:23.324: TCPHA: Sending pkt msg, conn_id = 396, seq no = 2959469308
*Feb 19 23:28:23.324: TCPHA: Sending pkt msg, conn_id = 41, seq no = 1270243395
*Feb 19 23:28:23.932: TCPHA: Sending pkt msg, conn_id = 42, seq no = 974255741
*Feb 19 23:28:23.932: TCPHA: Sending pkt msg, conn_id = 475, seq no = 3059612402
*Feb 19 23:28:24.544: TCPHA: Sending dummy pkt to standby; cid=109, size=19

*Feb 19 23:28:42.976: TCPHA: Recd IPC msg len 24, type 3
*Feb 19 23:28:42.976: TCPHA: Recd IPC msg len 24, type 3
*Feb 19 23:28:43.172: TCPHA: Recd IPC msg len 79, type 2
*Feb 19 23:28:43.172: TCPHA: Recd IPC msg len 79, type
```

debug ip tcp intercept

To display TCP intercept statistics, use the **debug ip tcp intercept**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip tcp intercept

no debug ip tcp intercept

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is sample output from the **debug ip tcp intercept** command:

```
Router# debug ip tcp intercept
```

A connection attempt arrives:

```
INTERCEPT: new connection (172.19.160.17:61774) => (10.1.1.30:23)
INTERCEPT: 172.19.160.17:61774 <- ACK+SYN (10.1.1.30:61774)
```

A second connection attempt arrives:

```
INTERCEPT: new connection (172.19.160.17:62030) => (10.1.1.30:23)
INTERCEPT: 172.19.160.17:62030 <- ACK+SYN (10.1.1.30:62030)
```

The router resends to both apparent clients:

```
INTERCEPT: retransmit 2 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 2 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
```

A third connection attempt arrives:

```
INTERCEPT: new connection (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: 171.69.232.23:1048 <- ACK+SYN (10.1.1.30:1048)
```

The router sends more retransmissions trying to establish connections with the apparent clients:

```
INTERCEPT: retransmit 4 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 4 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 2 (171.69.232.23:1048) <- (10.1.1.30:23) SYNRCVD
```

The router establishes the connection with the third client and resends to the server:

```
INTERCEPT: 1st half of connection is established (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: (171.69.232.23:1048) SYN -> 10.1.1.30:23
INTERCEPT: retransmit 2 (171.69.232.23:1048) -> (10.1.1.30:23) SYNSENT
```

The server responds; the connection is established:

```
INTERCEPT: 2nd half of connection established (171.69.232.23:1048) => (10.1.1.30:23)
INTERCEPT: (171.69.232.23:1048) ACK -> 10.1.1.30:23
```

The router resends to the first two apparent clients, times out, and sends resets:

```
INTERCEPT: retransmit 8 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 8 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 16 (172.19.160.17:61774) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmit 16 (172.19.160.17:62030) <- (10.1.1.30:23) SYNRCVD
INTERCEPT: retransmitting too long (172.19.160.17:61774) => (10.1.1.30:23) SYNRCVD
INTERCEPT: 172.19.160.17:61774 <- RST (10.1.1.30:23)
```

```
INTERCEPT: retransmitting too long (172.19.160.17:62030) => (10.1.1.30:23) SYNRCVD
INTERCEPT: 172.19.160.17:62030 <- RST (10.1.1.30:23)
```

debug ip tcp packet

To enable debug messages for received and sent TCP packets, use the **debug ip tcp packet** command in privileged EXEC mode. To disable TCP packet debug messages, use the **no** form of this command.

debug ip tcp packet [{**line-number** | **address** **ip-address** | {**aux** | **console** | **tty** | **vty**} **line-number** | **in** | **out** | **port** **port-number** | **slot/port** | **slot/subslot/port**}]
no debug ip tcp packet [{**line-number** | **address** **ip-address** | {**aux** | **console** | **tty** | **vty**} **line-number** | **in** | **out** | **port** **port-number** | **slot/port** | **slot/subslot/port**}]

Syntax Description

line-number	(Optional) Line number. Valid range is 0 to 710.
address *ip-address*	(Optional) Specifies the source or destination IP address.
aux *line-number*	(Optional) Specifies the auxiliary line.
console *line-number*	(Optional) Specifies the primary terminal line.
in	(Optional) Specifies the incoming segments.
out	(Optional) Specifies the outgoing segments.
port *port-number*	(Optional) Specifies the source or destination port number.
tty *line-number*	(Optional) Specifies the terminal controller.
vty *line-number*	(Optional) Specifies the virtual terminal.
slot / *port*	(Optional) Specifies the slot and port for modems. The slash mark is required.
slot / *subslot* / *port*	(Optional) Specifies the slot, subslot, and port for modems. The slash mark is required.

Command Default

If no optional arguments or keywords are entered, this command displays all TCP packet debug messages.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
11.1	This command was introduced.

Examples

The following is sample output from the **debug ip tcp packet**command:

```
Router# debug ip tcp packet
tcp0: I LISTEN 172.16.0.0:49620 172.16.0.1:80 seq 2116160325
OPTS 4 SYN WIN 1024
tcp0: O SYNRCVD 172.16.0.34:49620 172.16.0.1:80 seq 3992162775
```

```
OPTS 4 ACK 2116160325 SYN WIN 4128
tcp0: I SYNRCVD 172.16.0.34:49620 172.16.0.1:80 seq 2116160326
RST WIN 0
```

Related Commands

Command	Description
debug ip packet detail	Displays general IP debugging information and IP security option security transactions.
debug ip tcp driver	Displays information on TCP driver events; for example, connections opening or closing, or packets being dropped because of full queues.
debug ip tcp transactions	Displays information on significant TCP transactions such as state changes, retransmissions, and duplicate packets.

debug ip tcp transactions

To display information on significant TCP transactions such as state changes, retransmissions, and duplicate packets, use the **debug ip tcp transactions**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip tcp transactions
no debug ip tcp transactions

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.0	This command was introduced.
12.3(7)T	The command output was enhanced to account for the following conditions: TCP entering Fast Recovery mode, duplicate acknowledgments being received during Fast Recovery mode, and partial acknowledgments being received.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.

Usage Guidelines

This command is particularly useful for debugging a performance problem on a TCP/IP network that you have isolated above the data-link layer.

The **debug ip tcp transactions** command displays output for packets that the router sends and receives, but does not display output for packets that it forwards.

Examples

The following is sample output from the **debug ip tcp transactions**command:

```
Router# debug ip tcp transactions
TCP: sending SYN, seq 168108, ack 88655553
TCP0: Connection to 10.9.0.13:22530, advertising MSS 966
TCP0: state was LISTEN -> SYNRCVD [23 -> 10.9.0.13(22530)]
```

```
TCP0: state was SYNSENT -> SYNRCVD [23 -> 10.9.0.13(22530)]
TCP0: Connection to 10.9.0.13:22530, received MSS 956
TCP0: restart retransmission in 5996
TCP0: state was SYNRCVD -> ESTAB [23 -> 10.9.0.13(22530)]
TCP2: restart retransmission in 10689
TCP2: restart retransmission in 10641
TCP2: restart retransmission in 10633
TCP2: restart retransmission in 13384 -> 10.0.0.13(16151)]
TCP0: restart retransmission in 5996 [23 -> 10.0.0.13(16151)]
```

The following line from the **debug ip tcp transactions** command output shows that TCP has entered Fast Recovery mode:

```
fast re-transmit - sndcwnd - 512, snd_last - 33884268765
```

The following lines from the **debug ip tcp transactions** command output show that a duplicate acknowledgment is received when in Fast Recovery mode (first line) and a partial acknowledgment has been received (second line):

```
TCP0:ignoring second congestion in same window sndcwn - 512, snd_1st - 33884268765
TCP0:partial ACK received sndcwnd:338842495
```

The table below describes the significant fields shown in the display.

Table 22: debug ip tcp transactions Field Descriptions

Field	Description
TCP	Indicates that this is a TCP transaction.
sending SYN	Indicates that a synchronize packet is being sent.
seq 168108	Indicates the sequence number of the data being sent.
ack 88655553	Indicates the sequence number of the data being acknowledged.
TCP0	Indicates the TTY number (0, in this case) with which this TCP connection is associated.
Connection to 10.9.0.13:22530	Indicates the remote address with which a connection has been established.
advertising MSS 966	Indicates the maximum segment size that this side of the TCP connection is offering to the other side.

Field	Description
state was LISTEN -> SYNRCVD	Indicates that the TCP state machine changed state from LISTEN to SYNRCVD. Possible TCP states that can follow are: • CLOSED--Connection closed. • CLOSEWAIT--Received a FIN segment. • CLOSING--Received a FIN/ACK segment. • ESTAB--Connection established. • FINWAIT 1--Sent a FIN segment to start closing the connection. • FINWAIT 2--Waiting for a FIN segment. • LASTACK--Sent a FIN segment in response to a received FIN segment. • LISTEN--Listening for a connection request. • SYNRCVD--Received a SYN segment and responded. • SYNSENT--Sent a SYN segment to start connection negotiation. • TIMEWAIT--Waiting for the network to clear segments for this connection before the network no longer recognizes the connection as valid. This must occur before a new connection can be set up.
[23 -> 10.9.0.13(22530)]	The elements within these brackets are as follows: • The first field (23) indicates the local TCP port. • The second field (10.9.0.13) indicates the destination IP address. • The third field (22530) indicates the destination TCP port.
restart retransmission in 5996	Indicates the number of milliseconds until the next retransmission takes place.
sndcwnd - 512	Indicates the size of the send congestion window.
snd_last - 33884268765	Indicates the size of the last window.

debug ip traffic-export events

To enable debugging messages for exported IP packet events, use the **debug ip traffic-export**command in privileged EXEC mode. To disable debugging messages, use the **no** form of this command.

debug ip traffic-export events
no debug ip traffic-export events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)T	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.

Examples

The following is sample output from the **debug ip traffic-export events**command:

```
Router# debug ip traffic-export events
RITE:exported input packet # 547
RITE:exported input packet # 548
RITE:exported input packet # 549
RITE:exported input packet # 550
RITE:exported input packet # 551
RITE:exported input packet # 552
RITE:exported input packet # 553
RITE:exported input packet # 554
RITE:exported input packet # 555
RITE:exported input packet # 556
RITE:exported input packet # 557
RITE:exported input packet # 558
RITE:exported input packet # 559
RITE:exported input packet # 560
RITE:exported input packet # 561
RITE:exported input packet # 562
```

Related Commands

Command	Description
ip traffic-export profile	Creates or edits an IP traffic export profile and enables the profile on an ingress interface.

debug ip trigger-authentication

To display information related to automated double authentication, use the **debug ip trigger-authentication** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip trigger-authentication [**verbose**]
no debug ip trigger-authentication [**verbose**]

Syntax Description

verbose	(Optional) Specifies that the complete debugging output be displayed, including information about packets that are blocked before authentication is complete.

Command Modes

Privileged EXEC

Usage Guidelines

Use this command when troubleshooting automated double authentication.

This command displays information about the remote host table. Whenever entries are added, updated, or removed, a new debugging message is displayed.

What is the remote host table? Whenever a remote user needs to be user-authenticated in the second stage of automated double authentication, the local device sends a User Datagram Protocol (UDP) packet to the host of the remote user. Whenever such a UDP packet is sent, the host IP address of the user is added to a table. If additional UDP packets are sent to the same remote host, a new table entry is not created; instead, the existing entry is updated with a new time stamp. This remote host table contains a cumulative list of host entries; entries are deleted after a timeout period or after you manually clear the table by using the **clear ip trigger-authentication** command.

If you include the **verbose** keyword, the debugging output also includes information about packet activity.

Examples

The following is sample output from the **debug ip trigger-authentication** command. In this example, the local device at 172.21.127.186 sends a UDP packet to the remote host at 172.21.127.114. The UDP packet is sent to request the remote user's username and password (or PIN). (The output says "New entry added.")

After a timeout period, the local device has not received a valid response from the remote host, so the local device sends another UDP packet. (The output says "Time stamp updated.")

Then the remote user is authenticated, and after a length of time (the timeout period) the entry is removed from the remote host table. (The output says "remove obsolete entry.")

```
myfirewall# debug ip trigger-authentication
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.114, qdata=7C2504
              New entry added, timestamp=2940514234
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.114, qdata=7C2504
              Time stamp updated, timestamp=2940514307
TRIGGER_AUTH: remove obsolete entry, remote host=172.21.127.114
```

The following is sample output from the **debug ip trigger-authentication verbose** command. In this example, messages about packet activity are included because of the use of the **verbose** keyword.

You can see many packets that are being blocked at the interface because the user has not yet been double authenticated. These packets will be permitted through the interface only after the user has been double authenticated. (You can see packets being blocked when the output says "packet enqueued" and then "packet ignored.")

```
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.113, qdata=69FEEC
              Time stamp updated
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: UDP sent from 172.21.127.186 to 172.21.127.113, qdata=69FEEC
              Time stamp updated
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
TRIGGER_AUTH: packet enqueued, qdata=69FEEC
              remote host=172.21.127.113, local host=172.21.127.186 (if: 0.0.0.0)
TRIGGER_AUTH: packet ignored, qdata=69FEEC
```

debug ip trm

To enable debug information of the Trend Registration Module (TRM), use the **debug ip trm** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip trm [{**detailed** | **timers**}]
no debug ip trm [{**detailed** | **timers**}]

Syntax Description

detailed	(Optional) The system prints detailed information about the TRM. If not specified, the system displays basic status information.
timers	(Optional) The system prints information about timer events on the TRM. If not specified, the system displays basic status information.

Command Default

This command is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(15)XZ	This command was introduced.

Usage Guidelines

Use the **debug ip trm** to enable debug information of the TRM, which handles the registration between the system and the Trend Router Provisioning Server (TRPS).

Examples

The following is sample output from the **debug ip trm**command:

```
Router# debug ip trm
TRM: Exceeded retry timeouts. Setting server inactive
```

The following is sample output from the **debug ip trm detailed** command:

```
Router# debug ip trm detailed
TRM: Sending Reg Req to TRPS. Requesting AV Key = No
Modify Trend Global Parameter map
```

The following is sample output from the **debug ip trm timers** command:

```
Router# debug ip trm timers
TRM: Wait timer for active server. Sent Reg request
```

debug ip urd

To display debugging messages for URL Rendezvous Directory (URD) channel subscription report processing, use the **debug ip urd command in privileged EXEC**mode. To disable debugging output, use the **no** form of this command.

debug ip urd [{*hostnameip-address*}]
no debug ip urd

Syntax Description

hostname	(Optional) The domain Name System (DNS) name.
ip-address	(Optional) The IP address.

Command Default

If no host name or IP address is specified, all URD reports are debugged.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(3)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following is sample output from the **debug ip urd** command:

```
Router# debug ip urd
13:36:25 pdt:URD:Data intercepted from 171.71.225.103
13:36:25 pdt:URD:Enqueued string:
'/cgi-bin/error.pl?group=232.16.16.16&port=32620&source=171.69.214.1&li'
13:36:25 pdt:URD:Matched token:group
13:36:25 pdt:URD:Parsed value:232.16.16.16
13:36:25 pdt:URD:Creating IGMP source state for group 232.16.16.16
```

debug ip urlfilter

To enable debug information of URL filter subsystems, use the **debug ip urlfilter** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip urlfilter {**function-trace** | **detailed** | **events**}
no debug ip urlfilter {**function-trace** | **detailed** | **events**}

Syntax Description

function-trace	The system displays a sequence of important functions that are called when configuring URL filtering.
detailed	The system displays detailed information about various activities that occur during URL filtering.
events	The system displays various events such as queue event, timer event, and socket event.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(11)YU	This command was introduced.
12.2(15)T	This command was integrated into Cisco IOS Release 12.2(15)T.
12.4(15)XZ	This command was implemented on the Cisco 881 and Cisco 888 platforms.

Release	Modification
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Examples

The following is sample output from the **debug ip urlfilter** command when SmartFilter URL filtering configured:

```
Router# debug ip urlfilter detailed
urlfilter:
  Urlfilter Detailed Debugs debugging is on
Router# show ip urlfilter config

N2H2 URL Filtering is ENABLED
Primary N2H2 server configurations
=========================================
N2H2 server IP address:192.168.1.103
N2H2 server port:4005
N2H2 retransmission time out:6 (in seconds)
N2H2 number of retransmission:2
Secondary N2H2 servers configurations
============================================
Other configurations
=====================
Allow Mode:OFF
System Alert:ENABLED
Audit Trail:ENABLED
Log message on N2H2 server:DISABLED
Maximum number of cache entries:5
Maximum number of packet buffers:20
Maximum outstanding requests:1000
fw1_4#
1d15h:URLF:got a socket read event...
1d15h:URLF:socket recv failed.
1d15h:URLF:Closing the socket for server (192.168.1.103:4005)
1d15h:%URLF-3-SERVER_DOWN:Connection to the URL filter server 192.168.1.103 is down
1d15h:URLF:Opening a socket for server (192.168.1.103:4005)
1d15h:URLF:socket fd 0
1d15h:%URLF-5-SERVER_UP:Connection to an URL filter server(192.168.1.103) is made, the
router is returning from ALLOW MODE
1d15h:URLF:got cache idle timer event...
1d16h:URLF:got cache absolute timer event...
1d16h:URLF:got cache idle timer event...
1d16h:URLF:creating uis 0x63A95DB4, pending request 1
1d16h:URLF:domain name not found in the exclusive list
1d16h:URLF:got an cbac queue event...
1d16h:URLF:socket send successful...172.17.192.130:8080) -> 192.168.1.103:1052 seq 3344720064
 wnd 24820
1d16h:URLF:holding pak 0x634A8A08 (172.17.192.130:8080) -> 192.168.1.103:1052 seq 3344721524
 wnd 24820
1d16h:URLF:holding pak 0x634A98CC (172.17.192.130:8080) -> 192.168.1.103:1052 seq 3344722984
 wnd 24820
1d16h:URLF:got a socket read event...
1d16h:URLF:socket recv (header) successful.
1d16h:URLF:socket recv (data) successful.
1d16h:URLF:n2h2 lookup code = 1
1d16h:URLF:Site/URL Blocked:sis 0x63675DC4, uis 0x63A95DB4
1d16h:%URLF-4-URL_BLOCKED:Access denied URL 'http://www.example.com/', client
192.168.1.103:1052 server 172.17.192.130:8080
1d16h:URLF:(192.168.1.103:1052) RST -> 172.17.192.130:8080 seq 3361738063 wnd 0
```

```
1d16h:URLF:(172.17.192.130:8080) FIN -> 192.168.1.103:1052 seq 3344720064 wnd 0
1d16h:URLF:deleting uis 0x63A95DB4, pending requests 0
1d16h:URLF:got cache idle timer event...
1d16h:URLF:creating uis 0x63A95DB4, pending request 1
1d16h:URLF:domain name not found in the exclusive list
1d16h:URLF:got an cbac queue event...
1d16h:URLF:socket send successful...
1d16h:URLF:holding pak 0x634A812C (172.17.192.130:8080) -> 192.168.1.103:1101 seq 3589711120
 wnd 24820
1d16h:URLF:holding pak 0x634A2E7C (172.17.192.130:8080) -> 192.168.1.103:1101 seq 3589712580
 wnd 24820
1d16h:URLF:holding pak 0x634A3464 (172.17.192.130:8080) -> 192.168.1.103:1101 seq 3589714040
 wnd 24820
1d16h:URLF:got a socket read event...
1d16h:URLF:socket recv (header) successful.
1d16h:URLF:socket recv (data) successful.
1d16h:URLF:n2h2 lookup code = 0
1d16h:%URLF-6-URL_ALLOWED:Access allowed for URL 'http://www.example1.com/', client
192.168.1.103:1101 server 172.17.192.130:8080
1d16h:URLF:Site/URL allowed:sis 0x6367D0C4, uis 0x63A95DB4
1d16h:URLF:releasing pak 0x634A812C:(172.17.192.130:8080) -> 192.168.1.103:1101 seq 3589711120
 wnd 24820
1d16h:URLF:releasing pak 0x634A2E7C:(172.17.192.130:8080) -> 192.168.1.103:1101 seq 3589712580
 wnd 24820
1d16h:URLF:releasing pak 0x634A3464:(172.17.192.130:8080) -> 192.168.1.103:1101 seq 3589714040
 wnd 24820
1d16h:URLF:deleting uis 0x63A95DB4, pending requests 0
1d16h:URLF:got cache idle timer event...
1d16h:URLF:creating uis 0x63A9777C, pending request 1
1d16h:URLF:domain name not found in the exclusive list
1d16h:URLF:got an cbac queue event...
1d16h:URLF:socket send successful...
1d16h:URLF:got a socket read event...
1d16h:URLF:socket recv (header) successful.
1d16h:URLF:socket recv (data) successful.
1d16h:URLF:n2h2 lookup code = 1
1d16h:URLF:Site/URL Blocked:sis 0x63677ED4, uis 0x63A9777C
1d16h:%URLF-4-URL_BLOCKED:Access denied URL 'http://www.example2.com/', client
192.168.1.103:1123 server 172.17.192.130:8080
1d16h:URLF:(192.168.1.103:1123) RST -> 172.17.192.130:8080 seq 3536466275 wnd 0
1d16h:URLF:(172.17.192.130:8080) FIN -> 192.168.1.103:1123 seq 3618929551 wnd 0
1d16h:URLF:deleting uis 0x63A9777C, pending requests 0
1d16h:URLF:got cache idle timer event...
```

debug ip verify mib

To view debug output that displays the operation of Unicast Reverse Path Forwarding (RPF) MIB objects and the helper software, use the **debug ip verify mib** command in privileged EXEC mode. To disable debugging for Unicast RPF, use the **no** form of this command.

debug ip verify mib
no debug ip verify mib

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging activity for the operation of Unicast RPF MIB objects and helper software does not occur.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(31)SB2	This command was introduced.
12.2(33)SRC	This command was integrated into Cisco IOS Release 12.2(33)SRC.
12.4(20)T	This command was integrated into Cisco IOS Release 12.4(20)T.
12.2(33)SXI2	This command was integrated into Cisco IOS Release 12.2(33)SXI2.

Usage Guidelines

Debug information for the Unicast RPF MIB is collected only when logging is enabled. Unicast RPF messages are stored in the logging buffer, and they are not displayed on the console unless you use the **debug ip verify mib** command.

Examples

The following example shows sample output of the **debug ip verify mib**command:

```
Router> enable
Router# debug ip verify mib
01:29:45: cipUrpfScalar_get, searchType 161
01:29:45: ipurpfmib_get_scalars
01:29:45: cipUrpfScalar_get, searchType 161
01:29:45: cipUrpfScalar_get, searchType 161
01:29:45: ipurpfmib_get_scalars
01:29:45: cipUrpfScalar_get, searchType 161
01:29:45: cipUrpfScalar_get, searchType 161
01:29:45: ipurpfmib_get_scalars
01:29:45: cipUrpfScalar_get, searchType
161ipurpfmib_get_urpf_entryipurpfmib_get_urpf_entryipurpfmib_get_urpf_entryipurpfmib_get_
urpf_entry
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
01:29:45: cipUrpfIfMonEntry_get, searchType 161
01:29:45: ipurpfmib_get_urpf_ifmon_entry entry: ST 161, if 1, ip 1
```

Related Commands

Command	Description
show ip interface	Displays the usability status of interfaces configured for IP.

debug ip virtual-reassembly

To enable debugging of the virtual fragment reassembly (VFR) subsystem, use the **debug ip virtual-reassembly** command in privileged EXEC mode. To disable VFR debugging, use the **no** form of this command.

debug ip virtual-reassembly [**list** {**access-list** | **extended-access-list**}]
no debug ip virtual-reassembly [**list** {**access-list** | **extended-access-list**}]

Syntax Description

list	(Optional) Enables VFR conditional debugging.
access-list	Filters the generated list of VFR conditional debugging messages. The valid range is from 1 to 199.
extended-access-list	Filters the generated list of extended VFR conditional debugging messages. The valid range is from 1300 to 2699.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
15.0(1)M	The list keyword was introduced.

Examples

The following sample output from the **debug ip virtual-reassembly** command allows you to monitor datagram fragmentation and reassembly status--such as whether a datagram is incomplete and when fragments (from the datagram) are created (after a datagram is determined to be complete).

```
Router# debug ip virtual-reassembly
00:17:35: IP_VFR: fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:0, len:104) in fast
 path...
00:17:35: IP_VFR: created frag state for sa:13.0.0.2, da:17.0.0.2, id:11745...
00:17:35: IP_VFR: pak incomplete cpak-offset:0, cpak-len:104, flag: 1
00:17:35: IP_VFR: dgrm incomplete, returning...
00:17:35: IP_VFR: fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:104, len:104) in
fast path...
00:17:35: IP_VFR: cpak-offset:0, cpak-len:104, npak-offset:104
00:17:35: IP_VFR: pak incomplete cpak-offset:104, cpak-len:104, flag: 1
00:17:35: IP_VFR: dgrm incomplete, returning...
00:17:35: IP_VFR: fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:208, len:104) in
fast path...
00:17:35: IP_VFR: cpak-offset:0, cpak-len:104, npak-offset:104
00:17:35: IP_VFR: cpak-offset:104, cpak-len:104, npak-offset:208
00:17:35: IP_VFR: pak incomplete cpak-offset:208, cpak-len:104, flag: 1
00:17:35: IP_VFR: dgrm incomplete, returning...
00:17:35: IP_VFR: fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:312, len:104) in
fast path...
00:17:35: IP_VFR: cpak-offset:0, cpak-len:104, npak-offset:104
00:17:35: IP_VFR: cpak-offset:104, cpak-len:104, npak-offset:208
00:17:35: IP_VFR: cpak-offset:208, cpak-len:104, npak-offset:312
00:17:35: IP_VFR: pak incomplete cpak-offset:312, cpak-len:104, flag: 1
00:17:35: IP_VFR: dgrm incomplete, returning...
00:17:35: IP_VFR: fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:416, len:92) in fast
 path...
```

```
00:17:35: IP_VFR: cpak-offset:0, cpak-len:104, npak-offset:104
00:17:35: IP_VFR: cpak-offset:104, cpak-len:104, npak-offset:208
00:17:35: IP_VFR: cpak-offset:208, cpak-len:104, npak-offset:312
00:17:35: IP_VFR: cpak-offset:312, cpak-len:104, npak-offset:416
00:17:35: IP_VFR: dgrm complete, switching the frags.
00:17:35: IP_VFR: switching fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:0, len:104)
00:17:35: IP_VFR: switching fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:104,
len:104)
00:17:35: IP_VFR: switching fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:208,
len:104)
00:17:35: IP_VFR: switching fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:312,
len:104)
00:17:35: IP_VFR: switching fragment (sa:13.0.0.2, da:17.0.0.2, id:11745, offset:416, len:92)
00:17:35: IP_VFR: all fragments have been switched.
00:17:35: IP_VFR: pak_subblock_free - pak 0x64A3DC30
00:17:35: IP_VFR: pak_subblock_free - pak 0x6430F010
00:17:35: IP_VFR: pak_subblock_free - pak 0x6430F678
00:17:35: IP_VFR: pak_subblock_free - pak 0x643119B4
00:17:35: IP_VFR: deleted frag state for sa:13.0.0.2, da:17.0.0.2, id:11745
00:17:35: IP_VFR: pak_subblock_free - pak 0x64A3D5C8
```

Related Commands

Command	Description
ip virtual-reassembly	Enables VFR on an interface.

debug ip wccp

To display information about IPv4 Web Cache Communication Protocol (WCCP) services, use the **debug ip wccp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ip wccp {**default** | **vrf** *vrf-name* {**events** | **packets** [**control**]} | **events** | **packets** [{**bypass** | **control** | **redirect**}] | **platform** | **subblocks**}
no debug ip wccp {**default** | **vrf** *vrf-name* {**events** | **packets** [**control**]} | **events** | **packets** [{**bypass** | **control** | **redirect**}] | **platform** | **subblocks**}

Syntax Description

default	Displays information about default WCCP services.
vrf *vrf-name*	Specifies a virtual routing and forwarding (VRF) instance to associate with a service group.
events	Displays information about significant WCCP events.
packets	Displays information about every WCCP packet received or sent by the router.
control	(Optional) Displays information about WCCP control packets.
bypass	(Optional) Displays information about WCCP bypass packets.
redirect	(Optional) Displays information about WCCP redirect packets.
platform	Displays information about the WCCP platform application programming interface (API).
subblocks	Displays information about WCCP subblocks.

Command Default

Debug information is not displayed.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.0(1)M	This command was introduced. This command replaces the **debug ip wccp packets** and **debug ip wccp events** commands.
12.2(33)SRE	This command was integrated into Cisco IOS Release 12.2(33)SRE.
Cisco IOS XE Release 3.1S	This command was integrated into Cisco IOS XE Release 3.1S.

Usage Guidelines

When the **vrf** keyword is not used, the command displays debug information about all WCCP services on the router. The **default** keyword is used to specify default WCCP services.

Examples

The following is sample output from the **debug ip wccp events** command when a Cisco Cache Engine is added to the list of available Web caches:

```
Router# debug ip wccp events
WCCP-EVNT: Built I_See_You msg body w/1 usable web caches, change # 0000000A
WCCP-EVNT: Web Cache 192.168.25.3 added
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000B
WCCP-EVNT: Built I_See_You msg body w/2 usable web caches, change # 0000000C
```

The following is sample output from the **debug ip wccp packets** command. The router is sending keepalive packets to the Cisco Cache Engines at 192.168.25.4 and 192.168.25.3. Each keepalive packet has an identification number associated with it. When the Cisco Cache Engine receives a keepalive packet from the router, it sends a reply with the identification number back to the router.

```
Router# debug ip wccp packets
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003532
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003534
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003533
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003535
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003534
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003536
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003535
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003537
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.4 w/rcvd_id 00003536
WCCP-PKT: Sending I_See_You packet to 192.168.25.4 w/ rcvd_id 00003538
WCCP-PKT: Received valid Here_I_Am packet from 192.168.25.3 w/rcvd_id 00003537
WCCP-PKT: Sending I_See_You packet to 192.168.25.3 w/ rcvd_id 00003539
```

Related Commands

Command	Description
clear ip wccp	Clears the counter for packets redirected using WCCP.
ip wccp	Enables support of the specified WCCP service for participation in a service group.
ip wccp redirect	Enables packet redirection on an outbound or inbound interface using WCCP.
show ip interface	Lists a summary of the IP information and status of an interface.

debug ipc

To display debugging messages about interprocess communication (IPC) activity, use the **debug ipc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc {**all** | **ports** | **seats** | **sessions** | **zones**}
no debug ipc {**all** | **ports** | **seats** | **sessions** | **zones**}

Syntax Description

all	Displays all debugging IPC messages. A confirmation message will appear because enabling this keyword can severely impact performance.
ports	Displays debugging messages related to the creation and deletion of IPC ports.
seats	Displays debugging messages related to the creation and deletion of IPC nodes (seats).
sessions	Displays debugging messages related to the creation and deletion of IPC sessions.
zones	Displays debugging messages related to the creation and deletion of IPC zones.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2	This command was introduced.
12.3(11)T	The **sessions** and **zones** keywords were added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Use the **debug ipc** command to troubleshoot IPC issues discovered when the **show ipc** command is run. The debugging output varies depending on the types of IPC packets that are selected by the different keywords.

Caution Use the **debug ipc all** command with caution because it enables the **debug ipc packets** command and the volume of output can severely impact system performance. A confirmation message is displayed. We recommend that you use one of the other keywords to focus on a specific IPC activity and to limit the volume of output.

Examples

The following example shows the confirmation message that appears when the **debug ipc all** command is entered:

```
Router# debug ipc all
This may severely impact system performance. Continue? [confirm]
```

The following example shows how to enable the display of debugging messages about IPC sessions. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the IPC control session was opened to port 0x1030000, closed, and then cleared--followed by a series of header or data fields.

```
Router# debug ipc sessions
Session level events debugging is on
*Sep 14 13:13:35.435: IPC: Control Session opened to port 0x1030000
*Sep 14 13:13:35.439: -Traceback= 40779898 4077649C 40776A00 40777040 4077554C
*Sep 14 13:13:35.439: IPC: Session 0 to port 0x1030000 closed
*Sep 14 13:13:35.439: -Traceback= 4077A9D4 40776370 4077132C 40771A58 4062EC7C 4028EC8C
40649710 4057F87C
*Sep 14 13:13:35.439: IPC: Session handle of session 0 to port 0x1030000 cleared
*Sep 14 13:13:35.439: -Traceback= 407798EC 4077A9E0 40776370 4077132C 40771A58 4062EC7C
4028EC8C 40649710 4057F87C
```

Related Commands

Command	Description
debug ipc packets	Displays debugging messages about IPC packets.
show ipc	Displays IPC information.

debug ipc acks

To display debugging messages about interprocess communication (IPC) acknowledgments (ACKs), use the **debug ipc acks** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc acks [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**header dump**]
no debug ipc acks [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**header dump**]

Syntax Description

rx	(Optional) Displays debugging messages related to the retrieval of IPC ACK messages.
tx	(Optional) Displays debugging messages related to the transmission of IPC ACK messages.
dest	(Optional) Displays debugging messages related to a destination port of IPC ACK messages. If not specified, information about all destinations is displayed. • Use the *destination-port-id* argument to specify a hexadecimal number that represents a destination port ID. The range is from 0 to FFFFFFFF.
source	(Optional) Displays debugging information about messages from an IPC node. If not specified, information about all nodes is displayed. • Use the *source-seat-id* argument to specify a hexadecimal number that represents a source seat ID. The range is from 0 to FFFFFFFF.
session	(Optional) Displays debugging messages related to an IPC session. If not specified, information about all sessions is displayed. • Use the *session-id* argument to specify a session ID. The range is from 0 to 65535.
header dump	(Optional) Displays only the packet header information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.

Usage Guidelines

Use the **debug ipc acks** command to troubleshoot IPC ACK issues. To enable debugging for other IPC activities, use the **debug ipc** command.

Examples

The following example shows how to enable the display of packet headers only when debugging IPC ACK messages. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the server received an ACK HDR--followed by a series of header or data fields.

```
Router# debug ipc acks header dump
Aug 19 03:52:36.136:IPC:Server received ACK HDR:442A64E0 src:100000A, dst:406116E8,
index:-1, seq:22045, sz:0, type:65535, flags:2 hi:1F371, lo:0
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug ipc errors

To display debugging messages about interprocess communication (IPC) errors and warnings, use the **debug ipc errors**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc errors [**driver**] [**sequence**] [**timeout**]
no debug ipc errors [**driver**] [**sequence**] [**timeout**]

Syntax Description

driver	(Optional) Displays debugging messages related to IPC errors at the driver (transport) medium.
sequence	(Optional) Displays information related to IPC messages that have sequence-related issues, such as duplicate or unexpected messages.
timeout	(Optional) Displays only information related to IPC messages that have timed out.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2	This command was introduced.
12.3(11)T	The **driver**, **sequence**, and **timeout** keywords were added.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Use the **debug ipc errors** command to troubleshoot IPC error issues. To enable debugging for other IPC activities, use the **debug ipc** command. The debugging output varies depending on the type of IPC activity that is specified.

Examples

The following example shows how to enable the display of error debugging information about IPC messages that have timed out. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the message number 4428D3D0 timed out waiting for an acknowledgment (Ack)--followed by a series of header or data fields.

```
Router# debug ipc errors timeout
Message Timeouts debugging is on
*Sep 14 14:42:17.103: IPC: Message 4428D3D0 timed out waiting for Ack
*Sep 14 14:42:17.103: IPC:  MSG: ptr: 0x4428D3D0, flags: 0x88, retries: 6, seq: 0x1030002,
refcount: 2,
retry: 00:00:00, rpc_result = 0x0, data_buffer = 0x4442AB10, header = 0x4442AED4,
data = 0x4442AEF4
HDR: src: 0x10000, dst: 0x103000A, index: 0, seq: 2, sz: 512, type: 0, flags: 0x400
hi: 0x1EC, lo: 0x4442AEF4
DATA: 00 00 00 05 00 00 00 00 00 00 00 3A 00 00 00 00 00 00 00 00
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug ipc events

To display debugging messages about interprocess communication (IPC) events, use the **debug ipc events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc events [**flushes**] [**retries**]
no debug ipc events [**flushes**] [**retries**]

Syntax Description

flushes	(Optional) Displays only information related to IPC messages that are flushed.
retries	(Optional) Displays only information related to IPC messages that are re-sent.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2	This command was introduced.
12.3(11)T	The **flushes** and **retries** keywords were added.\
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Use the **debug ipc events** command to troubleshoot IPC events issues. To enable debugging for other IPC activities, use the **debug ipc** command.

Examples

The following example shows how to enable the display of debugging messages about IPC events:

```
Router# debug ipc events
Special Events debugging is on
```

The following example shows how to enable the display of event debugging information about IPC messages that are re-sent. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that there was a retry attempt for a specific message--followed by a series of header or data fields.

```
Router# debug ipc events retries
Message Retries debugging is on
*Sep 14 14:46:44.151: IPC: Retry attempt for MSG: ptr: 0x442AFE74, flags: 0x88,
retries:4, seq: 0x1030003,
refcount: 2, retry: 00:00:00, rpc_result = 0x0, data_buffer = 0x445EBA44,
header =0x445EBE08, data = 0x445EBE28
HDR: src: 0x10000, dst: 0x103000A, index: 0, seq: 3, sz: 512, type: 0, flags: 0x400
hi:0x201, lo: 0x445EBE28
DATA: 00 00 00 05 00 00 00 00 00 00 00 3A 00 00 00 00 00 00 03 D2
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug ipc fragments

To display debugging messages about interprocess communication (IPC) fragments, use the **debug ipc fragments**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc fragments [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**type** *application-type*] [**flags** *header-flag*] [**sequence** *sequence*] [**msgidhi** *msg-id-high*] [**msgidlo** *msg-id-low*] [**data offset** *offset-from-header* **value** *value-to-match* **dump** *bytes*] [**size** *size*] [**header dump**]
no debug ipc fragments [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**type** *application-type*] [**flags** *header-flag*] [**sequence** *sequence*] [**msgidhi** *msg-id-high*] [**msgidlo** *msg-id-low*] [**data offset** *offset-from-header* **value** *value-to-match* **dump** *bytes*] [**size** *size*] [**header dump**]

Syntax Description

rx	(Optional) Displays debugging messages related to the retrieval of IPC fragments.
tx	(Optional) Displays debugging messages related to the transmission of IPC fragments.
dest	(Optional) Displays debugging messages related to a destination port of IPC fragments. If not specified, information about all destinations is displayed. • Use the *destination-port-id* argument to specify a hexadecimal number that represents a destination port ID. The range is from 0 to FFFFFFFF.

source	(Optional) Displays debugging information about messages from an IPC node. If not specified, information about all nodes is displayed. • Use the *source-seat-id* argument to specify a hexadecimal number that represents a source seat ID. The range is from 0 to FFFFFFFF.
session	(Optional) Displays debugging messages related to an IPC session. If not specified, information about all sessions is displayed. • Use the *session-id* argument to specify a session ID. The range is from 0 to 65535.
type	(Optional) Displays debugging messages related to a type of IPC fragments. If not specified, information about all application types is displayed. • Use the *application-type* argument to specify a hexadecimal number that represents an application. The range is from 0 to FFFF.
flags	(Optional) Displays debugging messages related to an IPC fragment's header flag. If not specified, information about all header flags is displayed. • Use the *header-flag* argument to specify a hexadecimal number that represents a header flag value. The range is from 0 to FFFF.
sequence	(Optional) Displays debugging messages related to a sequence number of an IPC fragment. If not specified, information about all sequence numbers is displayed. • Use the *sequence* argument to specify a sequence number. The range is from 0 to 65535.
msgidhi	(Optional) Displays debugging messages related to the higher byte of the unique ID of an IPC fragment. • Use the *msg-id-high* argument to specify a hexadecimal number that represents a higher byte of the unique ID. The range is from 0 to FFFFFFFF.
msgidlo	(Optional) Displays debugging messages related to the lower byte of the unique ID of an IPC fragment. • Use the *msg-id-low* argument to specify a hexadecimal number that represents a lower byte of the unique ID. The range is from 0 to FFFFFFFF.

data	(Optional) Displays debugging messages related to the IPC fragment payload. If not specified, information about all of the IPC fragment's payload is displayed. • **offset** --(Optional) Displays offset IPC data. If this keyword is configured, the **value** keyword must also be configured. • Use the *offset-from-header* argument to specify the offset value from the start of the IPC data. The range is from 0 to 65535. • Use the **value** keyword to configure the value expected at the offset of the IPC data. • Use the *value-to-match* argument to specify the hexadecimal number that represents the value expected at the offset of the IPC data. The range is from 0 to FF. • **dump** --(Optional) Configures the number of data bytes to display. • Use the *bytes* argument to specify the number of data bytes. The range is from 0 to 65535.
size	(Optional) Displays IPC fragment debugging messages of a specific size. If not specified, information about messages of any size is displayed. • Use the *size* argument to specify the message size in rows. The range is from 0 to 65535.
header dump	(Optional) Displays only the packet header information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.

Usage Guidelines

Use the **debug ipc fragments**command to troubleshoot IPC fragment issues. To enable debugging for other IPC activities, use the **debug ipc** command.

Examples

The following example shows how to enable the display of debugging information about IPC fragments. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the server received a fragment message--followed by a series of header or data fields.

```
Router# debug ipc fragments
IPC Fragments debugging is on
01:43:55: IPC: Server received fragment MSG: ptr: 0x503A4348, flags: 0x100, retries: 0,
seq: 0x0,
refcount: 1, retry: never, rpc_result = 0x0, data_buffer = 0x433809E8, header = 0x8626748,
data = 0x8626768
HDR: src: 0x10000, dst: 0x2210015, index: 0, seq: 1, sz: 1468, type: 0, flags: 0x10
hi:0x9AA, lo: 0x7D0
DATA: 00 00 00 01 00 00 00 00 00 00 00 AA 00 00 00 00 00 00 17 E4
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug ipc nacks

To display debugging messages about interprocess communication (IPC) negative acknowledgments (NACKs), use the **debug ipc nacks** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc nacks [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**header dump**]
no debug ipc nacks [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**header dump**]

Syntax Description

rx	(Optional) Displays debugging messages related to the retrieval of IPC NACK messages.
tx	(Optional) Displays debugging messages related to the transmission of IPC NACK messages.
dest	(Optional) Displays debugging messages related to a destination port of IPC NACK messages. If not specified, information about all destinations is displayed. • Use the *destination-port-id* argument to specify a hexadecimal number that represents a destination port ID. The range is from 0 to FFFFFFFF.
source	(Optional) Displays debugging information about messages from an IPC node. If not specified, information about all nodes is displayed. • Use the *source-seat-id* argument to specify a hexadecimal number that represents a source seat ID. The range is from 0 to FFFFFFFF.
session	(Optional) Displays debugging messages related to an IPC session. If not specified, information about all sessions is displayed. • Use the *session-id* argument to specify a session ID. The range is from 0 to 65535.
header dump	(Optional) Displays only the packet header information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.

Usage Guidelines

Use the **debug ipc nacks** command to troubleshoot IPC NACK issues. To enable debugging for other IPC activities, use the **debug ipc** command.

Examples

The following example shows how to enable the display of packet headers only when debugging IPC NACK messages. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the server sent a NACK message and received a NACK header--followed by a series of header or data fields.

```
Router# debug ipc nacks header dump
IPC Nacks debugging is on
01:46:11: IPC: Server sent NACK MSG: ptr: 0x432A7428, flags: 0x100, retries: 0, seq: 0x0,
refcount: 1, retry: never, rpc_result = 0x0, data_buffer = 0x431E4B50, header = 0x855F508,
data = 0x855F528
HDR: src: 0x2210015, dst: 0x10000, index: 1, seq: 3, sz: 0, type: 0, flags: 0x100
hi: 0x4A9, lo: 0x85AA3E8
01:46:11: SP: IPC: Server received NACK HDR: E46A448 src: 2210015, dst: 10000, index: 1,
seq: 3, sz: 0, type: 0, flags: 100 hi: 4A9, lo: 85AA3E8
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug ipc packets

To display debugging messages about interprocess communication (IPC) packets, use the **debug ipc packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc packets [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**type** *application-type*] [**flags** *header-flag*] [**sequence** *sequence*] [**msgidhi** *msg-id-high*] [**msgidlo** *msg-id-low*] [**data offset** *offset-from-header* **value** *value-to-match* **dump** *bytes*] [**size** *size*] [**header dump**]
no debug ipc packets [{**rx** | **tx**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**type** *application-type*] [**flags** *header-flag*] [**sequence** *sequence*] [**msgidhi** *msg-id-high*] [**msgidlo** *msg-id-low*] [**data offset** *offset-from-header* **value** *value-to-match* **dump** *bytes*] [**size** *size*] [**header dump**]

Syntax Description

rx	(Optional) Displays debugging messages related to the retrieval of IPC packets.
tx	(Optional) Displays debugging messages related to the transmission of IPC packets.
dest	(Optional) Displays debugging messages related to a destination port of IPC packets. If not specified, information about all destinations is displayed. • Use the *destination-port-id* argument to specify a hexadecimal number that represents a destination port ID. The range is from 0 to FFFFFFFF.
source	(Optional) Displays debugging information about messages from an IPC node. If not specified, information about all nodes is displayed. • Use the *source-seat-id* argument to specify a hexadecimal number that represents a source seat ID. The range is from 0 to FFFFFFFF.

session	(Optional) Displays debugging messages related to an IPC session. If not specified, information about all sessions is displayed. • Use the *session-id* argument to specify a session ID. The range is from 0 to 65535.
type	(Optional) Displays debugging messages related to a type of IPC packet. If not specified, information about all application types is displayed. • Use the *application-type* argument to specify a hexadecimal number that represents an application. The range is from 0 to FFFF.
flags	(Optional) Displays debugging messages related to an IPC packet header flag. If not specified, information about all header flags is displayed. • Use the *header-flag* argument to specify a hexadecimal number that represents a header flag value. The range is from 0 to FFFF.
sequence	(Optional) Displays debugging messages related to a sequence number of an IPC packet. If not specified, information about all sequence numbers is displayed. • Use the *sequence* argument to specify a sequence number. The range is from 0 to 65535.
msgidhi	(Optional) Displays debugging messages related to the higher byte of the unique ID of an IPC packet. • Use the *msg-id-high* argument to specify a hexadecimal number that represents a higher byte of the unique ID. The range is from 0 to FFFFFFFF.
msgidlo	(Optional) Displays debugging messages related to the lower byte of the unique ID of an IPC packet. • Use the *msg-id-low* argument to specify a hexadecimal number that represents a lower byte of the unique ID. The range is from 0 to FFFFFFFF.
data	(Optional) Displays debugging messages related to the IPC packet payload. If not specified, information about all of the IPC packet's payload is displayed. • **offset** --(Optional) Displays offset IPC data. If this keyword is configured, the **value** keyword must also be configured. • Use the *offset-from-header* argument to specify the offset value from the start of the IPC data. The range is from 0 to 65535. • Use the **value** keyword to configure the value expected at the offset of the IPC data. • Use the *value-to-match* argument to specify the hexadecimal number that represents the value expected at the offset of the IPC data. The range is from 0 to FF. • **dump** --(Optional) Configures the number of data bytes to display. • Use the *bytes* argument to specify the number of data bytes. The range is from 0 to 65535.

size	(Optional) Displays IPC packet debugging messages of a specific size. If not specified, information about messages of any size is displayed. • Use the *size* argument to specify the message size in rows. The range is from 0 to 65535.
header dump	(Optional) Displays only the packet header information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.

Usage Guidelines

Use the **debug ipc packets** command to troubleshoot IPC packet issues. To enable debugging for other IPC activities, use the **debug ipc** command.

Caution Use the **debug ipc packets** command with caution because the volume of output can severely impact system performance. A confirmation message is displayed. We recommend that you use one of the optional keywords to focus on a specific IPC activity and to limit the volume of output.

Examples

The following example shows how to enable the display of IPC packet debugging messages and includes some sample output. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the IPC server received a message--followed by a series of header or data fields.

```
Router# debug ipc packets
This may severely impact system performance. Continue?[confirm] Y
Aug 19 030612.297 IPC Server received MSG ptr 0x441BE75C, flags 0x80, retries 0,
seq 0x0, refcount 1, retry never, rpc_result = 0x0, data_buffer = 0x443152A8,
header = 0x4431566C, data = 0x4431568C
HDR src 0x1060000, dst 0x1000C, index 2, seq 0, sz 28, type 770,
flags 0x40 hi 0x1F25B, lo 0x442F0BC0
DATA 00 00 00 06 00 00 00 02 00 00 00 06 00 E7 00 02 00 00 00 00
```

The following example shows how to enable the display of IPC messages received with a destination port of 0x1000C in session 1 with a message size of 500 rows.

```
Router# debug ipc packets rx dest 1000C session 1 size 500
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug ipc rpc

To display debugging messages about interprocess communication (IPC) remote-procedure call (RPC) packets, use the **debug ipc rpc**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipc rpc [{**rx** | **tx**}] [{**query** | **response**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**type** *application-type*] [**flags** *header-flag*] [**sequence** *sequence*] [**msgidhi** *msg-id-high*] [**msgidlo** *msg-id-low*] [**data offset** *offset-from-header* **value** *value-to-match* **dump** *bytes*] [**size** *size*] [**header dump**]
no debug ipc rpc [{**rx** | **tx**}] [{**query** | **response**}] [**dest** *destination-port-id*] [**source** *source-seat-id*] [**session** *session-id*] [**type** *application-type*] [**flags** *header-flag*] [**sequence** *sequence*] [**msgidhi** *msg-id-high*] [**msgidlo** *msg-id-low*] [**data offset** *offset-from-header* **value** *value-to-match* **dump** *bytes*] [**size** *size*] [**header dump**]

Syntax Description

rx	(Optional) Displays debugging messages related to the retrieval of IPC RPC packets.
tx	(Optional) Displays debugging messages related to the transmission of IPC RPC packets.
query	(Optional) Displays debugging messages related to IPC RPC queries.
response	(Optional) Displays debugging messages related to IPC RPC responses.
dest	(Optional) Displays debugging messages related to a destination port of IPC RPC packets. If not specified, information about all destinations is displayed. • Use the *destination-port-id* argument to specify a hexadecimal number that represents a destination port ID. The range is from 0 to FFFFFFFF.
source	(Optional) Displays debugging information about messages from an IPC node. If not specified, information about all nodes is displayed. • Use the *source-seat-id* argument to specify a hexadecimal number that represents a source seat ID. The range is from 0 to FFFFFFFF.
session	(Optional) Displays debugging messages related to an IPC session. If not specified, information about all sessions is displayed. • Use the *session-id* argument to specify a session ID. The range is from 0 to 65535.
type	(Optional) Displays debugging messages related to a type of IPC RPC message. If not specified, information about all application types is displayed. • Use the *application-type* argument to specify a hexadecimal number that represents an application. The range is from 0 to FFFF.
flags	(Optional) Displays debugging messages related to an IPC RPC message header flag. If not specified, information about all header flags is displayed. • Use the *header-flag* argument to specify a hexadecimal number that represents a header flag value. The range is from 0 to FFFF.

sequence	(Optional) Displays debugging messages related to a sequence number of an IPC RPC message. If not specified, information about all sequence numbers is displayed. • Use the *sequence* argument to specify a sequence number. The range is from 0 to 65535.
msgidhi	(Optional) Displays debugging messages related to the higher byte of the unique ID of an IPC RPC message. • Use the *msg-id-high* argument to specify a hexadecimal number that represents a higher byte of the unique ID. The range is from 0 to FFFFFFFF.
msgidlo	(Optional) Displays debugging messages related to the lower byte of the unique ID of an IPC RPC message. • Use the *msg-id-low* argument to specify a hexadecimal number that represents a lower byte of the unique ID. The range is from 0 to FFFFFFFF.
data	(Optional) Displays debugging messages related to the IPC RPC payload. If not specified, information about all of the IPC RPC's payload is displayed. • **offset** --(Optional) Displays offset IPC data. If this keyword is configured, the **value** keyword must also be configured. • Use the *offset-from-header* argument to specify the offset value from the start of the IPC data. The range is from 0 to 65535. • Use the **value** keyword to configure the value expected at the offset of the IPC data. • Use the *value-to-match* argument to specify the hexadecimal number that represents the value expected at the offset of the IPC data. The range is from 0 to FF. • **dump** --(Optional) Configures the number of data bytes to display. • Use the *bytes* argument to specify the number of data bytes. The range is from 0 to 65535.
size	(Optional) Displays IPC RPC debugging messages of a specific size. If not specified, information about messages of any size is displayed. • Use the *size* argument to specify the message size in rows. The range is from 0 to 65535.
header dump	(Optional) Displays only the packet header information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(11)T	This command was introduced.

Usage Guidelines

Use the **debug ipc rpc**command to troubleshoot IPC RPC packet issues. To enable debugging for other IPC activities, use the **debug ipc** command. The debugging output varies depending on the type of IPC activity that is specified.

Examples

The following example shows how to enable the display of packet headers only when debugging IPC RPC response messages. The debugging output varies depending on the type of IPC activity that is specified. Each entry includes some text explanation--the example below shows that the server received an RPC response--followed by a series of header or data fields.

```
Router# debug ipc rpc response header dump source 2210003
RPC debugging is on
01:53:43: SP: IPC: Server received RPC Reply HDR: E450048 src: 2210003, dst: 10000,
index:0, seq: 1716, sz: 4, type: 2914, flags: 208 hi: A07, lo: E264DE8
```

Related Commands

Command	Description
debug ipc	Displays IPC debugging information.

debug iphc ipc

To display the IP header compression (IPHC) interprocessor communication (IPC) messages that are passed between the route processor (RP) and line cards (LCs), use the **debug iphc ipc**command in privileged EXEC mode. To disable the display of these messages, use the **no** form of this command.

debug iphc ipc [{**events** | **statistics**}]
no debug iphc ipc [{**events** | **statistics**}]

Syntax Description

events	(Optional) Displays IPHC IPC command and control events.
statistics	(Optional) Displays IPHC IPC counter updates.

Command Default

IPHC IPC messages are not displayed.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(32)SY	This command was introduced.
12.4(10)	This command was integrated into Cisco IOS Release 12.4(10).

Usage Guidelines

If you issue the **debug iphc ipc** command without keywords, all the IPC messages that are passed between the RP and the LC are displayed. On routers with many interfaces and distributed systems, the number of IPC messages becomes unwieldy, because of all the counter updates. To display only the events that indicate interface state changes, issue the **debug iphc ipc events** command.

Examples

The following example enables the display of all IPHC IPC messages:

```
Router# debug iphc ipc
IPHC IPC statistics debugging is on
IPHC IPC event debugging is on
The following example disables IPHC IPC statistics debugging:
Router# no debug iphc ipc statistics
IPHC IPC statistics debugging is off
```

The following example enables the display of IPHC IPC event messages:

```
Router# debug iphc ipc events
IPHC IPC event debugging is on
```

The command output shows the event messages as the interface changes from enabled to administratively down:

```
%OSPF-5-ADJCHG: Process 1, Nbr 10.10.10.10 on Multilink8 from FULL to DOWN
%LINK-5-CHANGED: Interface Multilink8, changed state to administratively down.
IPHC IPC 2: Set Negotiated mesg (Mu PPP 128 2 0)
IPHC Mu8: Distributed FS disabled
IPHC IPC 2: Send Set Configured mesg (Mu PPP 128 2 0)
IPHC IPC Mu8: i/f state change complete (Up/Down: 0/1)
```

The following example enables the display of IPHC IPC counter updates:

```
Router# debug iphc ipc statistics
IPHC IPC statistics debugging is on
```

The command output shows the interface counter updates:

```
IPHC IPHC 2: recv Stats msg, count:4
IPHC IPC Mu8: stats update from LC
IPHC IPC Mu6: stats update from LC
IPHC IPC Se2/0/0/3:0: stats update from LC
IPHC IPC Se2/0/0/1:0: stats update from LC
```

Related Commands

Command	Description
show interfaces	Displays statistics for all interfaces.
show ipc	Displays IPC statistics.

debug ipv6 cef drop

To display debug messages for Cisco Express Forwarding for IPv6 (CEFv6) and distributed CEFv6 (dCEFv6) dropped packets, use the **debug ipv6 cef drop**command in privileged EXEC mode. To disable debug messages for CEFv6 and dCEFv6 dropped packets, use the **no** form of this command.

debug ipv6 cef drop [**rpf**]
no debug ipv6 cef drop

Syntax Description

rpf	(Optional) Displays packets dropped by the IPv6 CEF Unicast Reverse-Path Forwarding (Unicast RPF) feature.

Command Default

Debugging for CEFv6 and dCEFv6 dropped packets is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(25)S	The **rpf** keyword was added.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 cef drop** command is similar to the **debug ip cef drops** command, except that it is IPv6-specific.

Note By default, the network server sends the output from **debug** commands and system error messages to the console. To redirect debug output, use the **logging** command options in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on **debug** commands and redirecting debug output, refer to the Release 12.3 *Cisco IOS Debug Command Reference* .

Examples

The following is sample output from the **debug ipv6 cef drop**command:

```
Router# debug ipv6 cef drop
*Aug 30 08:20:51.169: IPv6-CEF: received packet on Serial6/0/2
*Aug 30 08:20:51.169: IPv6-CEF: found no adjacency for 2001:0DB8::1 reason 2
*Aug 30 08:20:51.169: IPv6-CEF: packet not switched: code 0x1
```

The table below describes the significant fields shown in the display.

Table 23: debug ipv6 cef drop Field Descriptions

Field	Description
IPv6-CEF: received packet on Serial6/0/2	Cisco Express Forwarding has received a packet addressed to the router via serial interface 6/0/2.

Field	Description
IPv6-CEF: found no adjacency for 2001:0DB8::1	Cisco Express Forwarding has found no adjacency for the IPv6 address prefix of 2001:0DB8::1.
IPv6-CEF: packet not switched	Cisco Express Forwarding has dropped the packet.

Related Commands

Command	Description
debug ipv6 cef events	Displays debug messages for CEFv6 and dCEFv6 general events.
debug ipv6 cef table	Displays debug messages for CEFv6 and dCEFv6 table modification events.

debug ipv6 cef events

To display debug messages for Cisco Express Forwarding for IPv6 (CEFv6) and distributed CEFv6 (dCEFv6) general events, use the **debug ipv6 cef events**command in privileged EXEC mode. To disable debug messages for CEFv6 and dCEFv6 general events, use the **no** form of this command.

debug ipv6 cef events
no debug ipv6 cef events

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging for CEFv6 and dCEFv6 general events is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 cef events**command is similar to the **debug ip cef events**command, except that it is IPv6-specific.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on **debug** commands and redirecting debug output, refer to the Release 12 *Cisco IOS Debug Command Reference* .

Examples

The following is sample output from the **debug ipv6 cef events**command:

```
Router# debug ipv6 cef events
IPv6 CEF packet events debugging is on
Router#
*Aug 30 08:22:57.809: %LINK-3-UPDOWN: Interface Serial6/0/2, changed state to up
*Aug 30 08:22:58.809: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial6/0/2, changed
 state to up
*Aug 30 08:23:00.821: CEFv6-IDB: Serial6/0/2 address 2001:0DB8::248 add download succeeded
```

The table below describes the significant fields shown in the display.

Table 24: debug ipv6 cef events Field Descriptions

Field	Description
Interface Serial6/0/2, changed state to up	Indicates that the interface hardware on serial interface 6/0/2 is currently active.
Line protocol on Interface Serial6/0/2, changed state to up	Indicates that the software processes that handle the line protocol consider the line usable for serial interface 6/0/2.
Serial6/0/2 address 2001:0DB8::248 add download succeeded	The IPv6 address 2001:0DB8::248 was downloaded successfully.

Related Commands

Command	Description
debug ipv6 cef table	Displays debug messages for CEFv6 and dCEFv6 table modification events.

debug ipv6 cef hash

To display debug messages for Cisco Express Forwarding for IPv6 (CEFv6) and distributed CEFv6 (dCEFv6) load-sharing hash algorithm events, use the **debug ipv6 cef hash**command in privileged EXEC mode. To disable debug messages for CEFv6 and dCEFv6 load-sharing hash algorithm events, use the **no** form of this command.

debug ipv6 cef hash
no debug ipv6 cef hash

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging for CEFv6 and dCEFv6 load-sharing hash algorithm events is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 cef hash**command is similar to the **debug ip cef hash**command, except that it is IPv6-specific.

Use this command when changing the load-sharing algorithm to display IPv6 hash table details.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

Related Commands

Command	Description
debug ipv6 cef events	Displays debug messages for CEFv6 and dCEFv6 general events.
debug ipv6 cef table	Displays debug messages for CEFv6 and dCEFv6 table modification events.

debug ipv6 cef receive

To display debug messages for Cisco Express Forwarding for IPv6 (CEFv6) and distributed CEFv6 (dCEFv6) packets that are process-switched on the router, use the **debug ipv6 cef receive**command in privileged EXEC mode. To disable debug messages for CEFv6 and dCEFv6 packets that are process-switched on the router, use the **no** form of this command.

debug ipv6 cef receive
no debug ipv6 cef receive

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging for CEFv6 and dCEFv6 packets that are process-switched on the router is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 cef receive**command is similar to the **debug ip cef receive**command, except that it is IPv6-specific.

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debug output, refer to the Release 12 *Cisco IOS Debug Command Reference* .

Examples

The following is sample output from the **debug ipv6 cef receive**command when another router in the network pings 2001:0DB8::2 which is a local address on this box:

```
Router# debug ipv6 cef receive
IPv6 CEF packet receives debugging is on
router#
*Aug 30 08:25:14.869: IPv6CEF-receive: Receive packet for 2001:0DB8::2
*Aug 30 08:25:14.897: IPv6CEF-receive: Receive packet for 2001:0DB8::2
*Aug 30 08:25:14.925: IPv6CEF-receive: Receive packet for 2001:0DB8::2
*Aug 30 08:25:14.953: IPv6CEF-receive: Receive packet for 2001:0DB8::2
*Aug 30 08:25:14.981: IPv6CEF-receive: Receive packet for 2001:0DB8::2
```

The table below describes the significant fields shown in the display.

Table 25: debug ipv6 cef receive Field Descriptions

Field	Description
IPv6CEF-receive: Receive packet for 2001:0DB8::2	Cisco Express Forwarding has received a packet addressed to the router.

Related Commands

Command	Description
debug ipv6 cef events	Displays debug messages for CEFv6 and dCEFv6 general events.
debug ipv6 cef table	Displays debug messages for CEFv6 and dCEFv6 table modification events.

debug ipv6 cef table

To display debug messages for Cisco Express Forwarding for IPv6 (CEFv6) and distributed CEFv6 (dCEFv6) table modification events, use the **debug ipv6 cef table**command in privileged EXEC mode. To disable debug messages for CEFv6 and dCEFv6 table modification events, use the **no** form of this command.

debug ipv6 cef table [**background**]
no debug ipv6 cef table [**background**]

Syntax Description

background	(Optional) Sets CEFv6 and dCEFv6 table background updates.

Command Default

Debugging for CEFv6 and dCEFv6 table modification events is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.

Usage Guidelines

The **debug ipv6 cef table**command is similar to the **debug ip cef table**command, except that it is IPv6-specific.

This command is used to record CEFv6 and dCEFv6 table events related to the Forwarding Information Base (FIB) tables. Types of events include the following:

- Routing updates that populate the FIB tables
- Flushing of the FIB tables
- Adding or removing of entries to the FIB tables
- Table reloading process

Note By default, the network server sends the output from debug commands and system error messages to the console. To redirect debug output, use the logging command options in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debug output, refer to the *Cisco IOS Debug Command Reference* .

Examples

The following is sample output from the **debug ipv6 cef table**command when a static route is added:

```
Router# debug ipv6 cef table
IPv6 CEF table debugging is on
router(config)# ipv6 route 5555::/64 serial 2/0 3000::2
router(config)#
*Feb 24 08:46:09.187: IPv6CEF-Table: Event add, 5555::/64
*Feb 24 08:46:09.187: IPv6 CEF table: Created path_list 01184570
*Feb 24 08:46:09.187: IPv6 CEF table: Adding path 01181A80 to path_list 01184570 old path
count=0
*Feb 24 08:46:09.187: IPv6 CEF table: No matching list for path list 01184570
*Feb 24 08:46:09.187: IPv6 CEF table: Adding fib entry 0117EE80 to path_list 01184570 old
refcount=0
*Feb 24 08:46:09.187: IPv6 CEF table: Added path_list 01184570 to hash 50
*Feb 24 08:46:09.187: IPv6 CEF: Linking path 01181A80 to adjacency 01138E28
*Feb 24 08:46:09.187: IPv6 CEF table: Created 0 loadinfos for path_list 01184570
*Feb 24 08:46:09.187: IPv6CEF-Table: Validated 5555::/64
```

The following is sample output when the static route is removed:

```
router(config)# no ipv6 route 5555::/64 serial 2/0 3000::2
router(config)#
*Feb 24 08:46:43.871: IPv6CEF-Table: Event delete, 5555::/64
*Feb 24 08:46:43.871: IPv6CEF-Table: Invalidated 5555::/64
*Feb 24 08:46:43.871: IPv6CEF-Table: Deleted 5555::/64
*Feb 24 08:46:43.871: IPv6 CEF table: Removing fib entry 0117EE80 from path_list 01184570
old refcount=1
*Feb 24 08:46:43.871: IPv6 CEF table: Removed path_list 01184570 from hash 50
*Feb 24 08:46:43.871: IPv6 CEF table: Freeing path_list 01184570 refcount=0
*Feb 24 08:46:43.871: IPv6 CEF table: Freeing all 1 paths in path_list 01184570
*Feb 24 08:46:43.871: IPv6 CEF: deleting path 01181A80
```

Related Commands

Command	Description
debug ipv6 cef events	Displays debug messages for CEFv6 and dCEFv6 general events.

debug ipv6 dhcp

To enable debugging for Dynamic Host Configuration Protocol (DHCP) for IPv6, use the **debug ipv6 dhcp** command in privileged EXEC mode. To disable debugging for DHCP for IPv6, use the **no** form of this command.

debug ipv6 dhcp [**detail**]
no debug ipv6 dhcp [**detail**]

Syntax Description

detail	(Optional) Displays detailed information about DHCP for IPv6 message decoding.

Command Default

Debugging for the DHCP for IPv6 is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)T	This command was introduced.
12.4(24)T	This command was integrated into Cisco IOS Release 12.4(24)T.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.
12.2(33)SRE	This command was modified. It was integrated into Cisco IOS Release 12.2(33)SRE.

Usage Guidelines

The **debug ipv6 dhcp detail** command is used to show debug information related to the server address assignment.

Examples

The following example enables debugging for DHCP for IPv6:

```
Router# debug ipv6 dhcp detail
IPv6 DHCP debugging is on (detailed)
```

Related Commands

Command	Description
debug ipv6 dhcp database	Enables debugging for the DHCP for IPv6 binding database agent.
debug ipv6 dhcp relay	Enables the DHCP for IPv6 relay agent debugging.

debug ipv6 dhcp database

To enable debugging for the Dynamic Host Configuration Protocol (DHCP) for IPv6 binding database agent, use the **debug ipv6 dhcp database** command in privileged EXEC mode. To disable the display of debug messages for the DHCP for IPv6 binding database agent, use the **no** form of this command.

debug ipv6 dhcp database
no debug ipv6 dhcp database

Syntax Description

This command has no keywords or arguments.

Command Default

Debugging for the DHCP for IPv6 binding database agent is disabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)T	This command was introduced.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.

Usage Guidelines

The **debug ipv6 dhcp database** command enables debugging for DHCP for IPv6 database processing.

Examples

The following example enables debugging for the DHCP for IPv6 binding database agent:

```
Router# debug ipv6 dhcp database
```

Related Commands

Command	Description
debug ipv6 dhcp	Enables debugging for DHCP for IPv6.

debug ipv6 dhcp redundancy

To enable Dynamic Host Configuration Protocol for IPv6 (DHCPv6) server redundancy debugging, use the **debug ipv6 dhcp redundancy** command in privileged EXEC mode. To disable DHCPv6 server redundancy debugging, use the **no** form of this command.

debug ipv6 dhcp redundancy [**detail**]
no debug ipv6 dhcp redundancy [**detail**]

Syntax Description

detail	(Optional) Displays detailed DHCPv6 High Availability (HA) packet information.

Command Default

DHCPv6 server redundancy debugging is disabled by default.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(1)S	This command was introduced.
Cisco IOS XE Release 3.5S	This command was integrated into Cisco IOS XE Release 3.5S.

Usage Guidelines

To debug DHCPv6 server redundancy, use the **debug ipv6 dhcp redundancy** command in privileged EXEC mode. To view detailed DHCPv6 HA packet information, use the optional **detail** keyword.

Examples

The following example shows how to enable DHCPv6 redundancy debugging:

```
Router# debug ipv6 dhcp redundancy
```

Related Commands

Command	Description
debug ipv6 dhcp relay	Enables DHCPv6 relay agent debugging.

debug ipv6 dhcp relay

To enable DHCP for IPv6 relay agent debugging, use the **debug ipv6 dhcp relay**command in user EXEC or privileged EXEC mode. To disable DHCP for IPv6 relay agent debugging, use the **no** form of this command.

debug ipv6 dhcp relay [**bulk-lease**]
no debug ipv6 dhcp relay [**bulk-lease**]

Syntax Description

bulk-lease	(Optional) Enables bulk lease query debugging flows.

Command Modes

User EXEC (>)
Privileged EXEC (#)

Command History

Release	Modification
12.3(11)T	This command was introduced.
Cisco IOS XE Release 2.1	This command was integrated into Cisco IOS XE Release 2.1.
15.1(1)S	This command was modified. The **bulk-lease** keyword was added.

Usage Guidelines

The DHCP functions for IPv6 client, server, and relay agent are mutually exclusive on an interface. When one of these functions is enabled and a user tries to configure a different function on the same interface, one of the following messages is displayed: Interface is in DHCP client mode, Interface is in DHCP server mode, or Interface is in DHCP relay mode.

Examples

The following example enables DHCP for IPv6 relay agent debugging:

```
Router# debug ipv6 dhcp relay
```

Related Commands

Command	Description
debug ipv6 dhcp	Enables DHCP debugging for IPv6.

debug ipv6 eigrp

To display information about the Enhanced Interior Gateway Routing Protocol (EIGRP) for IPv6 protocol, use the **debug ipv6 eigrp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 eigrp [*as-number*] [{**neighbor** *ipv6-address* | **notification** | **summary**}]
no debug ipv6 eigrp

Syntax Description

as-number	(Optional) Autonomous system number.
neighbor *ipv6-address*	(Optional) IPv6 address of the neighboring router.
notification	(Optional) Displays EIGRP for IPv6 events and notifications in the console of the router.
summary	(Optional) Displays a summary of EIGRP for IPv6 routing information.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.4(6)T	This command was introduced.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

Usage Guidelines

Because the **debug ipv6 eigrp** command generates a substantial amount of output, use it only when traffic on the network is light.

Examples

The following example enables debugging output:

```
Router# debug ipv6 eigrp
```

debug ipv6 icmp

To display debugging messages for IPv6 Internet Control Message Protocol (ICMP) transactions (excluding IPv6 ICMP neighbor discovery transactions), use the **debug ipv6 icmp**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ipv6 icmp
no debug ipv6 icmp

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging for IPv6 ICMP is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.
12.0(21)ST	This command was integrated into Cisco IOS Release 12.0(21)ST.
12.0(22)S	This command was integrated into Cisco IOS Release 12.0(22)S.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(28)SB	This command was integrated into Cisco IOS Release 12.2(28)SB.
12.2(25)SG	This command was integrated into Cisco IOS Release 12.2(25)SG.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Release	Modification
12.2(33)SXH	This command was integrated into Cisco IOS Release 12.2(33)SXH.
12.2(33)SB	This command's output was modified on the Cisco 10000 series router for the PRE3 and PRE4.
15.1(1)S	This command was integrated into Cisco IOS 15.1(1)S.

Usage Guidelines

The **debug ipv6 icmp**command is similar to the **debug ip icmp**command, except that it is IPv6-specific. When you run this command, you can view echo reply messages that are generated in response to echo requests.

Note By default, the network server sends the output from **debug** commands and system error messages to the console. To redirect debugging output, use the logging command options in global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

This command helps you determine whether the router is sending or receiving IPv6 ICMP messages. Use it, for example, when you are troubleshooting an end-to-end connection problem.

Note For more information about the fields in **debug ipv6 icmp** output, refer to RFC 2463, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)* .

Cisco 10000 Series Router Usage Guidelines

In Cisco IOS Release 12.2(33)SB, output from the debug ipv6 icmp command displays information similar to the following:

```
ICMPv6: Received echo reply from 2010:1:1:1:1:1:1:2
```

In Cisco IOS Release 12.2(31)SB, the debug ipv6 icmp command output displays information similar to the following:

```
ICMPv6: Received ICMPv6 packet from 2010:1:1:1:1:1:1:2, type 129
```

Examples

The following is sample output from the **debug ipv6 icmp**command:

```
Router# debug ipv6 icmp
13:28:40:ICMPv6:Received ICMPv6 packet from 2000:0:0:3::2, type 136
13:28:45:ICMPv6:Received ICMPv6 packet from FE80::203:A0FF:FED6:1400, type 135
13:28:50:ICMPv6:Received ICMPv6 packet from FE80::203:A0FF:FED6:1400, type 136
13:28:55:ICMPv6:Received ICMPv6 packet from FE80::203:A0FF:FED6:1400, type 135
```

The table below describes significant fields shown in the first line of the display.

Table 26: debug ipv6 icmp Field Descriptions

Field	Description
13:28:40:	Indicates the time (hours:minutes:seconds) at which the ICMP neighbor discovery event occurred.
n w*n* d: (not shown in sample output)	Indicates time (weeks, days) since last reboot of the event occurring. For example, 1w4d: indicates the time (since the last reboot) of the event occurring was 1 week and 4 days ago.
ICMPv6:	Indication that this message describes an ICMP version 6 packet.
Received ICMPv6 packet from 2000:0:0:3::2	IPv6 address from which the ICMP version 6 packet is received.
type 136	The number variable indicates one of the following IPv6 ICMP message types: • 1--Destination unreachable. The router cannot forward a packet that was sent or received. • 2--Packet too big. The router attempts to send a packet that exceeds the maximum transmission unit (MTU) of a link between itself and the packet destination. • 3--Time exceeded. Either the hop limit in transit or the fragment reassembly time is exceeded. • 4--Parameter problem. The router attempts to send an IPv6 packet that contains invalid parameters. An example is a packet containing a next header type unsupported by the router that is forwarding the packet. • 128--Echo request. The router received an echo reply. • 129--Echo reply. The router sent an echo reply. • 133--Router solicitation messages. Hosts send these messages to prompt routers on the local link to send router advertisement messages. • 134--Router advertisement messages. Routers periodically send these messages to advertise their link-layer addresses, prefixes for the link, and other link-specific information. These messages are also sent in response to router solicitation messages. • 135--Neighbor solicitation messages. Nodes send these messages to request the link-layer address of a station on the same link. • 136--Neighbor advertisement messages. Nodes send these messages, containing their link-local addresses, in response to neighbor solicitation messages. • 137--Redirect messages. Routers send these messages to hosts when a host attempts to use a less-than-optimal first hop address when forwarding packets. These messages contain a better first hop address that should be used instead.

Following are examples of the IPv6 ICMP messages types that can be displayed by the **debug ipv6 icmp** command:

- ICMP echo request and ICMP echo reply messages. In the following example, an ICMP echo request is sent to address 2052::50 and an ICMP echo reply is received from address 2052::50.

```
1w4d:ICMPv6:Sending echo request to 2052::50
1w4d:ICMPv6:Received echo reply from 2052::50
```

- ICMP packet too big messages. In the following example, a router tried to forward a packet to destination address 2052::50 via the next hop address 2052::52. The size of the packet was greater than 1280 bytes, which is the MTU of destination address 2052::50. As a result, the router receives an ICMP packet too big message from the next hop address 2052::52.

```
1w4d:Received ICMP too big from 2052::52 about 2052::50, MTU=1300
```

- ICMP parameter problem messages. In the following example, an ICMP parameter problem message is received from address 2052::52.

```
1w4d:Received ICMP parameter problem from 2052::52
```

- ICMP time exceeded messages. In the following example, an ICMP time exceeded message is received from address 2052::52.

```
1w4d:Received ICMP time exceeded from 2052::52
```

- ICMP unreachable messages. In the following example, an ICMP unreachable message with code 1 is received from address 2052::52. Additionally, an ICMP unreachable message with code 1 is sent to address 2060::20 about address 2062::20.

```
1w4d:Received ICMP unreachable code 1 from 2052::52
1w4d:Sending ICMP unreachable code 1 to 2060::20 about 2062::20
```

The table below lists the codes for ICMP unreachable messages.

Table 27: ICMP Unreachable Messages--Code Descriptions

Code	Description
0	The router has no route to the packet destination.
1	Although the router has a route to the packet destination, communication is administratively prohibited.
3	The address is unreachable.
4	The port is unreachable.

Related Commands

Command	Description
debug ipv6 nd	Displays debugging messages for IPv6 ICMP neighbor discovery transactions.