

debug clns packet

To display information about packet receipt and forwarding to the next interface, use the **debug clns packet** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug clns packet

no debug clns packet

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns packet

CLNS: Forwarding packet size 157
      from 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.00 STUPI-RBS
      to 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
CLNS: Echo PDU received on Ethernet0 from
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00!
CLNS: Sending from 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00 to
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

In the following lines, the first line indicates that a Connectionless Network Service (CLNS) packet of size 157 bytes is being forwarded. The second line indicates the network service access point (NSAP) and system name of the source of the packet. The third line indicates the destination NSAP for this packet. The fourth line indicates the next hop system ID, interface, and SNPA of the router interface used to forward this packet.

```
CLNS: Forwarding packet size 157
      from 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.00 STUPI-RBS
      to 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

In the following lines, the first line indicates that the router received an echo PDU on the specified interface from the source NSAP. The second line indicates which source NSAP is used to send a CLNS packet to the destination NSAP, as shown on the third line. The fourth line indicates the next hop system ID, interface, and SNPA of the router interface used to forward this packet.

```
CLNS: Echo PDU received on Ethernet0 from
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00!
CLNS: Sending from 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00 to
      47.0005.80ff.ef00.0000.0001.5940.1600.8906.4017.00
      via 1600.8906.4017 (Ethernet0 0000.0c00.bda8)
```

debug clns routing

To display debugging information for all Connectionless Network Service (CLNS) routing cache updates and activities involving the CLNS routing table, use the **debug clns routing** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug clns routing

no debug clns routing

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug clns routing** command:

```
Router# debug clns routing
```

```
CLNS-RT: cache increment:17  
CLNS-RT: Add 47.0023.0001.0000.0000.0003.0001 to prefix table, next hop 1920.3614.3002  
CLNS-RT: Aging cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06  
CLNS-RT: Deleting cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
```

The following line indicates that a change to the routing table has resulted in an addition to the fast-switching cache:

```
CLNS-RT: cache increment:17
```

The following line indicates that a specific prefix route was added to the routing table, and indicates the next hop system ID to that prefix route. In other words, when the router receives a packet with the prefix 47.0023.0001.0000.0000.0003.0001 in the destination address of that packet, it forwards that packet to the router with the MAC address 1920.3614.3002.

```
CLNS-RT: Add 47.0023.0001.0000.0000.0003.0001 to prefix table, next hop 1920.3614.3002
```

The following lines indicate that the fast-switching cache entry for a certain network service access point (NSAP) has been invalidated and then deleted:

```
CLNS-RT: Aging cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06  
CLNS-RT: Deleting cache entry for: 47.0023.0001.0000.0000.0003.0001.1920.3614.3002.06
```

debug cls message

To display information about Cisco Link Services (CLS) messages, use the **debug cls message** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cls message

no debug cls message

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug cls message** command displays the primitives (state), selector, header length, and data size.

Examples

The following is sample output from the **debug cls message** command. For example, CLS-->DLU indicates the direction of the flow that is described by the status. From CLS to DLU, a request was established to the connection endpoint. The header length is 48 bytes, and the data size is 104 bytes.

```
Router# debug cls message

(FRAS Daemon:CLS-->DLU):
  ID_STN.Ind to uSAP: 0x607044C4 sel: LLC hlen: 40, dlen: 54
(FRAS Daemon:CLS-->DLU):
  ID_STN.Ind to uSAP: 0x6071B054 sel: LLC hlen: 40, dlen: 46
(FRAS Daemon:DLU-->SAP):
  REQ_OPNSTN.Reg to pSAP: 0x608021F4 sel: LLC hlen: 48, dlen: 104
(FRAS Daemon:CLS-->DLU):
  REQ_OPNSTN.Cfm(NO_REMOTE_STN) to uCEP: 0x607FFE84 sel: LLC hlen: 48, dlen: 104
```

The status possibilities include the following: enabled, disabled, request open station, open station, close station, activate SA, deactivate SAP, XID, XID station, connect station, signal station, connect, disconnect, connected, data, flow, unnumbered data, modify SAP, test, activate ring, deactivate ring, test station, and unnumbered data station.

Related Commands

| Command | Description |
|---------------------------|------------------------------------------------------------------|
| debug fras error | Displays information about FRAS protocol errors. |
| debug fras message | Displays general information about FRAS messages. |
| debug fras state | Displays information about FRAS data-link control state changes. |

debug cls vdlc

To display information about Cisco Link Services (CLS) Virtual Data Link Control (VDLC), use the **debug cls vdlc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cls vdlc

no debug cls vdlc

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug cls message** command displays primitive state transitions, selector, and source and destination MAC and service access points (SAPs).

Also use the **show cls** command to display additional information on CLS VDLC.



Caution

Use the **debug cls vdlc** command with caution because it can generate a substantial amount of output.

Examples

The following messages are sample output from the **debug cls vdlc** command. In the following scenario, the SNA service point—also called *native service point (NSP)*—is setting up two connections through VDLC and data-link switching (DLSw): one from NSP to VDLC and one from DLSw to VDLC. VDLC joins the two.

The NSP initiates a connection from 4000.05d2.0001 as follows:

```
VDLC: Req Open Stn Req PSap 0x7ACE00, port 0x79DF98
      4000.05d2.0001(0C)->4000.1060.1000(04)
```

In the next message, VDLC sends a test station request to DLSw for destination address 4000.1060.1000.

```
VDLC: Send UFrame E3: 4000.05d2.0001(0C)->4000.1060.1000(00)
```

In the next two messages, DLSw replies with test station response, and NSP goes to a half-open state. NSP is waiting for the DLSw connection to VDLC.

```
VDLC: Sap to Sap TEST_STN_RSP VSap 0x7B68C0 4000.1060.1000(00)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPENING->VDLC_HALF_OPEN
```

The NSP sends an exchange identification (XID) and changes state as follows:

```
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_HALF_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to SAP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04) via bridging SAP (DLSw)
```

In the next several messages, DLSw initiates its connection, which matches the half-open connection with NSP:

```
VDLC: Req Open Stn Req PSap 0x7B68C0, port 0x7992A0
      4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: two-way connection established
VDLC: 4000.1060.1000(04)->4000.05d2.0001(0C): VDLC_IDLE->VDLC_OPEN
```

In the following messages, DLSw sends an XID response, and NSP's connection goes from the state XID Response Pending to Open. The XID exchange follows:

```

V DLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_XID_RSP_PENDING->V DLC_OPEN
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_OPEN->V DLC_XID_RSP_PENDING
V DLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
V DLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_XID_RSP_PENDING->V DLC_OPEN
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_OPEN->V DLC_XID_RSP_PENDING
V DLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
V DLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_XID_RSP_PENDING->V DLC_OPEN
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_OPEN->V DLC_XID_RSP_PENDING
V DLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
V DLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_XID_RSP_PENDING->V DLC_OPEN
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_OPEN->V DLC_XID_RSP_PENDING
V DLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)

```

When DLSw is ready to connect, the front-end processor (FEP) sends a set asynchronous balanced mode extended (SABME) command as follows:

```

V DLC: CEP to CEP CONNECT_REQ 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: 4000.05d2.0001(0C)->4000.1060.1000(04): V DLC_XID_RSP_PENDING->V DLC_OPEN

```

In the following messages, NSP accepts the connection and sends an unnumbered acknowledgment (UA) to the FEP:

```

V DLC: CEP to CEP CONNECT_RSP 4000.05d2.0001(0C)->4000.1060.1000(04)
V DLC: FlowReq QUENCH OFF 4000.1060.1000(04)->4000.05d2.0001(0C)

```

The following messages show the data flow:

```

V DLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)
.
.
.
V DLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)

```

Related Commands

| Command | Description |
|-----------------------------------|-----------------------------------------|
| debug cls message | Displays information about CLS messages |

debug cns config

To turn on debug messages related to the CNS Configuration Agent, use the **debug cns config** command in privileged EXEC mode. To turn off debug messages related to the Configuration Agent, use the **no** version of this command.

```
debug cns config {agent | all | connection | notify}
```

```
no debug cns config {agent | all | connection | notify}
```

Syntax Description

| | |
|-------------------|-----------------------------------------------------------------|
| agent | Displays debug messages related to the CNS configuration agent. |
| all | Displays all debug messages. |
| connection | Displays debug messages related to configuration connections. |
| notify | Displays debug messages related to CNS configurations. |

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|------------|-----------------------------------------------------------------------|
| 12.2(2)T | This command was introduced. |
| 12.0(18)ST | This command was integrated into Cisco IOS Release 12.0 ST. |
| 12.2(8)T | This command was implemented on the Cisco 2600 and Cisco 3600 series. |

Usage Guidelines

Use this command to turn on or turn off debug messages related to the CNS Configuration Agent.

Examples

In the following example, debugging messages are enabled for CNS configuration processes:

```
Router# debug cns config all

00:04:09: config_id_get: entered
00:04:09: config_id_get: Invoking cns_id_mode_get()
00:04:09: config_id_get: cns_id_mode_get() returned INTERNAL
00:04:09: config_id_get: successful exit cns_config_id=minnal,cns_config_id_len=6
00:04:09: cns_establish_connect_intf(): The device is already connected with the config
server
00:04:09: cns_initial_config_agent(): connecting with port 80
00:04:09: pull_config() entered
00:04:09: cns_config_id(): returning config_id=minnal
00:04:09: Message finished 150 readend
00:04:09: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 82
00:04:10: %SYS-5-CONFIG_I: Configured from console by console
```

Related Commands

| Command | Description |
|--------------------------------------|---------------------------------------------------------|
| debug cns config | Displays information on CNS configurations. |
| debug cns event | Displays information on CNS events. |
| debug cns management | Displays information on CNS management. |
| debug cns xml-parser | Displays information on the CNS XML parser. |
| cns config cancel | Cancels a CNS configuration. |
| cns config initial | Starts the initial CNS Configuration Agent. |
| cns config partial | Starts the partial CNS Configuration Agent. |
| cns config retrieve | Gets the configuration of a routing device using CNS. |
| show cns config | Displays information about the CNS Configuration Agent. |

debug cns event

To turn on debugging messages related to the Cisco Networking Services (CNS) Event Gateway, use the **debug cns event** command in privileged EXEC mode. To turn off the debug messages related to the Event Gateway, use the **no** form of this command.

```
debug cns event {agent | all | connection | subscriber}
```

```
no debug cns event {agent | all | connection | subscriber}
```

Syntax Description

| | |
|-------------------|-------------------------------------------------------|
| agent | Displays debug messages related to the event agent. |
| all | Displays all debug messages. |
| connection | Displays debug messages related to event connections. |
| subscriber | Displays debug messages related to subscribers. |

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|------------|----------------------------------------------------------------------------------|
| 12.2(2)T | This command was introduced. |
| 12.0(18)ST | This command was integrated into the Cisco IOS 12.0 ST Release. |
| 12.2(8)T | This command was implemented on Cisco 2600 series and Cisco 3600 series routers. |

Usage Guidelines

Use this command to turn on or turn off debug messages related to the CNS Event Gateway.

Examples

In the following example, debugging messages about all CNS Events are enabled:

```
Router# debug cns event all

00:09:14: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 82
00:09:14: event_agent():event_agent starting ..
00:09:14: event_agent_open_connection(): attempting socket connect to Primary Gateway
00:09:14: event_agent_open_connection():cns_socket_connect() succeeded:return_code=0
00:09:14: event_agent_open_connection():timeout_len=1:ka_total_timeout =0:
        total_timeout=0
00:09:14: event_id_get: entered
00:09:14: event_id_get: Invoking cns_id_mode_get()
00:09:14: event_id_get: cns_id_mode_get() returned INTERNAL
00:09:14: event_id_get: successful exit cns_event_id=test1, cns_event_id_len=5
00:09:14: ea_devid_send(): devid sent DUMP OF DEVID MSG
82C920A0:          00120000 00010774          .....t
82C920B0: 65737431 00000402 020000          est1.....
```

■ debug cns event

```

00:09:14: event_agent_get_input(): cli timeout=0: socket:0x0
00:09:14: process_all_event_agent_event_items():process_get_wakeup(&major, &minor)=TRUE:
major=0
.
.
.
00:09:14: add_subjectANDhandle_to_subject_table():p_subject_entry=0x82E3EEDC:
p_subject_entry_list=0x82619CD8
00:09:14: add_subjectANDhandle_to_subject_table():add 'user_entry' entry succeeded:user
entry =0x82C92AF4:queue_handle=0x82C913FC
00:09:14: %SYS-
5-CONFIG_I: Configured from console by console

```

Related Commands

| Command | Description |
|-----------------------|-------------------------------------------------|
| cns event | Configures the CNS Event Gateway. |
| show cns event | Displays information about the CNS Event Agent. |

debug cns management

To display information about Cisco Networking Services (CNS) management, use the **debug cns management** command in privileged EXEC mode. To disable the currently specified debugging, use the **no** form of this command.

```
debug cns management {snmp | xml}
```

```
no debug cns management {snmp | xml}
```

Syntax Description

| | |
|-------------|------------------------------------------------------------------------------------------------------------------------------|
| snmp | Displays debug messages related to nongranular Simple Network Management Protocol (SNMP) encapsulated CNS-management events. |
| xml | Displays debug messages related to granular eXtensible Markup Language (XML) encapsulated CNS-management events. |

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|----------|---------------------------------------------------------------------------------|
| 12.2(8)T | This command was introduced on Cisco 2600 series and Cisco 3600 series routers. |

Examples

In the following example, debugging messages about SNMP- and XML-encapsulated CNS-management events are enabled:

```
Router# debug cns management snmp
Router# debug cns management xml

Router# show debugging

CNS Management (SNMP Encapsulation) debugging is on
CNS Management (Encap XML) debugging is on

Router# show running-config | include cns

cns mib-access encapsulation snmp
cns mib-access encapsulation xml
cns notifications encapsulation snmp
cns notifications encapsulation xml
cns event 10.1.1.1 11011
Router#
00:12:50: Enqueued a notification in notif_q
00:12:50: ea_produce succeeded Subject:cisco.cns.mibaccess:notification Message Length:385
00:12:50: Trap sent via CNS Transport Mapping.
Router#
00:13:31: Response sent via CNS Transport Mapping.
Router#
00:14:38: Received a request
00:14:38: ea_produce succeeded Subject:cisco.cns.mibaccess:response Message Length:241
```

| Related Commands | Command | Description |
|------------------|--------------------------------------|-------------------------------------------------------------------------------------------|
| | cns event | Configures the CNS event gateway, which provides CNS event services to Cisco IOS clients. |
| | debug cns config | Displays information on CNS configurations. |
| | debug cns xml-parser | Displays information on the CNS XML parser. |
| | show debugging | Displays information about the types of debugging that are enabled for your router. |
| | show running-config | Displays the current running configuration. |

debug cns xml-parser

To turn on debug messages related to the Cisco Networking Services (CNS) eXtensible Markup Language (XML) parser, use the **debug cns xml-parser** command in privileged EXEC mode. To turn off debug messages related to the XML parser, use the **no** version of this command.

debug cns xml-parser

no debug cns xml-parser

Syntax Description

| | |
|-------------------|----------------------------------------------------|
| xml-parser | Turns on debug messages related to the XML parser. |
|-------------------|----------------------------------------------------|

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|------------|-----------------------------------------------------------------------|
| 12.2(2)T | This command was introduced. |
| 12.0(18)ST | This command was integrated into Cisco IOS Release 12.0(18)ST. |
| 12.2(8)T | This command was implemented on the Cisco 2600 and Cisco 3600 series. |

Examples

In the following example, debugging messages for the CNS XML parser are enabled:

```
Router# debug cns xml-parser

00:12:05: Registering tag <config-server>
00:12:05: Registering tag <server-info>
00:12:05: Registering tag <ip-address>
00:12:05: Registering tag <web-page>
00:12:05: Registering tag <config-event>
00:12:05: Registering tag <identifier>
00:12:05: Registering tag <config-id>
00:12:05: Registering tag <config-data>
00:12:05: Registering tag <cli>
00:12:05: Registering tag <error-info>
00:12:05: Registering tag <error-message>
00:12:05: Registering tag <line-number>
00:12:05: Registering tag <config-write>
00:12:05: Registering tag <exec-cmd-event>
00:12:05: Registering tag <identifier-exec>
00:12:05: Registering tag <event-response>
00:12:05: Registering tag <reply-subject>
00:12:05: Registering tag <server-response>
00:12:05: Registering tag <ip-address-exec>
00:12:05: Registering tag <port-number>
00:12:05: Registering tag <url>
00:12:05: Registering tag <cli-exec>
00:12:05: Registering tag <config-pwd>
00:12:06: Pushing tag <config-data> on to stack
```

```
00:12:06: open tag is <config-data>
00:12:06: Pushing tag <config-id> on to stack
00:12:06: open tag is <config-id>
00:12:06: Popping tag <config-id> off stack
00:12:06: close tag is </config-id>
00:12:06: Pushing tag <cli> on to stack
00:12:06: open tag is <cli>
00:12:06: Popping tag <cli> off stack
00:12:06: close tag is </cli>
00:12:06: Popping tag <config-data> off stack
00:12:06: close tag is </config-data>
00:12:06: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 96
```

Related Commands

| Command | Description |
|-----------------------|-------------------------------------------------|
| cns event | Configures the CNS Event Gateway. |
| show cns event | Displays information about the CNS Event Agent. |

debug compress

To debug compression, enter the **debug compress** privileged EXEC configuration command. To disable debugging output, use the **no** form of this command.

debug compress

no debug compress

Syntax Description This command has no arguments or keywords.

Defaults Disabled

| Command History | Release | Modification |
|-----------------|---------|------------------------------|
| | 10.0 | This command was introduced. |

Usage Guidelines Use this command to display output from the compression and decompression configuration you made. Live traffic must be configured through the Cisco 2600 access router with a data compression Advanced Interface Module (AIM) installed for this command to work.

Examples The following example is output from the **debug compress** command, which shows that compression is taking place on a Cisco 2600 access router using data compression AIM hardware compression is configured correctly:

```
Router# debug compress

COMPRESS debugging is on
Router#compr-in:pak:0x810C6B10 npart:0 size:103
pak:0x810C6B10 start:0x02406BD4 size:103 npart:0
compr-out:pak:0x8118C8B8 stat:0x00000000 npart:1 size:71 lcb:0xED
pak:0x8118C8B8 start:0x0259CD3E size:71 npart:1
    mp:0x8118A980 start:0x0259CD3E size:71

decmp-in:pak:0x81128B78 start:0x0255AF44 size:42 npart:1 hdr:0xC035
pak:0x81128B78 start:0x0255AF44 size:42 npart:1
    mp:0x81174480 start:0x0255AF44 size:42
decmp-out:pak:0x8118C8B8 start:0x025B2C42 size:55 npart:1 stat:0
pak:0x8118C8B8 start:0x025B2C42 size:55 npart:1
    mp:0x8118B700 start:0x025B2C42 size:55
```

[Table 40](#) describes the significant fields in the display.

Table 40 *debug compress Field Descriptions*

| Field | Description |
|-----------|-------------------------------------------------|
| compr-in | Indicates that a packet needs to be compressed. |
| compr-out | Indicates completion of compression of packet. |

Table 40 *debug compress Field Descriptions (continued)*

| Field | Description |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| decmp-in | Indicates receipt of a compressed packet that needs to be decompressed. |
| decmp-out | Indicates completion of decompression of a packet. |
| pak:0x810C6B10 | Provides the address in memory of a software structure that describes the compressed packet. |
| start:0x02406BD4 size:103 npart:0 | The “npart:0” indicates that the packet is contained in a single, contiguous area of memory. The start address of the packet is 0x02406bd4 and the size of the packet is 103. |
| start:0x0259CD3E size:71 npart:1 | The “npart:1” indicates that the packet is contained in 1 or more regions of memory. The start address of the packet is 0x0259CD3E and the size of the packet is 71. |
| mp:0x8118A980 start:0x0259CD3e size:71 | Describes one of these regions of memory. |
| mp:0x8118A980 | Provides the address of a structure describing this region. |
| start 0x0259CD3E | Provides the address of the start of this region. |

Related Commands

| Command | Description |
|--------------------------|----------------------------------------------------------------------------------------------------------|
| debug frame-relay | Displays debugging information about the packets that are received on a Frame Relay interface. |
| debug ppp | Displays information on traffic and exchanges in an internetwork implementing the PPP. |
| show compress | Displays compression statistics. |
| show diag | Displays hardware information including DRAM, SRAM, and the revision-level information on the line card. |

debug condition

To limit output for some debug commands based on specified conditions, use the **debug condition** command in privileged EXEC mode. To removed the specified condition, use the **no** form of this command.

```
debug condition {username username | called dial-string | caller dial-string | vcid vc-id | ip
ip-address}
```

```
no debug condition {condition-id | all}
```

Syntax Description

| | |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| username <i>username</i> | Generates debugging messages for interfaces with the specified username. |
| called <i>dial-string</i> | Generates debugging messages for interfaces with the called party number. |
| caller <i>dial-string</i> | Generates debugging messages for interfaces with the calling party number. |
| vcid <i>vc-id</i> | Generates debugging messages for the VC ID specified. |
| ip <i>ip-address</i> | Generates debugging messages for the IP address specified. |
| <i>condition-id</i> | Removes the condition indicated. |
| all | Removes all debugging conditions, and conditions specified by the debug condition interface command. Use this keyword to disable conditional debugging and reenable debugging for all interfaces. |

Defaults

All debugging messages for enabled protocol-specific **debug** commands are generated.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11.3(2)AA | This command was introduced. |
| 12.0(23)S | This command was integrated into Cisco IOS Release 12.0(23)S. This command was updated with the vcid and ip keywords to support the debugging of Any Transport over MPLS (AToM) messages. |
| 12.2(14)S | This command was integrated into Cisco IOS Release 12.2(14)S. |
| 12.2(15)T | This command was integrated into Cisco IOS Release 12.2(15)T. |

Usage Guidelines

Use the **debug condition** command to restrict the debug output for some commands. If any **debug condition** commands are enabled, output is only generated for interfaces associated with the specified keyword. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

If multiple **debug condition** commands are enabled, output is displayed if at least one condition matches. All the conditions do not need to match.

The **no** form of this command removes the debug condition specified by the condition identifier. The condition identifier is displayed after you use a **debug condition** command or in the output of the **show debug condition** command. If the last condition is removed, debugging output resumes for all interfaces. You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the **debug condition** command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions. The commands that are affected by the **debug condition** commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the **debug condition** command:

- **debug aaa {accounting | authorization | authentication}**
- **debug dialer {events | packets}**
- **debug isdn {q921 | q931}**
- **debug modem {oob | trace}**
- **debug ppp {all | authentication | chap | error | negotiation | multilink events | packet}**

Examples

Example 1

In the following example, the router displays debugging messages only for interfaces that use a username of fred. The condition identifier displayed after the command is entered identifies this particular condition.

```
Router# debug condition username fred
```

```
Condition 1 set
```

Example 2

The following example specifies that the router should display debugging messages only for VC 1000:

```
Router# debug condition vcid 1000
```

```
Condition 1 set
```

```
01:12:32: 1000 Debug: Condition 1, vcid 1000 triggered, count 1
```

```
01:12:32: 1000 Debug: Condition 1, vcid 1000 triggered, count 1
```

Other debugging commands are enabled, but they will only display debugging for VC 1000.

```
Router# debug mpls l2transport vc event
```

```
AToM vc event debugging is on
```

```
Router# debug mpls l2transport vc fsm
```

```
AToM vc fsm debugging is on
```

The following commands shut down the interface where VC 1000 is established.

```
Router(config)# interface s3/1/0
```

```
Router(config-if)# shut
```

The debugging output shows the change to the interface where VC 1000 is established.

```
01:15:59: AToM MGR [13.13.13.13, 1000]: Event local down, state changed from established to remote ready
```

```
01:15:59: AToM MGR [13.13.13.13, 1000]: Local end down, vc is down
```

```
01:15:59: AToM SMGR [13.13.13.13, 1000]: Processing imposition update, vc_handle 6227BCF0, update_action 0, remote_vc_label 18
```

```
01:15:59: AToM SMGR [13.13.13.13, 1000]: Imposition Disabled
```

```
01:15:59: ATOM SMGR [13.13.13.13, 1000]: Processing disposition update, vc_handle
6227BCF0, update_action 0, local_vc_label 755
01:16:01:%LINK-5-CHANGED: Interface Serial3/1/0, changed state to administratively down
01:16:02:%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/1/0, changed state to
down
```

Related Commands

| Command | Description |
|----------------------------------|--------------------------------------------------------------------|
| debug condition interface | Limits output for some debugging commands based on the interfaces. |

debug condition application voice

To display debugging messages for only the specified VoiceXML application, use the **debug condition application voice** command in privileged EXEC mode. To disable the debugging condition for an application, use the **no** form of this command.

debug condition application voice *application-name*

no debug condition application voice *application-name*

Syntax Description

| | |
|-------------------------|------------------------------------------------------------------------------------------------|
| <i>application-name</i> | Name of the VoiceXML application for which you want to display all enabled debugging messages. |
|-------------------------|------------------------------------------------------------------------------------------------|

Defaults

If this command is not configured, debugging messages are enabled for all VoiceXML applications.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|----------------------------------------------------------------------------------------------------------|
| 12.2(11)T | This command was introduced on the Cisco 3640, Cisco 3660, Cisco AS5300, Cisco AS5350, and Cisco AS5400. |

Usage Guidelines

- This command filters debugging output only for the **debug vxml** and **debug http client** commands, except that it does not filter output for the **debug vxml error**, **debug vxml background**, **debug http client error**, or **debug http client background** commands. It does not filter messages for any other debug commands such as the **debug voip ivr** command or the **debug voice ivr** command.
- This command filters debugging output for all VoiceXML applications except the application named in the command. When this command is configured, the gateway displays debugging messages only for the specified VoiceXML application.
- To filter debugging output with this command, the <cisco-debug> element must be enabled in the VoiceXML document. For more information about the <cisco-debug> element, refer to the [Cisco VoiceXML Programmer's Guide](#).
- To see debugging output for VoiceXML applications, you must first configure global debug commands such as the **debug vxml** command or the **debug http client** command. If no global debugging commands are turned on, you do not see debugging messages even if the **debug condition application voice** command is configured and the <cisco-debug> element is enabled in the VoiceXML document.
- This command can be configured multiple times to display output for more than one application.
- To see which debug conditions have been set, use the **show debug condition** command.

Examples

The following example disables debugging output for all applications except the myapp1 application, if the <cisco-debug> element is enabled in the VoiceXML documents that are executed by myapp1:

```
Router# debug condition application voice myapp1
```

Related Commands

| Command | Description |
|-----------------------------|------------------------------------------------------------------------------------|
| debug http client | Displays debugging messages for the HTTP client. |
| debug vxml | Displays debugging messages for VoiceXML features. |
| show debug condition | Displays the debugging conditions that have been enabled for VoiceXML application. |

debug condition glbp

To display debugging messages about Gateway Load Balancing Protocol (GLBP) conditions, use the **debug condition glbp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug condition glbp *interface-type interface-number group [forwarder]*

no debug condition glbp *interface-type interface-number group [forwarder]*

Syntax Description

| | |
|-------------------------|--------------------------------------------------------------------------------------|
| <i>interface-type</i> | Interface type for which output is displayed. |
| <i>interface-number</i> | Interface number for which output is displayed. |
| <i>group</i> | GLBP group number in the range from 0 to 1023. |
| <i>forwarder</i> | (Optional) Number in the range from 1 to 255 used to identify a virtual MAC address. |

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|---------------------------------------------------------------|
| 12.2(14)S | This command was introduced. |
| 12.2(15)T | This command was integrated into Cisco IOS Release 12.2(15)T. |

Examples

The following is sample output from the **debug condition glbp** command:

```
Router# debug condition glbp fastethernet 0/0 10 1
```

```
Condition 1 set
5d23h: Fa0/0 GLBP10.1 Debug: Condition 1, glbp Fa0/0 GLBP10.1 triggered, count 1
```

Related Commands

| Command | Description |
|---------------------------|----------------------------------------------------------------------------------------|
| debug glbp errors | Displays debugging messages about GLBP errors. |
| debug glbp events | Displays debugging messages about GLBP events. |
| debug glbp packets | Displays debugging messages about GLBP packets. |
| debug glbp terse | Displays a limited range of debugging messages about GLBP errors, events, and packets. |

debug ccsip events

To enable tracing of events that are specific to service provider interface (SPI), use the **debug ccsip events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip events

no debug ccsip events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

| | |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12.1(1)T | This command was introduced. |
| 12.1(3)T | The output of this command was changed. |
| 12.2(2)XA | Support was added for the Cisco AS5350 and Cisco AS5400. |
| 12.2(2)XB1 | This command was introduced on the Cisco AS5850. |
| 12.2(11)T | This command was integrated into Cisco IOS Release 12.2(11)T. |
| 12.2(15)T | Much of the information formerly found in the output of the debug ccsip events command is now reported in the output of the debug ccsip info and debug ccsip media commands. The debug ccsip events command now displays only the debugging information specifically related to SIP events. |

Usage Guidelines

This command previously traced all events posted to SIP SPI from all interfaces and also provided general SIP SPI information. Beginning with Cisco IOS Release 12.2(15)T, the **debug ccsip events** command displays only debugging information specifically related to SIP SPI events. Media stream and SIP SPI information is now reported in the **debug ccsip media** and **debug ccsip info** command output.



Note

This command is intended for use by Cisco technicians only.

Examples

The following is sample output from the **debug ccsip events** command for a Cisco 3660:

```
Router# debug ccsip events

SIP Call events tracing is enabled
Router#
Nov 15 18:20:25.779: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
Nov 15 18:20:25.779: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Nov 15 18:20:25.783: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:25.815: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Nov 15 18:20:25.819: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:28.339: Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
Nov 15 18:20:28.339: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:50.844: Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
```

■ **debug ccsip events**

```
Nov 15 18:20:50.844: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE  
Nov 15 18:20:50.848: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_DISCONNECT
```

Related Commands

| Command | Description |
|--------------------------|-------------------------------------------------|
| debug ccsip all | Enables all SIP-related debugging. |
| debug ccsip info | Enables tracing of general SIP SPI information. |
| debug ccsip media | Enables tracing of SIP call media streams. |

debug ccsip info

To enable tracing of general service provider interface (SPI) information, use the **debug ccsip info** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip info

no debug ccsip info

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|------------------------------|
| 12.2(15)T | This command was introduced. |

Usage Guidelines

Beginning in Cisco IOS Release 12.2(15)T, the **debug ccsip info** command is a separate option that displays general SIP SPI information for debug purposes. In past releases, this output was part of the **debug ccsip events** command.



Note

This command is intended for use by Cisco technicians only.

Examples

The following is sample output from the **debug ccsip info** command for a Cisco 3660:

```
Router# debug ccsip info

SIP Call info tracing is enabled
Router#
Nov 15 18:19:22.670: ****Adding to UAC table
Nov 15 18:19:22.670: adding call id E to table
Nov 15 18:19:22.670: CCSIP-SPI-CONTROL: act_idle_call_setup
Nov 15 18:19:22.670: act_idle_call_setup:Not using Voice Class Codec
Nov 15 18:19:22.670: act_idle_call_setup: preferred_codec set[0] type :g729r8 bytes: 20
Nov 15 18:19:22.670: sipSPICopyPeerDataToCCB: From CLI: Modem NSE payload = 100,
Passthrough = 0,Modem relay = 0, Gw-Xid = 1
SPRT latency 200, SPRT Retries = 12, Dict Size = 1024
String Len = 32, Compress dir = 3
Nov 15 18:19:22.670: ****Deleting from UAC table
Nov 15 18:19:22.670: ****Adding to UAC table
Nov 15 18:19:22.670: sipSPIUsetBillingProfile: sipCallId for billing records =
20A40C3B-D92C11D5-8015E1CC-C91F3F10@12.18.195.49
Nov 15 18:19:22.674: CCSIP-SPI-CONTROL: act_idle_connection_created
Nov 15 18:19:22.674: CCSIP-SPI-CONTROL: act_idle_connection_created: Connid(1) created to
172.18.193.190:5060, local_port 56981
Nov 15 18:19:22.674: CCSIP-SPI-CONTROL: sipSPIOutgoingCallSDP
Nov 15 18:19:22.674: convert_codec_bytes_to_ptime: Values :Codec: g729r8 codecbytes :20,
ptime: 10
Nov 15 18:19:22.674: sip_generate_sdp_xcaps_list: Modem Relay disabled. X-cap not needed
```

```

Nov 15 18:19:22.674: sipSPIAddLocalContact
Nov 15 18:19:22.674: sip_stats_method
Nov 15 18:19:22.690: HandleUdpSocketReads :Msg enqueued for SPI with IPAddr:
172.18.193.190:5060
Nov 15 18:19:22.690: CCSIP-SPI-CONTROL: act_sentininvite_new_message
Nov 15 18:19:22.690: CCSIP-SPI-CONTROL: sipSPICheckResponse
Nov 15 18:19:22.690: sip_stats_status_code
Nov 15 18:19:22.690: Roundtrip delay 16 milliseconds for method INVITE
Nov 15 18:19:22.706: HandleUdpSocketReads :Msg enqueued for SPI with IPAddr:
172.18.193.190:5060
Nov 15 18:19:22.706: CCSIP-SPI-CONTROL: act_recdproc_new_message
Nov 15 18:19:22.706: CCSIP-SPI-CONTROL: sipSPICheckResponse
Nov 15 18:19:22.706: sip_stats_status_code
Nov 15 18:19:22.706: Roundtrip delay 32 milliseconds for method INVITE
Nov 15 18:19:22.706: sipSPIGetSdpBody : Parse incoming session description
Nov 15 18:19:22.706: HandleSIPlxxSessionProgress: Content-Disposition received in 18x
response:session;handling=required
Nov 15 18:19:22.706: sipSPIDoMediaNegotiation: number of m lines is 1
Nov 15 18:19:22.706: sipSPIDoAudioNegotiation: Codec (g729r8) Negotiation Successful on
Static Payload
Nov 15 18:19:22.706: sipSPIDoPtimeNegotiation: One ptime attribute found - value:10
Nov 15 18:19:22.706: convert_ptime_to_codec_bytes: Values :Codec: g729r8 ptime :10,
codecbytes: 20
Nov 15 18:19:22.710: convert_codec_bytes_to_ptime: Values :Codec: g729r8 codecbytes :20,
ptime: 10
Nov 15 18:19:22.710: sipSPIDoDTMFRelayNegotiation: m-line index 1
Nov 15 18:19:22.710: sipSPIDoDTMFRelayNegotiation: Requested DTMF-RELAY option(s) not
found in Preferred DTMF-RELAY option list!
Nov 15 18:19:22.710: sip_sdp_get_modem_relay_cap_params:
Nov 15 18:19:22.710: sip_sdp_get_modem_relay_cap_params: NSE payload from X-cap = 0
Nov 15 18:19:22.710: sip_do_nse_negotiation: NSE Payload 100 found in SDP
Nov 15 18:19:22.710: sip_do_nse_negotiation: Remote NSE payload = local one = 100, Use it
Nov 15 18:19:22.710: sip_select_modem_relay_params: X-tmr not present in SDP. Disable
modem relay
Nov 15 18:19:22.710: sipSPIDoQoSNegotiation - SDP body with media description
Nov 15 18:19:22.710: ccsip_process_response_contact_record_route
Nov 15 18:19:22.710: CCSIP-SPI-CONTROL: ccsip_bridge: confID = 4, srcCallID = 14,
dstCallID = 13
Nov 15 18:19:22.710: sipSPIUpdateCcCallIds: old src/dest ccCallids: -1/-1, new src/dest
ccCallids: 14/13
Nov 15 18:19:22.710: sipSPIUpdateCcCallIds: old streamcallid=-1, new streamcallid=14
Nov 15 18:19:22.710: CCSIP-SPI-CONTROL: ccsip_caps_ind
Nov 15 18:19:22.710: ccsip_get_rtcp_session_parameters: CURRENT VALUES: stream_callid=14,
current_seq_num=0x1B1B
Nov 15 18:19:22.710: ccsip_get_rtcp_session_parameters: NEW VALUES: stream_callid=14,
current_seq_num=0x180C
Nov 15 18:19:22.710: ccsip_caps_ind: Load DSP with negotiated codec : g729r8, Bytes=20
Nov 15 18:19:22.710: ccsip_caps_ind: set forking flag to 0x0
Nov 15 18:19:22.710: sipSPISetDTMFRelayMode: set DSP for dtmf-relay =
CC_CAP_DTMF_RELAY_INBAND_VOICE_AND_OOB
Nov 15 18:19:22.710: sip_set_modem_caps: Negotiation already Done. Set negotiated Modem
caps
Nov 15 18:19:22.710: sip_set_modem_caps: Modem Relay & Passthru both disabled
Nov 15 18:19:22.710: sip_set_modem_caps: nse payload = 100, ptru mode = 0, ptru-codec=0,
redundancy=0, xid=0, relay=0, sprt-retry=12, latecnycy=200, compres-dir=3, dict=1024,
strnlen=32
Nov 15 18:19:22.710: ccsip_caps_ind: Load DSP with codec : g729r8, Bytes=20
Nov 15 18:19:22.710: CCSIP-SPI-CONTROL: ccsip_caps_ack
Nov 15 18:19:22.710: ccsip_caps_ack: set forking flag to 0x60FD1EAC
Nov 15 18:19:22.710: CCSIP-SPI-CONTROL: act_recdproc_connection_created
Nov 15 18:19:22.710: CCSIP-SPI-CONTROL: sipSPICheckSocketConnection: Connid(2) created to
172.18.193.190:5060, local_port 51663
Nov 15 18:19:22.714: sip_stats_method

```

```
Nov 15 18:19:22.722: HandleUdpSocketReads :Msg enqueued for SPI with IPAddr:
172.18.193.190:5060
Nov 15 18:19:22.722: CCSIP-SPI-CONTROL: act_recdproc_new_message
Nov 15 18:19:22.722: CCSIP-SPI-CONTROL: sipSPICheckResponse
Nov 15 18:19:22.722: sip_stats_status_code
Nov 15 18:19:22.722: Roundtrip delay 48 milliseconds for method PRACK
Nov 15 18:19:24.706: HandleUdpSocketReads :Msg enqueued for SPI with IPAddr:
172.18.193.190:5060
Nov 15 18:19:24.706: CCSIP-SPI-CONTROL: act_recdproc_new_message
Nov 15 18:19:24.706: CCSIP-SPI-CONTROL: sipSPICheckResponse
Nov 15 18:19:24.706: sip_stats_status_code
Nov 15 18:19:24.706: Roundtrip delay 2032 milliseconds for method PRACK
Nov 15 18:19:24.706: sipSPIGetSdpBody : Parse incoming session description
Nov 15 18:19:24.710: CCSIP-SPI-CONTROL: sipSPIUACSessionTimer
Nov 15 18:19:24.710: CCSIP-SPI-CONTROL: act_recdproc_continue_200_processing
Nov 15 18:19:24.710: CCSIP-SPI-CONTROL: act_recdproc_continue_200_processing: *** This ccb
is the parent
Nov 15 18:19:24.710: sipSPICompareRespMediaInfo
Nov 15 18:19:24.710: sipSPIDoMediaNegotiation: number of m lines is 1
Nov 15 18:19:24.710: sipSPIDoAudioNegotiation: Codec (g729r8) Negotiation Successful on
Static Payload
Nov 15 18:19:24.710: sipSPIDoPtimeNegotiation: One ptime attribute found - value:10
Nov 15 18:19:24.710: convert_ptime_to_codec_bytes: Values :Codec: g729r8 ptime :10,
codecbytes: 20
Nov 15 18:19:24.710: convert_codec_bytes_to_ptime: Values :Codec: g729r8 codecbytes :20,
ptime: 10
Nov 15 18:19:24.710: sipSPIDoDTMFRelayNegotiation: m-line index 1
Nov 15 18:19:24.710: sipSPIDoDTMFRelayNegotiation: Requested DTMF-RELAY option(s) not
found in Preferred DTMF-RELAY option list!
Nov 15 18:19:24.710: sip_sdp_get_modem_relay_cap_params:
Nov 15 18:19:24.710: sip_sdp_get_modem_relay_cap_params: NSE payload from X-cap = 0
Nov 15 18:19:24.710: sip_do_nse_negotiation: NSE Payload 100 found in SDP
Nov 15 18:19:24.710: sip_do_nse_negotiation: Remote NSE payload = local one = 100, Use it
Nov 15 18:19:24.710: sip_select_modem_relay_params: X-tmr not present in SDP. Disable
modem relay
Nov 15 18:19:24.710: sipSPIProcessMediaChanges
Nov 15 18:19:24.710: ccsip_process_response_contact_record_route
Nov 15 18:19:24.710: CCSIP-SPI-CONTROL: sipSPIProcess200Kforinvite
Nov 15 18:19:24.710: sip_stats_method
Nov 15 18:19:24.710: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
Nov 15 18:19:37.479: HandleUdpSocketReads :Msg enqueued for SPI with IPAddr:
172.18.193.190:52180
Nov 15 18:19:37.483: ****Found CCB in UAC table
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: act_active_new_message
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: sact_active_new_message_request
Nov 15 18:19:37.483: sip_stats_method
Nov 15 18:19:37.483: sip_stats_status_code
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: sipSPIInitiateCallDisconnect : Initiate call
disconnect(16) for outgoing call
Nov 15 18:19:37.483: udpsock_close_connect: Socket fd: 2 closed for connid 2 with remote
port: 5060
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: act_disconnecting_disconnect
Nov 15 18:19:37.483: CCSIP-SPI-CONTROL: sipSPICallCleanup
Nov 15 18:19:37.483: sipSPIIcpifUpdate :CallState: 4 Payout: 10230 DiscTime:1745148
ConnTime 1743871
Nov 15 18:19:37.483: ****Deleting from UAC table
Nov 15 18:19:37.483: Removing call id E
Nov 15 18:19:37.483: freeing ccb 63330954
```

Related Commands

| Command | Description |
|---------------------------|---------------------------------------------------------|
| debug ccsip all | Enables all SIP-related debugging. |
| debug ccsip events | Enables tracing of events that are specific to SIP SPI. |
| debug ccsip media | Enables tracing of SIP call media streams. |

debug ccsip media

To enable tracing of SIP call media streams, use the **debug ccsip media** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip media

no debug ccsip media

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

| Release | Modification |
|-----------|------------------------------|
| 12.2(15)T | This command was introduced. |

Usage Guidelines

Beginning in Cisco IOS Release 12.2(15)T, the **debug ccsip media** command is a separate option that displays debugging information specific to SIP media stream processing. In past releases, this output was part of the **debug ccsip events** command.



Note

This command is intended for use by Cisco technicians only.

Examples

The following is sample output from the **debug ccsip media** command for a Cisco 3660:

```
Router# debug ccsip media

SIP Call media tracing is enabled
Router#
Nov 15 18:19:53.835: sipSPISetMediaSrcAddr: media src addr for stream 1 = 172.18.195.49
Nov 15 18:19:53.835: sipSPIReserveRtpPort: reserved port 16500 for stream 1
Nov 15 18:19:53.867: sipSPIReplaceSDP
Nov 15 18:19:53.871: sipSPICopySdpInfo
Nov 15 18:19:53.871: sipSPIUpdCallWithSdpInfo:
Preferred Codec : g729r8, bytes :20
Preferred DTMF relay : inband-voice
Preferred NTE payload : 101
Early Media : No
Delayed Media : No
Bridge Done : No
New Media : No
DSP DNLD Reqd : No
Nov 15 18:19:53.871: sipSPISetMediaSrcAddr: media src addr for stream 1 = 172.18.195.49
Nov 15 18:19:53.871: sipSPIUpdCallWithSdpInfo:
M-line Index : 1
State : STREAM_ADDING (3)
Callid : -1
Negotiated Codec : g729r8, bytes :20
Negotiated DTMF relay : inband-voice
```

```

Negotiated NTE payload : 0
Media Srce Addr/Port : 172.18.195.49:16500
Media Dest Addr/Port : 172.18.193.190:19148
Nov 15 18:19:53.871: sipSPIProcessRtpSessions
Nov 15 18:19:53.871: sipSPIAddStream: Adding stream 1 (callid 16) to the VOIP RTP library
Nov 15 18:19:53.871: sipSPISetMediaSrcAddr: media src addr for stream 1 = 172.18.195.49
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: for m-line 1
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: rtcp_session info
laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148
Nov 15 18:19:53.871: sipSPIUpdateRtcpSession: No rtp session, creating a new one
Nov 15 18:19:53.871: sipSPISetStreamInfo: num_streams = 1
Nov 15 18:19:53.871: sipSPISetStreamInfo: adding stream type 0 from mline 1
Nov 15 18:19:53.871: sipSPISetStreamInfo: caps.stream_count=1,
caps.stream[0].stream_type=0x1, caps.stream_list.xmitFunc=voip_rtp_xmit,
caps.stream_list.context=0x634F1F2C (gccb)
Nov 15 18:19:55.555: sipSPICompareSDP
Nov 15 18:19:55.555: sipSPICompareStreams: stream 1 dest_port: old=19148 new=19148
Nov 15 18:19:55.555: sipSPICompareStreams: Flags set for stream 1: RTP_CHANGE=No
CAPS_CHANGE=No
Nov 15 18:19:55.555: sipSPICompareSDP: Flags set for call: NEW_MEDIA=No DSPDNLD_REQD=No
Nov 15 18:19:55.555: sipSPIReplaceSDP
Nov 15 18:19:55.555: sipSPICopySdpInfo
Nov 15 18:19:55.555: sipSPIUpdCallWithSdpInfo:
Preferred Codec : g729r8, bytes :20
Preferred DTMF relay : inband-voice
Preferred NTE payload : 101
Early Media : No
Delayed Media : No
Bridge Done : Yes
New Media : No
DSP DNLD Reqd : No
Nov 15 18:19:55.555: sipSPISetMediaSrcAddr: media src addr for stream 1 = 172.18.195.49
Nov 15 18:19:55.555: sipSPIUpdCallWithSdpInfo:
M-line Index : 1
State : STREAM_ACTIVE (3)
Callid : 16
Negotiated Codec : g729r8, bytes :20
Negotiated DTMF relay : inband-voice
Negotiated NTE payload : 0
Media Srce Addr/Port : 172.18.195.49:16500
Media Dest Addr/Port : 172.18.193.190:19148

```

Related Commands

| Command | Description |
|---------------------------|---------------------------------------------------------|
| debug ccsip all | Enables all SIP-related debugging. |
| debug ccsip events | Enables tracing of events that are specific to SIP SPI. |
| debug ccsip info | Enables tracing of general SIP SPI events. |

debug condition interface

To limit output for some debugging commands based on the interface, use the **debug condition interface** command in privileged EXEC mode. The **no** form of this command removes the interface condition and resets the interface so that it must be triggered by a condition.

debug condition interface {*interface* | **all**}

no debug condition interface {*interface* | **all**}

Syntax Description

| | |
|------------------|--------------------------------|
| <i>interface</i> | The interface type and number. |
| all | Displays all interfaces. |

Defaults

All debug messages for enabled debugging commands are displayed.

Usage Guidelines

Use this command to restrict the debug output for some commands to output based on its related interface. When you enter this command, debugging output is turned off for all interfaces except the specified interface. In addition, this command enables debugging output for conditional debugging events. Messages are displayed as different interfaces meet specific conditions.

The **no** form of the command has two functions:

- It disables the **debug condition interface** command for the specified interface. Output is no longer generated for the interface, assuming that the interface meets no other conditions. If the interface meets other active conditions, as set by another **debug condition** command, debugging output will still be generated for the interface.
- The command also resets the debugging trigger on the interface. If some other **debug condition** command has been enabled, this command resets the trigger on the interface. Output is stopped for that interface until the condition is met on the interface.

You will be asked for confirmation before removing the last condition or all conditions.

Not all debugging output is affected by the **debug condition** command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions. The commands that are affected by the **debug condition** commands are generally related to dial access functions, where a large amount of output is expected. Output from the following commands is controlled by the **debug condition** command:

- **debug aaa** {**accounting** | **authorization** | **authentication**}
- **debug dialer** {**events** | **packets**}
- **debug isdn** {**q921** | **q931**}
- **debug modem** {**oob** | **trace**}
- **debug ppp** {**all** | **authentication** | **chap** | **error** | **negotiation** | **multilink events** | **packet**}

Examples

In this example, only **debug** command output related to serial interface 1 is displayed. The condition identifier for this command is 1.

```
Router# debug condition interface serial1
```

```
Condition 1 set
```

Related Commands

| Command | Description |
|---------------------------------|-------------------------------------------------------------------------|
| debug condition | Limits output for some debugging commands based on specific conditions. |

debug confmodem

To display information associated with the discovery and configuration of the modem attached to the router, use the **debug confmodem** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug confmodem

no debug confmodem

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug confmodem** command is used in debugging configurations that use the **modem autoconfig** command.

Examples

The following is sample output from the **debug confmodem** command. In the first three lines, the router is searching for a speed at which it can communicate with the modem. The remaining lines show the actual sending of the modem command.

```
Router# debug confmodem

TTY4:detection speed(115200) response -----
TTY4:detection speed(57600) response -----
TTY4:detection speed(38400) response ---OK---
TTY4:Modem command: --AT&F&C1&D2S180=3S190=1S0=1--
TTY4: Modem configuration succeeded
TTY4: Done with modem configuration
```

debug conn

To display information from the connection manager, time-division multiplexing (TDM) and digital signal processor (DSP) clients, use the **debug conn** command in privileged EXEC mode. To disable debug mode, use the **no** form of the command.

debug conn

no debug conn

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|------------------------------------------------------------------------------------------------------------|
| | 12.1(5)XM | This command is supported on Cisco 3600 series routers. |
| | 12.2(4)T | This command is supported on Cisco 2600 series routers and was integrated into Cisco IOS Release 12.2(4)T. |

Examples The following example shows connection manager debugging output:

```
Router# debug conn
Connection Manager debugging is on

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.

Router(config)# connect conn1 t1 3/0 1 t1 4/0 1

Router(config-tdm-conn)# exit
*Mar 6 18:30:59:%CONN TDM:Segment attached to dsx1
*Mar 6 18:30:59:%CONN TDM:Parsed segment 1
*Mar 6 18:30:59:%CONN TDM:Segment attached to dsx1
*Mar 6 18:30:59:%CONN TDM:Parsed segment 2
*Mar 6 18:30:59:%CONN:Creating new connection
Router(config)#
*Mar 6 18:31:01:%CONN TDM:Interwork Segments
*Mar 6 18:31:01:%CONN TDM:Init Segment @ 61C26980
*Mar 6 18:31:01:%CONN TDM:Init Segment @ 61C26A44
*Mar 6 18:31:01:%CONN TDM:Activating Segment @ 61C26980
*Mar 6 18:31:01:%CONN:Segment alarms for conn conn1 are 2
*Mar 6 18:31:01:%CONN TDM:Activating Segment @ 61C26A44
*Mar 6 18:31:01:%CONN:Segment alarms for conn conn1 are 0
*Mar 6 18:31:01:%CONN TDM:Connecting Segments
*Mar 6 18:31:01:%CONN TDM:MAKING CONNECTION
*Mar 6 18:31:01:%CONN:cm_activate_connection, stat = 5
Router(config)#
```

debug cops

To display a one-line summary of each COPS message sent from and received by the router, use the **debug cops** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cops [detail]

no debug cops [detail]

Syntax Description

| | |
|---------------|-----------------------------------------------------------------------------------------------------|
| detail | (Optional) Displays additional debug information, including the contents of COPS and RSVP messages. |
|---------------|-----------------------------------------------------------------------------------------------------|

Defaults

COPS process debugging is not enabled.

Command History

| Release | Modification |
|----------|------------------------------|
| 12.1(1)T | This command was introduced. |

Usage Guidelines

To generate a complete record of the policy process, enter this command and, after entering a carriage return, enter the additional command **debug ip rsvp policy**.

Examples

This first example displays the one-line COPS message summaries, as the router goes through six different events.

```
Router# debug cops
COPS debugging is on
```

Event 1

The router becomes configured to communicate with a policy server:

```
Router# configure terminal

Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# ip rsvp policy cops servers 2.0.0.1
Router(config)#
15:13:45:COPS: Opened TCP connection to 2.0.0.1/3288
15:13:45:COPS: ** SENDING MESSAGE **
15:13:45:COPS OPN message, Client-type:1, Length:28. Handle:[NONE]
15:13:45:COPS: ** RECEIVED MESSAGE **
15:13:45:COPS CAT message, Client-type:1, Length:16. Handle:[NONE]
Router(config)#
```

Event 2

The router receives a PATH message:

```
15:13:53:COPS:** SENDING MESSAGE **
15:13:53:COPS REQ message, Client-type:1, Length:216. Handle:[ 00 00 04 01]
15:13:53:COPS:** RECEIVED MESSAGE **
15:13:53:COPS DEC message, Client-type:1, Length:104. Handle:[ 00 00 04 01]
Router(config)#
```

Event 3

The router receives a unicast FF RESV message:

```
15:14:00:COPS:** SENDING MESSAGE **
15:14:00:COPS REQ message, Client-type:1, Length:148. Handle:[ 00 00 05 01]
15:14:00:COPS:** RECEIVED MESSAGE **
15:14:00:COPS DEC message, Client-type:1, Length:64. Handle:[ 00 00 05 01]
15:14:00:COPS:** SENDING MESSAGE **
15:14:00:COPS RPT message, Client-type:1, Length:24. Handle:[ 00 00 05 01]
Router(config)#
```

Event 4

The router receives a RESV tear:

```
15:14:06:COPS:** SENDING MESSAGE **
15:14:06:COPS DRQ message, Client-type:1, Length:24. Handle:[ 00 00 05 01]
Router(config)#
```

Event 5

The router receives a PATH tear:

```
15:14:11:COPS:** SENDING MESSAGE **
15:14:11:COPS DRQ message, Client-type:1, Length:24. Handle:[ 00 00 04 01]
Router(config)#
```

Event 6

The router gets configured to cease communicating with the policy server:

```
Router(config)# no ip rsvp policy cops servers
15:14:23:COPS:** SENDING MESSAGE **
15:14:23:COPS CC message, Client-type:1, Length:16. Handle:[NONE]
15:14:23:COPS:Closed TCP connection to 2.0.0.1/3288
Router(config)#
```

This second example uses the **detail** keyword to display the contents of the COPS and RSVP messages, and additional debugging information:

```
Router# debug cops detail
```

```
COPS debugging is on
```

```
02:13:29:COPS:** SENDING MESSAGE **
  COPS HEADER:Version 1, Flags 0, Opcode 1 (REQ), Client-type:1, Length:216
  HANDLE (1/1) object. Length:8.    00 00 21 01
  CONTEXT (2/1) object. Length:8.   R-type:5.    M-type:1
  IN_IF (3/1) object. Length:12.   Address:10.1.2.1.   If_index:4
  OUT_IF (4/1) object. Length:12.  Address:10.33.0.1. If_index:3
  CLIENT SI (9/1) object. Length:168. CSI data:
02:13:29: SESSION                type 1 length 12:
02:13:29:   Destination 10.33.0.1, Protocol_Id 17, Don't Police , DstPort 44
02:13:29: HOP                    type 1 length 12:0A010201
02:13:29:                        :00000000
02:13:29: TIME_VALUES            type 1 length 8 :00007530
02:13:29: SENDER_TEMPLATE        type 1 length 12:
02:13:29:   Source 10.31.0.1, udp_source_port 44
02:13:29: SENDER_TSPEC           type 2 length 36:
02:13:29:   version=0, length in words=7
02:13:29:   Token bucket fragment (service_id=1, length=6 words
02:13:29:     parameter id=127, flags=0, parameter length=5
02:13:29:     average rate=1250 bytes/sec, burst depth=10000 bytes
02:13:29:     peak rate =1250000 bytes/sec
02:13:29:     min unit=0 bytes, max unit=1514 bytes
02:13:29: ADSPEC                 type 2 length 84:
02:13:29: version=0 length in words=19
02:13:29: General Parameters break bit=0 service length=8
02:13:29:                        IS Hops:1
02:13:29:   Minimum Path Bandwidth (bytes/sec):1250000
02:13:29:   Path Latency (microseconds):0
02:13:29:   Path MTU:1500
02:13:29: Guaranteed Service break bit=0 service length=8
02:13:29:   Path Delay (microseconds):192000
02:13:29:   Path Jitter (microseconds):1200
02:13:29:   Path delay since shaping (microseconds):192000
02:13:29:   Path Jitter since shaping (microseconds):1200
02:13:29: Controlled Load Service break bit=0 service length=0
02:13:29:COPS:Sent 216 bytes on socket,
02:13:29:COPS:Message event!
02:13:29:COPS:State of TCP is 4
02:13:29:In read function
02:13:29:COPS:Read block of 96 bytes, num=104 (len=104)
02:13:29:COPS:** RECEIVED MESSAGE **
  COPS HEADER:Version 1, Flags 1, Opcode 2 (DEC), Client-type:1, Length:104
  HANDLE (1/1) object. Length:8.    00 00 21 01
  CONTEXT (2/1) object. Length:8.   R-type:1.    M-type:1
  DECISION (6/1) object. Length:8.  COMMAND cmd:1, flags:0
  DECISION (6/3) object. Length:56. REPLACEMENT 00 10 0E 01 61 62 63 64 65 66 67
  68 69 6A 6B 6C 00 24 0C 02 00
  00 00 07 01 00 00 06 7F 00 00 05 44 9C 40 00 46 1C 40 00 49 98
  96 80 00 00 00 C8 00 00 01 C8
  CONTEXT (2/1) object. Length:8.   R-type:4.    M-type:1
  DECISION (6/1) object. Length:8.  COMMAND cmd:1, flags:0

02:13:29:Notifying client (callback code 2)
02:13:29:COPS:** SENDING MESSAGE **
  COPS HEADER:Version 1, Flags 1, Opcode 3 (RPT), Client-type:1, Length:24
  HANDLE (1/1) object. Length:8.    00 00 21 01
  REPORT (12/1) object. Length:8.   REPORT type COMMIT (1)
```

■ **debug cops**

```
02:13:29:COPS:Sent 24 bytes on socket,  
02:13:29:Timer for connection entry is zero
```

To see an example where the **debug cops** command is used along with the **debug ip rsvp policy** command, refer to the second example of the **debug ip rsvp policy** command.

Related Commands

| Command | Description |
|-----------------------------|-----------------------------------------------------|
| debug ip rsvp policy | Displays debug messages for RSVP policy processing. |

debug cot

To display information about the COT functionality, use the **debug cot** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cot {api | dsp | queue | detail}

no debug cot {api | dsp | queue | detail}

Syntax Description

| | |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| api | Displays information about the COT Application Program Interface (API). |
| dsp | Displays information related to the COT/DSP interface. Typical DSP functions include data modems, voice codecs, fax modems and codecs, and low-level signaling such as CAS/R2. |
| queue | Display information related to the COT internal queue. |
| detail | Display information about COT internal detail; summary of the debug cot api , debug cot dsp , and debug cot queue commands. |

Command History

| Release | Modification |
|---------|------------------------------|
| 11.3(7) | This command was introduced. |

Examples

The following is sample output of the **debug cot api** command.

Figure 1 Sample debug cot api Command Output

```
Router# debug cot api

COT API debugging is on
08:29:55: cot_request_handler(): CDB@0x60DEDE14, req(COT_CHECK_TONE_ON):
08:29:55:      shelf 0 slot 0 appl_no 1 ds0 1
08:29:55:      freqTX 2010 freqRX 1780 key 0xFFF1 duration 60000
```

[Table 41](#) describes the significant fields in the display.

Table 41 debug cot api Field Descriptions

| Field | Description |
|----------|----------------------------------------------------------------------------------------------|
| CDB | Internal controller information. |
| req | Type of COT operation requested. |
| shelf | Shelf ID of the COT operation request. |
| slot | Designates the slot number, 1 to 4. |
| appl-no | Hardware unit that provides the external interface connections from a router to the network. |
| ds0 | Number of the COT operation request. |
| key | COT operation identifier. |
| duration | Timeout duration of the COT operation. |

Table 41 *debug cot api Field Descriptions (continued)*

| Field | Description |
|--------|------------------------------------|
| freqTX | Requested transmit tone frequency. |
| freqRX | Requested receive tone frequency. |

The following is sample output of the **debug cot dsp** command.

Figure 2 *Sample debug cot dsp Command Output*

```
Router# debug cot dsp

Router#
00:10:42:COT:DSP (1/1) Allocated
00:10:43:In cot_callback
00:10:43: returned key 0xFFFF1, status = 0
00:10:43:COT:Received DSP Q Event
00:10:43:COT:DSP (1/1) Done
00:10:43:COT:DSP (1/1) De-allocated
```

[Table 42](#) describes the significant fields in the display.

Table 42 *debug cot dsp Field Descriptions*

| Field | Description |
|------------------------|----------------------------------------------------------------------------------|
| DSP (1/1) Allocated | Slot and port of the DSP allocated for the COT operation. |
| Received DSP Q Event | Indicates the COT subsystem received an event from the DSP. |
| DSP (1/1) Done | Slot and port of the DSP transitioning to IDLE state. |
| DSP (1/1) De-allocated | Slot and port of the DSP de-allocated after the completion of the COT operation. |

The following is sample output of the **debug cot queue** command.

```
Router# debug cot queue

Router#
00:11:26:COT(0x60EBB48C):Adding new request (0x61123DBC) to In
Progress Q
00:11:26:COT(0x60EBB48C):Adding COT(0x61123DBC) to the Q head
00:11:27:In cot_callback
00:11:27: returned key 0xFFFF1, status = 0
```

Table 43 describes the significant fields in the display.

Table 43 *debug cot api Field Descriptions*

| Field | Description |
|--------------------|---------------------------------------|
| COT | Internal COT operation request. |
| Adding new request | Internal COT operation request queue. |

The following is sample output of the **debug cot detail** command.

Router# **debug cot detail**

```
Router#
00:04:57:cot_request_handler():CDB@0x60EBB48C, req(COT_CHECK_TONE_ON):

00:04:57:    shelf 0 slot 0 appl_no 1 ds0 1
00:04:57:    freqTX 1780 freqRX 2010 key 0xFFFF1 duration 1000

00:04:57:COT:DSP (1/0) Allocated
00:04:57:COT:Request Transition to COT_WAIT_TD_ON
00:04:57:COT(0x60EBB48C):Adding new request (0x61123DBC) to In
Progress Q
00:04:57:COT(0x60EBB48C):Adding COT(0x61123DBC) to the Q head
00:04:57:COT:Start Duration Timer for Check Tone Request
00:04:58:COT:Received Timer Event
00:04:58:COT:T24 Timer Expired
00:04:58:COT Request@ 0x61123DBC, CDB@ 0x60EBB48C, Params@0x61123E08
00:04:58:  request type = COT_CHECK_TONE_ON
00:04:58:  shelf 0 slot 0 appl_no 1 ds0 1
00:04:58:  duration 1000 key FFF1 freqTx 1780 freqRx 2010
00:04:58:  state COT_WAIT_TD_ON_CT
00:04:58:  event_proc(0x6093B55C)

00:04:58:Invoke NI2 callback to inform COT request status
00:04:58:In cot_callback
00:04:58:  returned key 0xFFFF1, status = 0
00:04:58:Return from NI2 callback
00:04:58:COT:Request Transition to IDLE
00:04:58:COT:Received DSP Q Event
00:04:58:COT:DSP (1/0) Done
00:04:58:COT:DSP (1/0) De-allocated
```

Because the **debug cot detail** command is a summary of the **debug cot api**, **debug cot dsp**, and **debug cot queue** commands, the field descriptions are the same.

debug cpp event

To display general Combinet Proprietary Protocol (CPP) events, use the **debug cpp event** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cpp event

no debug cpp event

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debug cpp event** command displays events such as CPP sequencing, group creation, and keepalives.

Examples

One or more of the messages in [Table 44](#) appear when you use the **debug cpp event** command. Each message begins with the short name of the interface the event occurred on (for example, SERIAL0:1 or BRI0:1) and might contain one or more packet sequence numbers or remote site names.

Table 44 *debug cpp event Messages*

| Message | Description |
|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BRI0:1: negotiation complete | Call was set up on the interface (in this example, BRI0:1). |
| BRI0:1: negotiation timed out | Call timed out. |
| BRI0:1: sending negotiation packet | Negotiation packet was sent to set up the call. |
| BRI0:1: out of sequence packet - got 10, range 1 8 | Packet was received that was out of sequence. The first number displayed in the message is the sequence number received, and the following numbers are the range of valid sequence numbers. |
| BRI0:1: Sequence timer expired - Lost 11 Trying sequence 12 | Timer expired before the packet was received. The first number displayed in the message is the sequence number of the packet that was lost, and the second number is the next sequence number. |
| BRI0:1: Line Integrity Violation | Router fails to maintain keepalives. |
| BRI0:1: create cpp group ber19 destroyed cpp group ber19 | Dialer group is created on the remote site (in this example, ber19). |

Related Commands

| Command | Description |
|---------------------------------------|----------------------------------|
| debug cpp negotiation | Displays CPP negotiation events. |
| debug cpp packet | Displays CPP packets. |

debug cpp negotiation

To display Combinet Proprietary Protocol (CPP) negotiation events, use the **debug cpp negotiation** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cpp negotiation

no debug cpp negotiation

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debug cpp negotiation** command displays events such as the type of packet and packet size being sent.

Examples

The following is sample output from the **debug cpp negotiation** command. In this example, a sample connection is shown.

```
Router# debug cpp negotiation

%LINK-3-UPDOWN: Interface BRI0: B-Channel 2, changed state to down
%LINK-3-UPDOWN: Interface BRI0, changed state to up
%SYS-5-CONFIG_I: Configured from console by console
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
BR0:1:(I) NEG packet - len 77
  attempting proto:2
  ether id:0040.f902.c7b4
  port 1 number:5559876
  port 2 number:5559876
  origination port:1
  remote name:berl9
  password is correct
```

[Table 45](#) describes the significant fields in the display.

Table 45 Debug CPP Negotiation Field Descriptions

| Field | Description |
|-------------------------------|--------------------------------------------------------|
| BR0:1 (I) NEG packet - len 77 | Interface name, packet type, and packet size. |
| attempting proto: | CPP protocol type. |
| ether id: | Ethernet address of the destination router. |
| port 1 number: | ISDN phone number of remote B channel #1. |
| port 2 number: | ISDN phone number of remote B channel #2. |
| origination port: | B channel 1 or 2 called. |
| remote name: | Remote site name to which this call is connecting. |
| password is correct | Password is accepted so the connection is established. |

■ debug cpp negotiation

| Related Commands | Command | Description |
|------------------|----------------------------------|---------------------------------------------------|
| | debug cot | Displays information about the COT functionality. |
| | debug cpp packet | Displays CPP packets. |

debug cpp packet

To display Combinet Proprietary Protocol (CPP) packets, use the **debug cpp packet** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug cpp packet

no debug cpp packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

CPP allows a router to engage in negotiation over an ISDN B channel to establish connections with a Combinet bridge.

The **debug cpp packet** command displays the hexadecimal values of the packets.

Examples

The following is sample output from the **debug cpp packet** command. This example shows the interface name, packet type, packet size, and the hexadecimal values of the packet.

```
Router# debug cpp packet

BR0:1:input packet - len 60
00 00 00 00 00 00 00 40 F9 02 C7 B4 08 01 6 00 01
08 00 06 04 00 02 00 40 F9 02 C7 B4 83 6C A1 02!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 64/66/68 ms
BR0:1 output packet - len 116
06 00 00 40 F9 02 C7 B4 00 00 0C 3E 12 3A 08 00
45 00 00 64 00 01 00 00 FF 01 72 BB 83 6C A1 01
```

Related Commands

| Command | Description |
|---------------------------------------|---------------------------------------------------|
| debug cot | Displays information about the COT functionality. |
| debug cpp negotiation | Displays CPP negotiation events. |

debug crm

To view Carrier Resource Manager (CRM) information, use the **debug crm** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crm

no debug crm

Syntax Description This command has no arguments or keywords.

Defaults This command is disabled by default.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|------------------------------|
| | 12.2(11)T | This command was introduced. |

Usage Guidelines Disable console logging and use buffered logging before using the **debug crm** command. Using the **debug crm** command generates a large volume of debugs, which can affect router performance.

Examples A sample output of the **debug crm** command is shown below.

The output shows that the route label, which will be either a trunk group label or carrier ID, is att1. Mask 1 indicates that it is an incoming voice update. Count type 1 indicates the number of voice calls is being incremented.

```
00:17:53: crm_call_update:route label att1, mask 1, count type 1
00:17:53: crm_call_update:for att1
00:17:53: route label type 1
00:17:53: event type 1
00:17:53: reason for event 0
00:17:53: max capacity mask 0
00:17:53: current capacity mask 1
```

Table 46 provides an alphabetical listing of the **debug crm** command fields and a description of each field.

Table 46 *debug crm Field Descriptions*

| Field | Description |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| count type | Indicates whether the number of voice calls is being incremented or decremented. 1 = Incremented -1 = Decrement |
| current capacity mask | Indicates type of current capacity update from CRM to SPI. |
| event type | 0 = Update all carriers 1 = Update single carrier |
| mask | Mask for CRM call update. |
| max capacity mask | Indicates type of maximum capacity update from CRM to SPI. |
| reason for event | Reason for this event being sent: 0 = Current capacity update 1 = Max capacity update 2 = Both capacity update 3 = Delete carrier |
| route label | Either the trunk group label or carrier id. |
| route label type | Indicates the type of trunk. 0 = Invalid 1 = TDM 2 = VOIP H323 3 = VOIP SIP 4 = VOIP MGCP 5 = VOIPN2P |

Related Commands

| Command | Description |
|------------------|-------------------------------------------------------------------|
| max-calls | Specifies the maximum number of calls the trunk group can handle. |

debug crypto engine

To display debug messages about crypto engines, which perform encryption and decryption, use the **debug crypto engine** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto engine

no debug crypto engine

Syntax Description

This command has no arguments or keywords.

Command History

| Release | Modification |
|---------|------------------------------|
| 12.0 | This command was introduced. |

Usage Guidelines

Use the **debug crypto engine** command to display information pertaining to the crypto engine, such as when Cisco IOS software is performing encryption or decryption operations.

The crypto engine is the actual mechanism that performs encryption and decryption. A crypto engine can be software or a hardware accelerator. Some platforms can have multiple crypto engines; therefore, the router will have multiple hardware accelerators.

Examples

The following is sample output from the **debug crypto engine** command. The first sample output shows messages from a router that successfully generates RSA keys. The second sample output shows messages from a router that decrypts the RSA key during Internet Key Exchange (IKE) negotiation.

```
Router# debug crypto engine

00:25:13:CryptoEngine0:generate key pair
00:25:13:CryptoEngine0:CRYPTO_GEN_KEY_PAIR
00:25:13:CRYPTO_ENGINE:key process suspended and continued
00:25:14:CRYPTO_ENGINE:key process suspended and continuedcr

Router# debug crypto engine

00:27:45:%SYS-5-CONFIG_I:Configured from console by console
00:27:51:CryptoEngine0:generate alg parameter
00:27:51:CRYPTO_ENGINE:Dh phase 1 status:0
00:27:51:CRYPTO_ENGINE:Dh phase 1 status:0
00:27:51:CryptoEngine0:generate alg parameter
00:27:52:CryptoEngine0:calculate pkey hmac for conn id 0
00:27:52:CryptoEngine0:create ISAKMP SKEYID for conn id 1
00:27:52:Crypto engine 0:RSA decrypt with public key
00:27:52:CryptoEngine0:CRYPTO_RSA_PUB_DECRYPT
00:27:52:CryptoEngine0:generate hmac context for conn id 1
00:27:52:CryptoEngine0:generate hmac context for conn id 1
00:27:52:Crypto engine 0:RSA encrypt with private key
```

```
00:27:52:CryptoEngine0:CRYPTO_RSA_PRIV_ENCRYPT
00:27:53:CryptoEngine0:clear dh number for conn id 1
00:27:53:CryptoEngine0:generate hmac context for conn id 1
00:27:53:validate proposal 0
00:27:53:validate proposal request 0
00:27:54:CryptoEngine0:generate hmac context for conn id 1
00:27:54:CryptoEngine0:generate hmac context for conn id 1
00:27:54:ipsec allocate flow 0
00:27:54:ipsec allocate flow 0
```

Related Commands

| Command | Description |
|--------------------------------|--------------------------|
| crypto key generate rsa | Generates RSA key pairs. |

debug crypto engine accelerator logs

To enable logging of commands and associated parameters sent from the VPN module driver to the VPN module hardware using a debug flag, use the **debug crypto engine accelerator logs** command in privileged EXEC mode.

debug crypto engine accelerator logs

no debug crypto engine accelerator logs

Syntax Description

This command has no arguments or keywords.

Defaults

The logging of commands sent from the VPN module driver to the VPN module hardware is disabled.

Command History

| Release | Modification |
|-----------|-----------------------------------------------------------------------|
| 12.1(1)XC | This command was introduced on the Cisco 1720 and Cisco 1750 routers. |

Usage Guidelines

Use the **debug crypto engine accelerator logs** command when encryption traffic is sent to the router and a problem with the encryption module is suspected.

This command is intended only for Cisco TAC personnel to collect debugging information.

Examples

The command **debug crypto engine accelerator logs** uses a debug flag to log commands and associated parameters sent from the VPN module driver to the VPN module hardware as follows:

```
Router# debug crypto engine accelerator logs

encryption module logs debugging is on
```

Related Commands

| Command | Description |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| crypto engine accelerator | Enables or disables the crypto engine accelerator if it exists. |
| show crypto engine accelerator logs | Prints information about the last 32 CGX Library packet processing commands, and associated parameters sent from the VPN module driver to the VPN module hardware. |
| show crypto engine accelerator sa-database | Prints active (in-use) entries in the platform-specific VPN module database. |
| show crypto engine configuration | Displays the Cisco IOS crypto engine of your router. |

debug crypto ipsec

To display IPsec events, use the **debug crypto ipsec** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug crypto ipsec

no debug crypto ipsec

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug crypto ipsec** command. In this example, security associations (SAs) have been successfully established.

```
Router# debug crypto ipsec
```

IPsec requests SAs between 172.21.114.123 and 172.21.114.67, on behalf of the **permit ip host 172.21.114.123 host 172.21.114.67** command. It prefers to use the transform set esp-des w/esp-md5-hmac, but it will also consider ah-sha-hmac.

```
00:24:30: IPSEC(sa_request): ,
(key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
protocol= ESP, transform= esp-des esp-md5-hmac ,
lifedur= 120s and 4608000kb,
spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
00:24:30: IPSEC(sa_request): ,
(key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
protocol= AH, transform= ah-sha-hmac ,
lifedur= 120s and 4608000kb,
spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x0.
```

IKE asks for SPIs from IPsec. For inbound security associations, IPsec controls its own SPI space.

```
00:24:34: IPSEC(key_engine): got a queue event...
00:24:34: IPSEC(spi_response): getting spi 3029740121d for SA
from 172.21.114.67 to 172.21.114.123 for prot 3
00:24:34: IPSEC(spi_response): getting spi 5250759401d for SA
from 172.21.114.67 to 172.21.114.123 for prot 2
```

IKE will ask IPsec if it accepts the SA proposal. In this case, it will be the one sent by the local IPsec in the first place:

```
00:24:34: IPSEC(validate_proposal_request): proposal part #1,
(key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
protocol= ESP, transform= esp-des esp-md5-hmac ,
lifedur= 0s and 0kb,
spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
```

After the proposal is accepted, IKE finishes the negotiations, generates the keying material, and then notifies IPsec of the new security associations (one security association for each direction).

```
00:24:35: IPSEC(key_engine): got a queue event...
```

The following output pertains to the inbound SA. The `conn_id` value references an entry in the crypto engine connection table.

```
00:24:35: IPSEC(initialize_sas): ,
(key eng. msg.) dest= 172.21.114.123, src= 172.21.114.67,
  dest_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
  src_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
  protocol= ESP, transform= esp-des esp-md5-hmac ,
  lifedur= 120s and 4608000 kb,
  spi= 0x120F043C(302974012), conn_id= 29, keysize= 0, flags= 0x4
```

The following output pertains to the outbound SA:

```
00:24:35: IPSEC(initialize_sas): ,
(key eng. msg.) src= 172.21.114.123, dest= 172.21.114.67,
  src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
  dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
  protocol= ESP, transform= esp-des esp-md5-hmac ,
  lifedur= 120s and 4608000kb,
  spi= 0x38914A4(59315364), conn_id= 30, keysize= 0, flags= 0x4
```

IPsec now installs the SA information into its SA database.

```
00:24:35: IPSEC(create_sa): sa created,
(sa) sa_dest= 172.21.114.123, sa_prot= 50,
  sa_spi= 0x120F043C(302974012),
  sa_trans= esp-des esp-md5-hmac , sa_conn_id= 29
00:24:35: IPSEC(create_sa): sa created,
(sa) sa_dest= 172.21.114.67, sa_prot= 50,
  sa_spi= 0x38914A4(59315364),
  sa_trans= esp-des esp-md5-hmac , sa_conn_id= 30
```

The following is sample output for the `debug crypto ipsec` command as seen on the peer router. In this example, IKE asks IPsec if it will accept an SA proposal. Although the peer sent two proposals, IPsec accepted the first proposal.

```
00:26:15: IPSEC(validate_proposal_request): proposal part #1,
(key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
  dest_proxy= 172.21.114.67/255.255.255.255/0/0 (type=1),
  src_proxy= 172.21.114.123/255.255.255.255/0/0 (type=1),
  protocol= ESP, transform= esp-des esp-md5-hmac ,
  lifedur= 0s and 0kb,
  spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x4
```

IKE asks for SPIs.

```
00:26:15: IPSEC(key_engine): got a queue event...
00:26:15: IPSEC(spi_response): getting spi 593153641d for SA
  from 172.21.114.123 to 172.21.114.67 for prot 3
```

IKE does the remaining processing, completing the negotiation and generating keys. It then tells IPsec about the new SAs.

```
00:26:15: IPSEC(key_engine): got a queue event...
```

The following output pertains to the inbound SA:

```
00:26:15: IPSEC(initialize_sas): ,
(key eng. msg.) dest= 172.21.114.67, src= 172.21.114.123,
  dest_proxy= 172.21.114.67/0.0.0.0/0/0 (type=1),
  src_proxy= 172.21.114.123/0.0.0.0/0/0 (type=1),
  protocol= ESP, transform= esp-des esp-md5-hmac ,
  lifedur= 120s and 4608000kb,
  spi= 0x38914A4(59315364), conn_id= 25, keysize= 0, flags= 0x4
```

The following output pertains to the outbound SA:

```
00:26:15: IPSEC(initialize_sas): ,
(key eng. msg.) src= 172.21.114.67, dest= 172.21.114.123,
  src_proxy= 172.21.114.67/0.0.0.0/0/0 (type=1),
  dest_proxy= 172.21.114.123/0.0.0.0/0/0 (type=1),
  protocol= ESP, transform= esp-des esp-md5-hmac ,
  lifedur= 120s and 4608000kb,
  spi= 0x120F043C(302974012), conn_id= 26, keysize= 0, flags= 0x4
```

IPSec now installs the SA information into its SA database:

```
00:26:15: IPSEC(create_sa): sa created,
(sa) sa_dest= 172.21.114.67, sa_prot= 50,
  sa_spi= 0x38914A4(59315364),
  sa_trans= esp-des esp-md5-hmac , sa_conn_id= 25
00:26:15: IPSEC(create_sa): sa created,
(sa) sa_dest= 172.21.114.123, sa_prot= 50,
  sa_spi= 0x120F043C(302974012),
  sa_trans= esp-des esp-md5-hmac , sa_conn_id= 26
```

debug crypto ipsec client ezvpn

To display information showing the configuration and implementation of the Voice digital signal processor (DSP) Control Message Logger feature, use the **debug crypto ipsec client ezvpn** command in privileged EXEC mode. To turn off debugging of the Voice DSP Control Message Logger feature, use the **no** form of this command.

debug crypto ipsec client ezvpn

no debug crypto ipsec client ezvpn

Syntax Description This command has no arguments or keywords.

Command Modes Privileged EXEC

| Command History | Release | Modification |
|-----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | 12.2(4)YA | This command was introduced for the Cisco 806, Cisco 826, Cisco 827, and Cisco 828 routers, the Cisco 1700 series routers, and the Cisco uBR905 and Cisco uBR925 cable access routers. |
| | 12.2(13)T | Support was added to the Cisco IOS Release 12.2 T train. |

Usage Guidelines To force the Voice DSP Control Message Logger feature to reestablish the Virtual Private Network (VPN) connections, use the **clear crypto sa** and **clear crypto isakmp** commands to delete the IP Security (IPSec) security associations and Internet Key Exchange (IKE) connections, respectively.

Examples The following example shows debugging of the Voice DSP Control Message Logger feature being turned on, as well as typical debugging messages that appear when the VPN tunnel is created:

```
Router# debug crypto ipsec client ezvpn

EzVPN debugging is on
Router#
3d17h: EZVPN: New State: READY
3d17h: EZVPN: Current State: READY
3d17h: EZVPN: Event: MODE_CONFIG_REPLY
3d17h: ezvpn_mode_config
3d17h: ezvpn_parse_mode_config_msg
3d17h: EZVPN: Attributes sent in message:
3d17h:     DNS Primary: 172.168.0.250
3d17h:     DNS Secondary: 172.168.0.251
3d17h:     NBMS/WINS Primary: 172.168.0.252
3d17h:     NBMS/WINS Secondary: 172.168.0.253
3d17h:     Default Domain: cisco.com
3d17h: EZVPN: New State: SS_OPEN
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_READY
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_READY
```

```

3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: MTU_CHANGED
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_UP
3d17h: EZVPN: New State: IPSEC_ACTIVE
3d17h: EZVPN: Current State: IPSEC_ACTIVE
3d17h: EZVPN: Event: MTU_CHANGED
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: IPSEC_ACTIVE
3d17h: EZVPN: Event: SOCKET_UP

```

The following example shows the typical display for a VPN tunnel that is reset with the **clear crypto ipsec client ezvpn** command:

```

Router# clear crypto ipsec client ezvpn

3d17h: EZVPN: Current State: READY
3d17h: EZVPN: Event: RESET
3d17h: ezvpn_reconnect_request
3d17h: ezvpn_close
3d17h: ezvpn_connect_request
3d17h: EZVPN: New State: READY
3d17h: EZVPN: Current State: READY
3d17h: EZVPN: Event: MODE_CONFIG_REPLY
3d17h: ezvpn_mode_config
3d17h: ezvpn_parse_mode_config_msg
3d17h: EZVPN: Attributes sent in message:
3d17h:      DNS Primary: 172.168.0.250
3d17h:      DNS Secondary: 172.168.0.251
3d17h:      NBMS/WINS Primary: 172.168.0.252
3d17h:      NBMS/WINS Secondary: 172.168.0.253
3d17h:      Split Tunnel List: 1
3d17h:          Address      : 172.168.0.128
3d17h:          Mask           : 255.255.255.128
3d17h:          Protocol      : 0x0
3d17h:          Source Port    : 0
3d17h:          Dest Port     : 0
3d17h:      Split Tunnel List: 2
3d17h:          Address      : 172.168.1.128
3d17h:          Mask           : 255.255.255.128
3d17h:          Protocol      : 0x0
3d17h:          Source Port    : 0
3d17h:          Dest Port     : 0
3d17h:      Default Domain: cisco.com
3d17h: ezvpn_nat_config
3d17h: EZVPN: New State: SS_OPEN
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_READY
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_READY
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: MTU_CHANGED
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: SS_OPEN
3d17h: EZVPN: Event: SOCKET_UP
3d17h: EZVPN: New State: IPSEC_ACTIVE
3d17h: EZVPN: Current State: IPSEC_ACTIVE
3d17h: EZVPN: Event: MTU_CHANGED
3d17h: EZVPN: No state change
3d17h: EZVPN: Current State: IPSEC_ACTIVE

```

```
3d17h: EZVPN: Event: SOCKET_UP
```

The following example shows the typical display for a VPN tunnel that is removed from the interface with the **no crypto ipsec client ezvpn** command:

```
Router# no crypto ipsec client ezvpn

4d16h: EZVPN: Current State: IPSEC ACTIVE
4d16h: EZVPN: Event: REMOVE INTERFACE CFG
4d16h: ezvpn_close_and_remove
4d16h: ezvpn_close
4d16h: ezvpn_remove
4d16h: EZVPN: New State: IDLE
```

Related Commands

| Command | Description |
|----------------------------|---------------------------------------------------------------------|
| debug crypto ipsec | Displays debugging messages for generic IPsec events. |
| debug crypto isakmp | Displays debugging messages for Internet Key Exchange (IKE) events. |

debug crypto isakmp

To display messages about Internet Key Exchange (IKE) events, use the **debug crypto isakmp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto isakmp aaa

no debug crypto isakmp aaa

Syntax Description

| | |
|------------|------------------------------|
| aaa | Specifies accounting events. |
|------------|------------------------------|

Command Modes

Privileged EXEC

Command History

| Release | Modifications |
|-----------|-----------------------------------|
| 11.3 T | This command was introduced. |
| 12.2(15)T | The aaa keyword was added. |

Examples

The following is sample output from the **debug crypto isakmp** command for an IKE peer that initiates an IKE negotiation.

First, IKE negotiates its own security association (SA), checking for a matching IKE policy.

```
Router# debug crypto isakmp
```

```
20:26:58: ISAKMP (8): beginning Main Mode exchange
20:26:58: ISAKMP (8): processing SA payload. message ID = 0
20:26:58: ISAKMP (8): Checking ISAKMP transform 1 against priority 10 policy
20:26:58: ISAKMP: encryption DES-CBC
20:26:58: ISAKMP: hash SHA
20:26:58: ISAKMP: default group 1
20:26:58: ISAKMP: auth pre-share
20:26:58: ISAKMP (8): atts are acceptable. Next payload is 0
```

IKE has found a matching policy. Next, the IKE SA is used by each peer to authenticate the other peer.

```
20:26:58: ISAKMP (8): SA is doing pre-shared key authentication
20:26:59: ISAKMP (8): processing KE payload. message ID = 0
20:26:59: ISAKMP (8): processing NONCE payload. message ID = 0
20:26:59: ISAKMP (8): SKEYID state generated
20:26:59: ISAKMP (8): processing ID payload. message ID = 0
20:26:59: ISAKMP (8): processing HASH payload. message ID = 0
20:26:59: ISAKMP (8): SA has been authenticated
```

Next, IKE negotiates to set up the IP Security (IPSec) SA by searching for a matching transform set.

```
20:26:59: ISAKMP (8): beginning Quick Mode exchange, M-ID of 767162845
20:26:59: ISAKMP (8): processing SA payload. message ID = 767162845
20:26:59: ISAKMP (8): Checking IPSec proposal 1
20:26:59: ISAKMP: transform 1, ESP_DES
20:26:59: ISAKMP: attributes in transform:
20:26:59: ISAKMP: encaps is 1
20:26:59: ISAKMP: SA life type in seconds
20:26:59: ISAKMP: SA life duration (basic) of 600
```

```

20:26:59: ISAKMP:      SA life type in kilobytes
20:26:59: ISAKMP:      SA life duration (VPI) of
                0x0 0x46 0x50 0x0
20:26:59: ISAKMP:      authenticator is HMAC-MD5
20:26:59: ISAKMP (8): atts are acceptable.

```

A matching IPSec transform set has been found at the two peers. Now the IPSec SA can be created (one SA is created for each direction).

```

20:26:59: ISAKMP (8): processing NONCE payload. message ID = 767162845
20:26:59: ISAKMP (8): processing ID payload. message ID = 767162845
20:26:59: ISAKMP (8): processing ID payload. message ID = 767162845
20:26:59: ISAKMP (8): Creating IPSec SAs
20:26:59:      inbound SA from 155.0.0.2 to 155.0.0.1 (proxy 155.0.0.2 to 155.0.0.1 )
20:26:59:      has spi 454886490 and conn_id 9 and flags 4
20:26:59:      lifetime of 600 seconds
20:26:59:      lifetime of 4608000 kilobytes
20:26:59:      outbound SA from 155.0.0.1      to 155.0.0.2      (proxy 155.0.0.1
to 155.0.0.2
)
20:26:59:      has spi 75506225 and conn_id 10 and flags 4
20:26:59:      lifetime of 600 seconds
20:26:59:      lifetime of 4608000 kilobytes

```

The following is sample output from the **debug crypto isakmp** command using the **aaa** keyword:

```
Router# debug crypto isakmp aaa
```

Start Example

```

01:38:55: ISAKMP AAA: Sent Accounting Message
01:38:55: ISAKMP AAA: Accounting message successful
01:38:55: ISAKMP AAA: Rx Accounting Message
01:38:55: ISAKMP AAA: Adding Client Attributes to Accounting Record
01:38:55: ISAKMP AAA: Accounting Started

```

Update Example

```

01:09:55: ISAKMP AAA: Accounting received kei with flags 0x1042
01:09:55: ISAKMP AAA: Updating Stats
01:09:55:      Previous in acc (PKTS) IN: 10 OUT: 10
01:09:55:      Traffic on sa (PKTS) IN: 176 OUT: 176

```

Related Commands

| Command | Description |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| crypto isakmp profile | Defines an ISAKMP profile and audits IPSec user sessions. |
| crypto map (global IPSec) | Enters crypto map configuration mode and creates or modifies a crypto map entry, creates a crypto profile that provides a template for configuration of a dynamically created crypto map, or configures a client accounting list. |

debug crypto key-exchange

To show Digital Signature Standard (DSS) public key exchange messages, use the **debug crypto key-exchange** command in privileged EXEC mode. The **no** form of this command disables debugging output.

debug crypto key-exchange

no debug crypto key-exchange

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Encryption and authentication are provided by a software service on the router called a *crypto engine*. The crypto engine performs authentication through DSS public and private keys when a connection is set up. DSS is a means of sending a “signature” at the end of a message that positively identifies the author of the message. The signature cannot be forged or duplicated by others, so whoever received a message with a DSS signature knows exactly who sent the message.

If the process of exchanging DSS public keys with a peer router by means of the **config crypto key-exchange** command is not successful, try to exchange DSS public keys again after enabling the **debug crypto key-exchange** command to help you diagnose the problem.

Examples

The following is sample output from the **debug crypto key-exchange** command. The first shows output from the initiating router in a key exchange. The second shows output from the passive router in a key exchange. The number of bytes received should match the number of bytes sent from the initiating side, although the number of messages can be different.

```
Router# debug crypto key-exchange
```

```
CRYPTO-KE: Sent 4 bytes.  
CRYPTO-KE: Sent 2 bytes.  
CRYPTO-KE: Sent 2 bytes.  
CRYPTO-KE: Sent 2 bytes.  
CRYPTO-KE: Sent 64 bytes.
```

```
Router# debug crypto key-exchange
```

```
CRYPTO-KE: Received 4 bytes.  
CRYPTO-KE: Received 2 bytes.  
CRYPTO-KE: Received 2 bytes.  
CRYPTO-KE: Received 2 bytes.  
CRYPTO-KE: Received 49 bytes.  
CRYPTO-KE: Received 15 bytes.
```

Related Commands

| Command | Description |
|----------------------------|-----------------------------------------------------------------------|
| debug crypto sesgmt | Displays connection setup messages and their flow through the router. |

debug crypto mib

To display debug messages for the IP Security (IPSec) MIB subsystem, use the **debug crypto mib** command in privileged EXEC mode. To disable the IPSec MIB debug message notifications, use the **no** form of this command.

debug crypto mib

no debug crypto mib

Syntax Description This command has no arguments or keywords.

Defaults Message notification debugging is not enabled.

Command Modes Privileged EXEC

Command History

| Release | Modification |
|----------|------------------------------------------------------|
| 12.1(4)E | This command was introduced. |
| 12.2(4)T | This command was integrated into Cisco IOS 12.2(4)T. |

Examples

The following example shows IPSec MIB debug message notification being enabled:

```
Router# debug crypto mib

Crypto IPSec Mgmt Entity debugging is on
```

Related Commands

| Command | Description |
|-----------------------------------------------------------|---------------------------------------------------------|
| show crypto mib ipsec flowmib history failure size | Displays the size of the IPSec failure history table. |
| show crypto mib ipsec flowmib history tunnel size | Displays the size of the IPSec tunnel history table. |
| show crypto mib ipsec flowmib version | Displays the IPSec Flow MIB version used by the router. |

debug crypto pki messages

To display debug messages for the details of the interaction (message dump) between the certification authority (CA) and the router, use the **debug crypto pki messages** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto pki messages

no debug crypto pki messages

Syntax Description

This command has no arguments or keywords.

Defaults

Disabled

Command History

| Release | Modification |
|---------|------------------------------|
| 12.0 | This command was introduced. |

Usage Guidelines

Use the **debug crypto pki messages** command to display messages about the actual data being sent and received during public key infrastructure (PKI) transactions.

You can also use the **show crypto ca certificates** command to display information about your certificate.

Examples

The following example is sample output for the **debug crypto pki messages** command:

```
Router# debug crypto pki messages

Fingerprint: 2CFC6265 77BA6496 3AEFCB50 29BC2BF2
00:48:23:Write out pkcs#10 content:274
00:48:23:30 82 01 0E 30 81 B9 02 01 00 30 22 31 20 30 1E 06 09 2A 86
00:48:23:48 86 F7 0D 01 09 02 16 11 70 6B 69 2D 33 36 61 2E 63 69 73
00:48:23:63 6F 2E 63 6F 6D 30 5C 30 0D 06 09 2A 86 48 86 F7 0D 01 01
00:48:23:01 05 00 03 4B 00 30 48 02 41 00 DD 2C C6 35 A5 3F 0F 97 6C
00:48:23:11 E2 81 95 01 6A 80 34 25 10 C4 5F 3D 8B 33 1C 19 50 FD 91
00:48:23:6C 2D 65 4C B6 A6 B0 02 1C B2 84 C1 C8 AC A4 28 6E EF 9D 3B
00:48:23:30 98 CB 36 A2 47 4E 7E 6F C9 3E B8 26 BE 15 02 03 01 00 01
00:48:23:A0 32 30 10 06 09 2A 86 48 86 F7 0D 01 09 07 31 03 13 01 63
00:48:23:30 1E 06 09 2A 86 48 86 F7 0D 01 09 0E 31 11 14 0F 30 0D 30
00:48:23:0B 06 03 55 1D 0F 04 04 03 02 05 A0 30 0D 06 09 2A 86 48 86
00:48:23:F7 0D 01 01 04 05 00 03 41 00 2C FD 88 2C 8A 13 B6 81 88 EA
00:48:23:5C FD AE 52 8F 2C 13 95 9E 9D 8B A4 C9 48 32 84 BF 05 03 49
00:48:23:63 27 A3 AC 6D 74 EB 69 E3 06 E9 E4 9F 0A A8 FB 20 F0 02 03
00:48:23:BE 90 57 02 F2 75 8E 0F 16 60 10 6F BE 2B
00:48:23:Enveloped Data ...

00:48:23:30 80 06 09 2A 86 48 86 F7 0D 01 07 03 A0 80 30 80 02 01 00
00:48:23:31 80 30 82 01 0F 02 01 00 30 78 30 6A 31 0B 30 09 06 03 55
00:48:23:04 06 13 02 55 53 31 0B 30 09 06 03 55 04 08 13 02 43 41 31
00:48:23:13 30 11 06 03 55 04 07 13 0A 53 61 6E 74 61 20 43 72 75 7A
00:48:23:31 15 30 13 06 03 55 04 0A 13 0C 43 69 73 63 6F 20 53 79 73
00:48:23:74 65 6D 31 0E 30 0C 06 03 55 04 0B 13 05 49 50 49 53 55 31
```

```

00:48:23:Signed Data 1382 bytes
00:48:23:30 80 06 09 2A 86 48 86 F7 0D 01 07 02 A0 80 30 80 02 01 01
00:48:23:31 0E 30 0C 06 08 2A 86 48 86 F7 0D 02 05 05 00 30 80 06 09
00:48:23:2A 86 48 86 F7 0D 01 07 01 A0 80 24 80 04 82 02 75 30 80 06
00:48:23:02 55 53 31 0B 30 09 06 03 55 04 08 13 02 43 41 31 13 30 11
00:48:23:33 34 5A 17 0D 31 30 31 31 31 35 31 38 35 34 33 34 5A 30 22
00:48:23:31 20 30 1E 06 09 2A 86 48 86 F7 0D 01 09 02 16 11 70 6B 69
00:48:23:2D 33 36 61 2E 63 69 73 63 6F 2E 63 6F 6D 30 5C 30 0D 06 09
00:48:23:2A 86 48 86 F7 0D 01 01 01 05 00 03 4B 00 30 48 02 41 00 DD
00:48:23:2C C6 35 A5 3F 0F 97 6C 11 E2 81 95 01 6A 80 34 25 10 C4 5F
00:48:23:3D 8B 33 1C 19 50 FD 91 6C 2D 65 4C B6 A6 B0 02 1C B2 84 C1
00:48:23:86 F7 0D 01 01 01 05 00 04 40 C6 24 36 D6 D5 A6 92 80 5D E5
00:48:23:15 F7 3E 15 6D 71 E1 D0 13 2B 14 64 1B 0C 0F 96 BF F9 2E 05
00:48:23:EF C2 D6 CB 91 39 19 F8 44 68 0E C5 B5 84 18 8B 2D A4 B1 CD
00:48:23:3F EC C6 04 A5 D9 7C B1 56 47 3F 5B D4 93 00 00 00 00 00
00:48:23:00 00
00:48:24:Received pki message:1778 types
00:48:24:30 82 06 EE 06 09 2A 86 48 86 F7 0D 01 07 02 A0 82 06 DF 30
00:48:24:82 06 DB 02 01 01 31 0E 30 0C 06 08 2A 86 48 86 F7 0D 02 05
00:48:24:05 00 30 82 04 C5 06 09 2A 86 48 86 F7 0D 01 07 01 A0 82 04
00:48:24:B6 04 82 04 B2 30 82 04 AE 06 09 2A 86 48 86 F7 0D 01 07 03
00:48:24:0E 61 85 48 B1 DA 3D 73 F1 4B D8 5E 03 6E F3 E5 72 5D D7 17
00:48:24:17 3D 03 19 B3 8F 06 8B FE FB B1 CE D4 4C 4D 1B 81 CF 59 B7
00:48:24:78 DD 27 BA 28 2F 85 09 F0 61 74 0F 0F 92 F0 C8 C7 5B 96 E7
00:48:24:71 AF 87 D2 72 75 B7 F7 89 6F E4 E7 57 84 76 53 0B 50 8A B9
00:48:24:05 54 6F 06 75 72 8A AF 54 A6 EF 70 2D 15 6C B7 30 91 1C 00
00:48:24:CB 26 80 8D DC 89 77 57 1E D5 7A 37 86 BE 44 F8 66 60
00:48:24:Verified signed data 1202 bytes:
00:48:24:30 82 04 AE 06 09 2A 86 48 86 F7 0D 01 07 03 A0 82 04 9F 30
00:48:24:82 04 9B 02 01 00 31 81 9F 30 81 9C 02 01 00 30 46 30 22 31
00:48:24:20 30 1E 06 09 2A 86 48 86 F7 0D 01 09 02 16 11 70 6B 69 2D
00:48:24:33 36 61 2E 63 69 73 63 6F 2E 63 6F 6D 02 20 34 45 45 41 44
00:48:24:E2 55 65 DE DB 23 91 D7 60 53 96 64 BE F2 30 A7 8B 1B D9 EB
00:48:24:2E EB 9B 0D 75 EC 8E AF C0 9C 62 78 29 E0 97 00 EA 84 80 DD
00:48:24:AB 83 32 89 3E 5B A9 9F A9 9A 6D 3A 87 E2 71 16 C9 C1 E4 DB
00:48:24:FA 5A FC F3 31 98 2B 8E 55 71 C4 F6 BF CE 45 CA A5 47 40 9B
00:48:24:19 E3 1A C3 F5 ED 4D 81 1F 6F 34 35 E2 00 B3 93 DD A0 6A 74
00:48:24:EA 2B A8 D4 32 53 A7 86 50 71 5E 2A 64 BE 4B B1 72 AB 8C DA
00:48:24:AB 7A 2A 07 C0 7E C1 A7 12 31 33 AB 94 E0 3B A2 68 17 DE CE
00:48:24:57 70 2D 0B F5 C8 A7 FC FE 40 74 E8 EB 9C 82 77 DE A4 FA 75
00:48:24:FF 6F 7B E6 74 E2 F5 A1 9A C8 3C 23 DB 4A 90 BE 4A 94 EB 8B
00:48:24:ED F3
00:48:24:Decrypted enveloped content:
00:48:24:30 82 03 C8 06 09 2A 86 48 86 F7 0D 01 07 02 A0 82 03 B9 30
00:48:24:82 03 B5 02 01 01 31 00 30 0B 06 09 2A 86 48 86 F7 0D 01 07
00:48:24:01 A0 82 03 9D 30 82 03 99 30 82 03 43 A0 03 02 01 02 02 0A
00:48:24:70 45 B3 F6 00 00 00 00 01 23 30 0D 06 09 2A 86 48 86 F7 0D
00:48:24:35 35 32 32 5A 30 22 31 20 30 1E 06 09 2A 86 48 86 F7 0D 01
00:48:24:09 02 13 11 70 6B 69 2D 33 36 61 2E 63 69 73 63 6F 2E 63 6F
00:48:24:6D 30 5C 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01 05 00 03 4B
00:48:24:00 30 48 02 41 00 DD 2C C6 35 A5 3F 0F 97 6C 11 E2 81 95 01
00:48:24:6A 80 34 25 10 C4 5F 3D 8B 33 1C 19 50 FD 91 6C 2D 65 4C B6
00:48:24:63 6F 2E 63 6F 6D 2F 43 65 72 74 45 6E 72 6F 6C 6C 2F 6D 73
00:48:24:63 61 2D 72 6F 6F 74 5F 6D 73 63 61 2D 72 6F 6F 74 2E 63 72
00:48:24:74 30 41 06 08 2B 06 01 05 05 07 30 02 86 35 66 69 6C 65 3A
00:48:24:2F 2F 5C 5C 6D 73 63 61 2D 72 6F 6F 74 5C 43 65 72 74 45 6E
00:48:24:72 6F 6C 6C 5C 6D 73 63 61 2D 72 6F 6F 74 5F 6D 73 63 61 2D
00:48:24:72 6F 6F 74 2E 63 72 74 30 0D 06 09 2A 86 48 86 F7 0D 01 01
00:48:24:05 05 00 03 41 00 56 30 AD 99 1F FA 0D 1A C3 3D 71 2A DB A0
00:48:24:48 C5 EB C8 D4 FE 62 49 9C 69 5D E4 80 77 19 3E 07 B8 2B 4F
00:48:24:9A D7 72 A7 26 25 61 AE 5B 1C B5 7B 4C 18 CA 17 C3 D0 76 84
00:48:24:75 41 92 74 5E A4 E8 9E 09 60 31 00
00:48:24:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority

```

Related Commands

| Command | Description |
|--------------------------------------|----------------------------------------------------------------------------------------------------|
| crypto ca enroll | Obtains the certificate of your router from the CA. |
| debug crypto pki transactions | Displays debug messages for the trace of interaction (message type) between the CA and the router. |
| show crypto ca certificates | Displays information about your certificate, the certificate of the CA, and any RA certificates. |