

# debug clns esis events

To display uncommon End System-to-Intermediate System (ES-IS) events, including previously unknown neighbors, neighbors that have aged out, and neighbors that have changed roles (ES-IS, for example), use the **debug clns esis events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug clns esis events**

**no debug clns esis events**

---

**Syntax Description** This command has no arguments or keywords.

---

**Command Modes** Privileged EXEC

---

**Examples** The following is sample output from the **debug clns esis events** command:

```
Router# debug clns esis events

ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```

The following line indicates that the router received a hello packet (ISH) from the IS at MAC address aa00.0400.2c05 on Ethernet interface 1. The hold time (or number of seconds to consider this packet valid before deleting it) for this packet is 30 seconds.

```
ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
```

The following line indicates that the router received a hello packet (ESH) from the ES at MAC address aa00.0400.9105 on the Ethernet interface 1. The hold time is 150 seconds.

```
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
```

The following line indicates that the router sent an IS hello packet on the Ethernet interface 0 to all ESs on the network. The network entity title (NET) address of the router is 49.0001.0400.AA00.6904.00; the hold time for this packet is 299 seconds; and the header length of this packet is 20 bytes.

```
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```

# debug clns esis packets

To enable display information on End System-to-Intermediate System (ES-IS) packets that the router has received and sent, use the **debug clns esis packets** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug clns esis packets**

**no debug clns esis packets**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## Examples

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

```
Router# debug clns esis packets
```

```
ES-IS: ISH sent to All ESs (Ethernet0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
ES-IS: ISH sent to All ESs (Ethernet1): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
ES-IS: ISH from aa00.0400.6408 (Ethernet0), HT 299
ES-IS: ISH sent to All ESs (Tunnel0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.0906.4023.00, HT 299, HLEN 34
IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
```

The following line indicates that the router has sent an IS hello packet on Ethernet interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Ethernet0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
```

The following line indicates that the router has sent an IS hello packet on Ethernet interface 1 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Ethernet1): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```

The following line indicates that the router received a hello packet on Ethernet interface 0 from an intermediate system, aa00.0400.6408. The hold time for this packet is 299 seconds.

```
ES-IS: ISH from aa00.0400.6408 (Ethernet0), HT 299
```

The following line indicates that the router has sent an IS hello packet on Tunnel interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Tunnel0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```

The following line indicates that on Ethernet interface 0, the router received a hello packet from an end system with an SNPA of 0000.0c00.bda8. The hold time for this packet is 300 seconds.

```
IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
```

# debug clns events

To display Connectionless Network Service (CLNS) events that are occurring at the router, use the **debug clns events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug clns events**

**no debug clns events**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## Examples

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

```
Router# debug clns events

CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00
      via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
CLNS: Forwarding packet size 117
      from 39.0001.2222.2222.2222.00
      to 49.0002.0001.AAAA.AAAA.AAAA.00
      via 49.0002 (Ethernet3 0000.0c00.b5a3)
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18,
      redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```

The following line indicates that the router received an echo protocol data unit (PDU) on Ethernet interface 3 from source network service access point (NSAP) 39.0001.2222.2222.2222.00. The exclamation point at the end of the line has no significance.

```
CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
```

The following lines indicate that the router at source NSAP 39.0001.3333.3333.3333.00 is sending a CLNS echo packet to destination NSAP 39.0001.2222.2222.2222.00 via an IS with system ID 2222.2222.2222. The packet is being sent on Ethernet interface 3, with a MAC address of 0000.0c00.3a18.

```
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00
      via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
```

The following lines indicate that a CLNS echo packet 117 bytes in size is being sent from source NSAP 39.0001.2222.2222.2222.00 to destination NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 via the router at NSAP 49.0002. The packet is being forwarded on the Ethernet interface 3, with a MAC address of 0000.0c00.b5a3.

```
CLNS: Forwarding packet size 117
      from 39.0001.2222.2222.2222.00
      to 49.0002.0001.AAAA.AAAA.AAAA.00
      via 49.0002 (Ethernet3 0000.0c00.b5a3)
```

The following lines indicate that the router sent a redirect packet on the Ethernet interface 3 to the NSAP 39.0001.2222.2222.2222.00 at MAC address 0000.0c00.3a18 to indicate that NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 can be reached at MAC address 0000.0c00.b5a3.

```
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18,  
      redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```

# 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. To disable debugging output, use the **no** form of this command.

**debug clns packet**

**no debug clns packet**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## 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 subnetwork point of attachment (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 protocol data unit (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. To disable debugging output, use the **no** form of this command.

**debug clns routing**

**no debug clns routing**

---

**Syntax Description** This command has no arguments or keywords.

---

**Command Modes** Privileged EXEC

---

**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. To disable debugging output, use the **no** form of this command.

**debug cls message**

**no debug cls message**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

**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 dependent logical unit (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 service access point (SAP), XID, exchange identification (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
<b>debug fras error</b>	Displays information about FRAS protocol errors.
<b>debug fras message</b>	Displays general information about FRAS messages.
<b>debug fras state</b>	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.

**Command Modes** Privileged EXEC

**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 systems network architecture (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:

```

V DLC: Req Open Stn Req PSap 0x7B68C0, port 0x7992A0
      4000.1060.1000(04)->4000.05d2.0001(0C)
V DLC: two-way connection established
V DLC: 4000.1060.1000(04)->4000.05d2.0001(0C): V DLC_IDLE->V DLC_OPEN

```

In the following messages, DLSw sends an XID response, and the NSP 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
<code>debug cls message</code>	Displays information about CLS messages.

# debug cns config

To turn on debugging messages related to the Cisco Networking Services (CNS) Configuration Agent, use the **debug cns config** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

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

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

## Syntax Description

<b>agent</b>	Displays debugging messages related to the CNS configuration agent.
<b>all</b>	Displays all debugging messages.
<b>connection</b>	Displays debugging messages related to configuration connections.
<b>notify</b>	Displays debugging 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(18)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 debugging 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**

<b>Command</b>	<b>Description</b>
<b>cns config cancel</b>	Cancels a CNS configuration.
<b>cns config initial</b>	Starts the initial CNS Configuration Agent.
<b>cns config partial</b>	Starts the partial CNS Configuration Agent.
<b>cns config retrieve</b>	Gets the configuration of a routing device using CNS.
<b>debug cns event</b>	Displays information on CNS events.
<b>debug cns exec</b>	Displays information on CNS management.
<b>debug cns xml-parser</b>	Displays information on the CNS XML parser.
<b>show cns config</b>	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 disable debugging output, use the **no** form of this command.

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

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

## Syntax Description

<b>agent</b>	Displays debugging messages related to the event agent.
<b>all</b>	Displays all debugging messages.
<b>connection</b>	Displays debugging messages related to event connections.
<b>subscriber</b>	Displays debugging 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 Release 12.0(18)ST.
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 debugging 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.....
```

```

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**

<b>Command</b>	<b>Description</b>
<b>cns event</b>	Configures the CNS Event Gateway.
<b>show cns event</b>	Displays information about the CNS Event Agent.

# debug cns exec

To display debugging messages about CNS exec agent services, use the **debug cns exec** command in privileged EXEC mode. To disable debugging output, use the **no** or **undebug** form of this command.

**debug cns exec** {agent | all | decode | messages}

**no debug cns exec** {agent | all | decode | messages}

**undebug cns exec** {agent | all | decode | messages}

## Syntax Description

<b>agent</b>	Displays debugging messages related to the exec agent.
<b>all</b>	Displays all debugging messages.
<b>decode</b>	Displays debugging messages related to image agent connections.
<b>messages</b>	Displays debugging output related to messages generated by exec agent services.

## Defaults

Debugging output is disabled.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.3(1)	This command was introduced.

## Usage Guidelines

Use the **debug cns exec** command to troubleshoot CNS exec agent services.

## Examples

The following example shows a debugging message for the CNS exec agent when a response has been posted to HTTP:

```
Router# debug cns exec agent
4d20h: CNS exec agent: response posted
```

## Related Commands

Command	Description
<b>cns exec</b>	Configures CNS Exec Agent services.

# debug cns image

To display debugging messages about Cisco Networking Services (CNS) image agent services, use the **debug cns image** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug cns image {agent | all | connection | error}
```

```
no debug cns image {agent | all | connection | error}
```

## Syntax Description

<b>agent</b>	Displays debugging messages related to the image agent.
<b>all</b>	Displays all debugging messages.
<b>connection</b>	Displays debugging messages related to image agent connections.
<b>error</b>	Displays debugging messages related to errors generated by image agent services.

## Defaults

If no keyword is specified, all debugging messages are displayed.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.3(1)	This command was introduced.

## Usage Guidelines

Use the **debug cns image** command to troubleshoot CNS image agent services.

# debug cns management

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

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

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

## Syntax Description

<b>snmp</b>	Displays debugging messages related to nongranular Simple Network Management Protocol (SNMP) encapsulated CNS-management events.
<b>xml</b>	Displays debugging 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.

## 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**

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

# debug cns xml-parser

To turn on debugging 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 disable debugging output, use the **no** form of this command.

**debug cns xml-parser**

**no debug cns xml-parser**

**Syntax Description** This command has no arguments or keywords.

**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**

<b>Command</b>	<b>Description</b>
<b>cns event</b>	Configures the CNS Event Gateway.
<b>show cns event</b>	Displays information about the CNS Event Agent.

---

# debug compress

To debug compression, enter the **debug compress** command in privileged EXEC mode. 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 Modes** Privileged EXEC

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 47 describes the significant fields shown in the display.

**Table 47** *debug compress Field Descriptions*

Field	Description
compr-in	Indicates that a packet needs to be compressed.
compr-out	Indicates completion of compression of packet.
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
<b>debug frame-relay</b>	Displays debugging information about the packets that are received on a Frame Relay interface.
<b>debug ppp</b>	Displays information on traffic and exchanges in an internetwork implementing the PPP.
<b>show compress</b>	Displays compression statistics.
<b>show diag</b>	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 remove 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* | **calling** *tid/imsi string*}}

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

## Syntax Description

<b>username</b> <i>username</i>	Generates debugging messages for interfaces with the specified username.
<b>called</b> <i>dial-string</i>	Generates debugging messages for interfaces with the called party number.
<b>caller</b> <i>dial-string</i>	Generates debugging messages for interfaces with the calling party number.
<b>vcid</b> <i>vc-id</i>	Generates debugging messages for the VC ID specified.
<b>ip</b> <i>ip-address</i>	Generates debugging messages for the IP address specified.
<b>calling</b> <i>tid/imsi string</i>	Displays events related to general packet radio service (GPRS) tunneling protocol (GTP) processing on the gateway GPRS support node (GGSN) based on the tunnel identifier (TID) or international mobile system identifier (IMSI) in a Packet Data Protocol (PDP) Context Create Request message.
<i>condition-id</i>	Removes the condition indicated.
<b>all</b>	Removes all debugging conditions, and conditions specified by the <b>debug condition interface</b> command. Use this keyword to disable conditional debugging and reenables 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 <b>vcid</b> and <b>ip</b> 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.
12.3(2)XB	This command was introduced on the GGSN.
12.3(8)T	The <b>calling</b> keyword and <i>tid/imsi string</i> argument were added.

## 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**
- **debug isdn {q921 | q931}**
- **debug modem {oob | trace}**
- **debug ppp {all | authentication | chap | error | negotiation | multilink events | packet}**

Ensure that you enable TID/IMSI-based conditional debugging by entering **debug condition calling** before configuring **debug gprs gtp** and **debug gprs charging**. In addition, ensure that you disable the **debug gprs gtp** and **debug gprs charging** commands using the **no debug all** command before disabling conditional debugging using the **no debug condition** command. This will prevent a flood of debugging messages when you disable conditional debugging.

## 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
<b>debug condition interface</b>	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 debugging output, use the **no** form of this command.

**debug condition application voice** *application-name*

**no debug condition application voice** *application-name*

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

<b>Defaults</b>	If this command is not configured, debugging messages are enabled for all VoiceXML applications.
-----------------	--

<b>Command Modes</b>	Privileged EXEC
----------------------	-----------------

Command History	Release	Modification
	12.2(11)T	This command was introduced for the Cisco 3640, Cisco 3660, Cisco AS5300, Cisco AS5350, and Cisco AS5400.

<b>Usage Guidelines</b>	<ul style="list-style-type: none"> <li>• This command filters debugging output only for the <b>debug vxml</b> and <b>debug http client</b> commands, except that it does not filter output for the <b>debug vxml error</b>, <b>debug vxml background</b>, <b>debug http client error</b>, or <b>debug http client background</b> commands. It does not filter messages for any other debug commands such as the <b>debug voip ivr</b> command or the <b>debug voice ivr</b> command.</li> <li>• 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.</li> <li>• To filter debugging output with this command, the &lt;cisco-debug&gt; element must be enabled in the VoiceXML document. For more information about the &lt;cisco-debug&gt; element, refer to the <a href="#">Cisco VoiceXML Programmer's Guide</a>.</li> <li>• To see debugging output for VoiceXML applications, you must first configure global <b>debug</b> commands such as the <b>debug vxml</b> command or the <b>debug http client</b> command. If no global <b>debug</b> commands are turned on, you do not see debugging messages even if the <b>debug condition application voice</b> command is configured and the &lt;cisco-debug&gt; element is enabled in the VoiceXML document.</li> <li>• This command can be configured multiple times to display output for more than one application.</li> <li>• To see which debug conditions have been set, use the <b>show debug condition</b> command.</li> </ul>
-------------------------	--

**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
<b>debug http client</b>	Displays debugging messages for the HTTP client.
<b>debug vxml</b>	Displays debugging messages for VoiceXML features.
<b>show debug condition</b>	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
<b>debug glbp errors</b>	Displays debugging messages about GLBP errors.
<b>debug glbp events</b>	Displays debugging messages about GLBP events.
<b>debug glbp packets</b>	Displays debugging messages about GLBP packets.
<b>debug glbp terse</b>	Displays a limited range of debugging messages about GLBP errors, events, and packets.

# debug condition interface

To limit output for some debug commands on the basis of the interface or virtual circuit, use the **debug condition interface** command in privileged EXEC mode. To remove the interface condition and reset the interface so that it must be triggered by a condition, use the **no** form of this command.

**debug condition interface** *interface-type interface-number* [**dlci** *dlci*] [**vc** {*vci* | *vpi/vci*}]

**no debug condition interface** *interface-type interface-number* [**dlci** *dlci*] [**vc** {*vci* | *vpi/vci*}]

## Syntax Description

<i>interface-type interface-number</i>	Interface type and number. No space is required between the interface type and number. Some interfaces require a slash between the type and number.
<b>dlci</b> <i>dlci</i>	(Optional) If the interface to be debugged is a Frame Relay-encapsulated interface, specifies the data-link connection identifier (DLCI).
<b>vc</b> { <i>vci</i>   <i>vpi/vci</i> }	(Optional) If the interface to be debugged is an ATM-encapsulated interface, specifies the virtual channel identifier (VCI) or virtual path identifier/virtual channel identifier (VPI/VCI) pair. (The slash is required.)

## Defaults

All debugging messages for enabled **debug** commands are displayed.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.0(28)S	The <b>dlci</b> and <b>vc</b> keywords were added for additional Frame Relay and ATM functionality.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.

## Usage Guidelines

Use this command to restrict the debugging output for some commands on the basis of an interface or virtual circuit. When you enter this command, debugging output is turned off for all interfaces except the specified interface or virtual circuit. In addition, this command enables conditional debugging to limit output for specific 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 applicable conditions. If the interface meets other conditions that have been set by another **debug condition** command, debugging output will still be generated for the interface.
- If some other **debug condition** command has been enabled, output is stopped for that interface until the condition is met on the interface. You will be asked for confirmation before the last condition or all conditions are removed.

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**
- **debug atm**
- **debug dialer events**
- **debug frame-relay**
- **debug isdn**
- **debug modem**
- **debug ppp**

One or more ATM-encapsulated interfaces must be enabled, and one or more of the following **debug** commands must be enabled to use conditional debugging with ATM:

- **debug atm arp**
- **debug atm counters**
- **debug atm errors**
- **debug atm events**
- **debug atm oam**
- **debug atm packet**
- **debug atm state**

One or more of the following **debug** commands must be enabled to use conditional debugging with Frame Relay:

- **debug frame-relay adjacency**
- **debug frame-relay ipc**
- **debug frame-relay lmi**
- **debug frame-relay packet**
- **debug frame-relay pseudowire**

### Examples

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

```
Router# debug condition interface serial 1
```

```
Condition 1 set
```

The following example enables an ATM interface, specifies an IP address for the interface, turns on conditional debugging for that interface with a VPI/VCI pair of 255/62610, and verifies that debugging has been enabled:

```
Router> enable

Password:
Router# configure terminal

Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# interface ATM 2/0
Router(config-if)# ip address 10.0.0.5 255.255.255.0
Router(config-if)# pvc 255/62610
Router(config-if-atm-vc)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router#
2w3d: %SYS-5-CONFIG_I: Configured from console by console
Router# debug atm state
ATM VC States debugging is on
Router# debug condition interface ATM2/0 vc 255/62610
Condition 1 set
Router#
2w3d: ATM VC Debug: Condition 1, atm-vc 255/62610 AT2/0 triggered, count 1
Router# show debug condition
Condition 1: atm-vc 255/62610 AT2/0 (1 flags triggered)
      Flags: ATM VC
```

In the following example, Frame Relay conditional debugging is enabled on Frame Relay DLCI 105:

```
Router# debug condition interface serial 4/3 dlci 105
Router# debug frame-relay packet
```

**Related Commands**

Command	Description
<b>debug condition</b>	Limits output for some debug commands on the basis of specific conditions.

# debug condition match-list

To run a filtered debug on a voice call, use the **debug condition match-list** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug condition match-list number {exact-match | partial-match}
```

```
no debug condition match-list number {exact-match | partial-match}
```

Syntax Description	
<i>number</i>	Numeric label that uniquely identifies the match list. Range is 1 to 16. The number for the match list is set using the <b>call filter match-list</b> command.
<b>exact-match</b>	All related debug output is filtered until all conditions in the match list are explicitly met. This is the best choice for most situations because the output is the most concise.
<b>partial-match</b>	No related debug output is filtered until there is a single explicit match failure. As long as zero or more conditions are met, debug output will not be filtered. This choice is useful in debugging call startup problems like digit collection, but is not ideal for many situations because there is much debug output until matches explicitly fail.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.3(4)T	This command was introduced.

**Examples** In this output example, the following configuration was used:

```
call filter match-list 1 voice
  incoming calling-number 8288807
  incoming called-number 6560729
  incoming port 7/0:D
```

The following is sample output for the **debug condition match-list 1** command. The next several lines match the above conditions.

```
Router# debug condition match-list 1

07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_gcfm_incoming_c
ond_notify: add incoming port cond success: 7/0:D

07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_gcfm_incoming_c
ond_notify: add incoming dialpeer tag success:1
07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_update_dsm_stre
am_mgr_filter_flag: cannot find dsp_stream_mgr_t
07:22:19://-1/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_update_dsm_stre
am_mgr_filter_flag: update dsp_stream_mgr_t debug flag
```

```
07:22:19: //49/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_insert_cdb:
,cdb 0x6482C518, CallID=49
07:22:19://49/3C0B9468-15C8-11D4-8013-000A8A389BA8/VTSP:(7/0:D):0:0:0/vtsp_do_call_setup_i
nd: Call ID=98357, guid=3C0B9468-15C8-11D4-8013-000A8A389BA8
```

Table 48 describes the significant fields shown in the display.

**Table 48** *debug condition match-list Field Descriptions*

Field	Description
3C0B9468-15C8-11D4-8013-000A8A389BA8	Shows the global unique identifier (GUID).
VTSP:	Identifies the voice telephony service provider (VTSP) module.
(7/0:D):0:0:0	Shows the port name, channel number, DSP slot, and DSP channel number for the VTSP module.

**Related Commands**

Command	Description
<b>call filter match-list voice</b>	Creates a call filter match list for debugging voice calls.
<b>debug call filter inout</b>	Displays the debug trace inside the GCFM.
<b>show call filter match-list</b>	Displays call filter match lists.

# debug condition standby

To filter the output of the **debug standby** command on the basis of interface and Hot Standby Router Protocol (HSRP) group number, use the **debug condition standby** command in privileged EXEC mode. To remove the specified filter condition, use the **no** form of this command.

**debug condition standby** *interface group-number*

**no debug condition standby** *interface group-number*

## Syntax Description

<i>interface</i>	Filters output on the basis of the interface.
<i>group-number</i>	Filters output on the basis of HSRP group number. The range is 0 to 255 for HSRP Version 1 and 0 to 4095 for HSRP Version 2.

## Defaults

All debugging messages for the **debug standby** command are generated.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.1(2)	This command was introduced.

## Usage Guidelines

Use the **debug condition standby** command to restrict the debug output for the **debug standby** command. If the **debug condition standby** command is enabled, output is generated only for the interfaces and HSRP group numbers specified. The interface you specify must be a valid interface capable of supporting HSRP. The group can be any group (0 to 255 for HSRPv1 and 0 to 4095 for HSRPv2).

Use the **no** form of this command to remove the HSRP debug condition. 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.

You can set debug conditions for groups that do not exist, which allows you to capture debug information during the initialization of a new group.

You must enable the **debug standby** command in order for any HSRP debug output to be produced. If you do not configure the **debug condition standby** command after entering the **debug standby** command, then debug output is produced for all groups on all interfaces.

## Examples

In the following example, the router displays debugging messages only for Ethernet interface 0/0 that are part of HSRP group 23:

```
Router# debug standby
HSRP debugging is on
```

```
Router# debug condition standby ethernet0/0 23
```

```

Condition 1 set
00:27:39: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:42: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:45: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:48: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
00:27:51: HSRP: Et0/0 Grp 23 Hello out 10.0.0.1 Active pri 100 vIP 172.16.6.5
    
```

The following example shows how to remove an HSRP debug condition:

```
Router# no debug condition standby ethernet0/0 23
```

This condition is the last hsrp condition set.  
 Removing all conditions may cause a flood of debugging messages to result, unless specific debugging flags are first removed.

```

Proceed with removal? [yes/no]: Y
Condition 1 has been removed.
    
```

**Related Commands**

Command	Description
<b>debug condition interface</b>	Limits output for some debugging commands based on the interfaces.
<b>debug standby</b>	Displays HSRP state changes.
<b>debug standby errors</b>	Displays error messages related to HSRP.
<b>debug standby events</b>	Displays events related to HSRP.
<b>debug standby events icmp</b>	Displays debugging messages for the HSRP ICMP redirects filter.
<b>debug standby packets</b>	Displays debugging information for packets related to HSRP.

# 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. To disable debugging output, use the **no** form of this command.

**debug confmodem**

**no debug confmodem**

---

**Syntax Description** This command has no arguments or keywords.

---

**Command Modes** Privileged EXEC

---

**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 debugging output, use the **no** form of this 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 Common Open Policy Service (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]**

<b>Syntax Description</b>	<b>detail</b>	(Optional) Displays additional debug information, including the contents of COPS and Resource Reservation Protocol (RSVP) messages.
---------------------------	---------------	---

**Defaults** COPS process debugging is not enabled.

**Command Modes** Privileged EXEC

<b>Command History</b>	<b>Release</b>	<b>Modification</b>
	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)
```

```
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**

<b>Command</b>	<b>Description</b>
<b>debug ip rsvp policy</b>	Displays debugging messages for RSVP policy processing.

# debug cot

To display information about the Continuity Test (COT) functionality, use the **debug cot** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug cot** { **api** | **dsp** | **queue** | **detail** }

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

## Syntax Description

<b>api</b>	Displays information about the COT application programming interface (API).
<b>dsp</b>	Displays information related to the COT/Digital Signal Processor configuration (DSP) interface. Typical DSP functions include data modems, voice codecs, fax modems and codecs, and low-level signaling such as channel-associated signaling (CAS)/R2.
<b>queue</b>	Display information related to the COT internal queue.
<b>detail</b>	Display information about COT internal detail; summary of the <b>debug cot api</b> , <b>debug cot dsp</b> , and <b>debug cot queue</b> commands.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
11.3(7)	This command was introduced.

## Examples

The following is sample output from the **debug cot api** command:

```
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 0xFFFF1 duration 60000
```

[Table 49](#) describes the significant fields shown in the display.

**Table 49** *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.

**Table 49** *debug cot api Field Descriptions (continued)*

Field	Description
ds0	Number of the COT operation request.
key	COT operation identifier.
duration	Timeout duration of the COT operation.
freqTX	Requested transmit tone frequency.
freqRX	Requested receive tone frequency.

The following is sample output from the **debug cot dsp** command:

```
Router# debug cot dsp

Router#
00:10:42:COT:DSP (1/1) Allocated
00:10:43:In cot_callback
00:10:43: returned key 0xFFF1, 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 50](#) describes the significant fields shown in the display.

**Table 50** *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 from 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 0xFFF1, status = 0
```

Table 51 describes the significant fields shown in the display.

**Table 51** *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 from 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



## Note

Effective with Release 12.3(4)T, the **debug cpp event** command is no longer available in Cisco IOS software.

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.

## Command History

Release	Modification
11.2	This command was introduced.
12.3(4)T	This command was removed and is no longer available in Cisco IOS software.

## 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 52](#) 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 52** *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.

**Table 52** *debug cpp event Messages (continued)*

<b>Message</b>	<b>Description</b>
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**

<b>Command</b>	<b>Description</b>
<b>debug cpp negotiation</b>	Displays CPP negotiation events.
<b>debug cpp packet</b>	Displays CPP packets.

# debug cpp negotiation



## Note

Effective with Release 12.3(4)T, the **debug cpp negotiation** command is no longer available in Cisco IOS software.

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.

## Command History

Release	Modification
11.2	This command was introduced.
12.3(4)T	This command was removed and is no longer available in Cisco IOS software.

## 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:ber19
  password is correct
```

[Table 53](#) describes the significant fields in the display.

**Table 53** *debug CPP negotiation Field Descriptions*

<b>Field</b>	<b>Description</b>
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.

**Related Commands**

<b>Command</b>	<b>Description</b>
<b>debug cot</b>	Displays information about the COT functionality.
<b>debug cpp packet</b>	Displays CPP packets.

# debug cpp packet



## Note

Effective with Release 12.3(4)T, the **debug cpp packet** command is no longer available in Cisco IOS software.

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.

## Command History

Release	Modification
11.2	This command was introduced.
12.3(4)T	This command was removed and is no longer available in Cisco IOS software.

## 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 0. !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
<b>debug cot</b>	Displays information about the COT functionality.
<b>debug cpp negotiation</b>	Displays CPP negotiation events.

# debug crm

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

```
debug crm [all | default | detail | error [call [informational] | software [informational]] | function | inout]
```

```
no debug crm
```

## Syntax Description

<b>all</b>	(Optional) Displays all CRM debugging messages.
<b>default</b>	(Optional) Displays detail, error, and inout information. This option also runs if no keywords are added.
<b>detail</b>	(Optional) Displays non-inout information related to call processing, such as call updates or call acceptance checking.
<b>error</b>	(Optional) Displays CRM error messages.
<b>call</b>	(Optional) Displays call processing errors.
<b>informational</b>	(Optional) Displays minor errors and major errors. Without the <b>informational</b> keyword, only major errors are displayed.
<b>software</b>	(Optional) Displays software errors.
<b>function</b>	(Optional) Displays CRM function names and exit points from each function so that call processing can be traced within the CRM subsystem.
<b>inout</b>	(Optional) Displays information from the functions that form the external interfaces of CRM to other modules or subsystems.

## Defaults

Debugging is not enabled.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.2(11)T	This command was introduced.
12.3(8)T	The <b>all</b> , <b>default</b> , <b>detail</b> , <b>error</b> , <b>call</b> , <b>informational</b> , <b>software</b> , <b>function</b> , and <b>inout</b> keywords were added.

## 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**

The following is sample output from the **debug crm all** command for an incoming ISDN call on a trunk group:

```
Router# debug crm all

01:21:23: //-1/xxxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:21:23: //-1/xxxxxxxxxxxxx/CRM/print_event:
    RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both
Capacity Update,
    Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
01:21:23: //-1/xxxxxxxxxxxxx/CRM/crm_call_update:
    Increment the call count
    CarrierID=2023, TrunkGroupLabel=2023
    Update is for TrunkGroupLabel, Mask=0x00000001
01:21:23: //-1/xxxxxxxxxxxxx/CRM/crm_call_update:
    IncomingVoiceCalls=1
Router#
01:21:48: //-1/xxxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:21:48: //-1/xxxxxxxxxxxxx/CRM/print_event:
    RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both
Capacity Update,
    Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
01:22:13: //-1/xxxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:22:13: //-1/xxxxxxxxxxxxx/CRM/print_event:
    RouteLabel=2023, CarrierType=TDM, EventType=Single RouteLabel Update, EventReason=Both
Capacity Update,
    Max Capacity mask 0x0000001F, Current Capacity Mask 0x0000001F
Router#
01:22:18: //-1/xxxxxxxxxxxxx/CRM/crm_call_update:
    Decrement the call count
    CarrierID=2023, TrunkGroupLabel=2023
    Update is for TrunkGroupLabel, Mask=0x00000001
01:22:18: //-1/xxxxxxxxxxxxx/CRM/crm_call_update:
    IncomingVoiceCalls=0
```

Table 54 describes the significant fields shown in the display.

**Table 54** *debug crm all Field Descriptions*

Field	Description
//-1/xxxxxxxxxxxxx/CRM/print_event:	The format of this message is //callid/GUID/CRM/function name: <ul style="list-style-type: none"> <li>• CallEntry ID is -1. This indicates that the CallEntry ID is unavailable.</li> <li>• GUID is xxxxxxxxxxxxxxxx. This indicates that the GUID is unavailable.</li> <li>• CRM is the module name.</li> <li>• The print_event: field shows that the CRM is showing the print event function.</li> </ul>
RouteLabel	Either the trunk group label or carrier ID.
CarrierType	Indicates the type of trunk.
EventType	Indicates if a single route or all routes are updated.
EventReason	Shows the reason for this event being sent.

**Related Commands**

<b>Command</b>	<b>Description</b>
<b>max-calls</b>	Specifies the maximum number of calls the trunk group can handle.

# debug crypto condition

To define conditional debug filters, use the **debug crypto condition** command in privileged EXEC mode. To disable conditional debugging, use the **no** form of this command.

```
debug crypto condition [connid integer engine-id integer] [flowid integer engine-id integer]
[fvr string] [ivrf string] [peer [group string] [hostname string] [ipv4 ipaddress]
[subnet subnet mask] [username string]] [spi integer] reset
```

```
no debug crypto condition [connid integer engine-id integer] [flowid integer engine-id integer]
[fvr string] [ivrf string] [peer [group string] [hostname string] [ipv4 ipaddress]
[subnet subnet mask] [username string]] [spi integer] reset
```

Syntax Description	
<b>connid</b> <i>integer</i> <sup>1</sup>	(Optional) Internet Key Exchange (IKE) and IP Security (IPSec) connection ID filter. Valid values range from 1 to 32766.
<b>engine-id</b> <i>integer</i>	(Optional) Crypto engine ID value, which can be retrieved via the <b>show crypto isakmp sa detail</b> command. Valid values are 1, which represents software engines, and 2, which represents hardware engines.
<b>flowid</b> <i>integer</i>	(Optional) IPSec flow-ID filter. Valid values range from 1 to 32766.
<b>fvr</b> <i>string</i> <sup>1</sup>	(Optional) Front-door virtual private network (VPN) routing and forwarding (FVRF) filter. The <i>string</i> argument must be the name string of an FVRF instance.
<b>ivrf</b> <i>string</i> <sup>1</sup>	(Optional) Inside VRF (IVRF) filter. The <i>string</i> argument must be the name string of an IVRF instance.
<b>peer</b> <sup>1</sup>	(Optional) IKE peer filter. At least one of the following keywords and arguments must be used: <ul style="list-style-type: none"> <li><b>group</b> <i>string</i>—Unity group name filter of the IKE peer.</li> <li><b>hostname</b> <i>string</i>—Fully qualified domain name (FQDN) host name filter of the IKE peer.</li> <li><b>ipv4</b> <i>ipaddress</i>—IP address filter of the IKE peer.</li> <li><b>subnet</b> <i>subnet mask</i>—Range of IKE peer IP addresses.</li> <li><b>username</b> <i>string</i>—FQDN username filter of the IKE peer.</li> </ul>
<b>spi</b> <i>integer</i> <sup>1</sup>	(Optional) Security policy index (SPI) filter. The integer must be a 32-bit unsigned integer.
<b>reset</b>	(Optional) Deletes all crypto debug filters. <p><b>Note</b> It is suggested that you turn off all crypto global debugging before using this keyword; otherwise, your system may be flooded with debug messages.</p>

- Additional conditional filters (ipv4 address, subnet mask, username, hostname, group, connection-ID, flow-ID, SPI, FVRF, and IVRF) can be specified more than once by repeating the **debug crypto condition** command with any of the available filters.

## Defaults

Crypto conditional debugging is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.3(2)T	This command was introduced.

**Usage Guidelines** Before enabling the **debug crypto condition** command, you must decide what debug condition types (also known as debug filters) and values will be used. The volume of debug messages is dependent on the number of conditions you define.



**Note** Specifying numerous debug conditions may consume CPU cycles and have a negative effect on router performance.

To begin crypto conditional debugging, you must also enable at least one global crypto debug command—**debug crypto isakmp**, **debug crypto ipsec**, and **debug crypto engine**; otherwise, conditional debugging will not occur. This requirement helps to ensure that the performance of the router will not be impacted when conditional debugging is not being used.



**Note** Debug message filtering for hardware crypto engines is not supported.

**Examples** The following example shows how to display debug messages when the peer IP address is 10.1.1.1, 10.1.1.2, or 10.1.1.3 and when the connection-ID 2000 of crypto engine 0 is used. This example also shows how to enable global debug crypto CLIs and enable the **show crypto debug-condition** command to verify conditional settings.

```
Router# debug crypto condition connid 2000 engine-id 1
Router# debug crypto condition peer ipv4 10.1.1.1
Router# debug crypto condition peer ipv4 10.1.1.2
Router# debug crypto condition peer ipv4 10.1.1.3
Router# debug crypto condition unmatched
! Verify crypto conditional settings.
Router# show crypto debug-condition
```

```
Crypto conditional debug currently is turned ON
IKE debug context unmatched flag:ON
IPsec debug context unmatched flag:ON
Crypto Engine debug context unmatched flag:ON
```

```
IKE peer IP address filters:
10.1.1.1 10.1.1.2 10.1.1.3
```

```
Connection-id filters:[connid:engine_id]2000:1,
! Enable global crypto CLIs to start conditional debugging.
Router# debug crypto isakmp
Router# debug crypto ipsec
Router# debug crypto engine
```

The following example shows how to disable all crypto conditional settings via the **reset** keyword:

```
Router# debug crypto condition reset

! Verify that all crypto conditional settings have been disabled.

Router# show crypto debug-condition

Crypto conditional debug currently is turned OFF
IKE debug context unmatched flag:OFF
IPsec debug context unmatched flag:OFF
Crypto Engine debug context unmatched flag:OFF
```

#### Related Commands

Command	Description
<b>debug crypto condition unmatched</b>	Displays crypto conditional debug messages when context information is unavailable to check against debug conditions.
<b>show crypto debug-condition</b>	Displays crypto debug conditions that have already been enabled in the router.
<b>show crypto ipsec sa</b>	Displays the settings used by current SAs.
<b>show crypto isakmp sa</b>	Displays all current IKE SAs at a peer.

# debug crypto condition unmatched

To display crypto conditional debug messages when context information is unavailable to check against debug conditions, use the **debug crypto condition unmatched** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

**debug crypto condition unmatched [isakmp | ipsec | engine]**

**no debug crypto condition unmatched [isakmp | ipsec | engine]**

## Syntax Description

**isakmp | ipsec | engine** (Optional) One or more of these keywords can be enabled to display debug messages for the specified areas. If none of these keywords are entered, debug messages for all crypto areas will be shown.

## Defaults

Debug messages that do not have context information to match any debug conditions (filters) will not be printed.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.3(2)T	This command was introduced.

## Usage Guidelines

After the **debug crypto condition** command has been enabled, you can use the **debug crypto condition unmatched** command to define whether the debug output is being printed when no context information is available in the code to check against the debug filters. For example, if the crypto engine's connection-ID is the filter that the debug conditions are being checked against, the **debug crypto condition unmatched** command displays debug messages in the early negotiation phase when a connection-ID is unavailable to check against debug conditions.

## Examples

The following example shows how to enable debug messages for all crypto-related areas:

```
Router# debug crypto condition unmatched
```

## Related Commands

Command	Description
<b>debug crypto condition</b>	Defines conditional debug filters.
<b>show crypto debug-condition</b>	Displays crypto debug conditions that have already been enabled in the router.
<b>show crypto ipsec sa</b>	Displays the settings used by current SAs.
<b>show crypto isakmp sa</b>	Displays all current IKE SAs at a peer.

# debug crypto engine

To display debugging 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 Modes** Privileged EXEC

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 Rivest, Shamir, and Adelman (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 virtual private network (VPN) module driver to the VPN module hardware using a debug flag, use the **debug crypto engine accelerator logs** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**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 Modes

Privileged EXEC

## 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 **debug crypto engine accelerator logs** command 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
	<b>crypto engine accelerator</b>	Enables or disables the crypto engine accelerator if it exists.
	<b>show crypto engine accelerator logs</b>	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.
	<b>show crypto engine accelerator sa-database</b>	Prints active (in-use) entries in the platform-specific VPN module database.
	<b>show crypto engine configuration</b>	Displays the Cisco IOS crypto engine of your router.

# debug crypto error

To enable error debugging for a crypto area, use the **debug crypto error** command in privileged EXEC mode. To disable crypto error debugging, use the **no** form of this command.

**debug crypto {isakmp | ipsec | engine} error**

**no debug crypto {isakmp | ipsec | engine} error**

Syntax Description		
<b>isakmp</b>		Debug messages are shown for Internet Key Exchange (IKE)-related error operations only.
<b>ipsec</b>		Debug messages are shown for IP Security (IPSec)-related error operations only.
<b>engine</b>		Debug messages are shown for crypto engine-related error operations only.

**Defaults** Crypto error debugging is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.3(2)T	This command was introduced.

**Usage Guidelines** The **debug crypto error** command will display only error-related debug messages; that is, an error debug will not be shown if the operation is functioning properly.

This command should be used when debug conditions cannot be determined; for example, enable this command when a random, small subset of IKE peers is failing negotiation.



**Note** The global crypto command-line interfaces (CLIs) (the **debug crypto isakmp**, **debug crypto ipsec**, and **debug crypto engine** commands) will override the **debug crypto error** command. Thus, this command should not be used in conjunction with the global CLIs because you may overwhelm the router.



**Note** Debug message filtering for crypto hardware engines is not supported.

**Examples** The following example shows how to enable IPSec-related error messages:

```
Router# debug crypto error ipsec error
```

# debug crypto ha

To display crypto high availability debugging information, use the **debug crypto ha** command in privileged EXEC mode. To disable debugging messages, use the **no** form of this command.

**debug crypto ha**

**no debug crypto ha**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

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

**Examples** The following example is sample output from the **debug crypto ha** command:

```
Router# debug crypto ha

Active router:

Router# show debug

Cryptographic Subsystem:
  Crypto High Availability Manager debugging is on
vrf-lite-R1#
*Sep 28 21:27:50.899:Sending IKE Add SA Message
*Sep 28 21:27:50.899:HA Message 0:flags=0x01 len=394 HA_IKE_MSG_ADD_SA (2)
*Sep 28 21:27:50.899:  ID:04000003
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_MY_COOKIE (2) len 8
*Sep 28 21:27:50.899:    9B 1A 76 AA 99 11 1A 1F
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_HIS_COOKIE (3) len 8
*Sep 28 21:27:50.899:    E2 A2 A3 5F 53 1D EA 15
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_SRC (4) len 4
*Sep 28 21:27:50.899:    04 00 00 05
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_DST (5) len 4
*Sep 28 21:27:50.899:    04 00 00 03
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_PEER_PORT (6) len 2
*Sep 28 21:27:50.899:    01 F4
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_F_VRF (7) len 1
*Sep 28 21:27:50.899:    00
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_INIT_OR_RESP (8) len 1
*Sep 28 21:27:50.899:    00
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_NAT_DISCOVERY (9) len 1
*Sep 28 21:27:50.899:    02
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_IDTYPE (38) len 1
*Sep 28 21:27:50.899:    01
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_PROTOCOL (39) len 1
*Sep 28 21:27:50.899:    11
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_PORT (40) len 2
*Sep 28 21:27:50.899:    01 F4
*Sep 28 21:27:50.899:  attr HA_IKE_ATT_ADDR (41) len 4
```

```

*Sep 28 21:27:50.899: 04 00 00 05
*Sep 28 21:27:50.899: attr HA_IKE_ATT_MASK (42) len 4
*Sep 28 21:27:50.899: 00 00 00 00
*Sep 28 21:27:50.899: attr HA_IKE_ATT_ID_STR (44) len 4
*Sep 28 21:27:50.899: 00 00 00 00
*Sep 28 21:27:50.899: attr HA_IKE_ATT_PEERS_CAPABILITIES (11) len 4
*Sep 28 21:27:50.899: 00 00 07 7F
*Sep 28 21:27:50.899: attr HA_IKE_ATT_MY_CAPABILITIES (12) len 4
*Sep 28 21:27:50.899: 00 00 07 7F
*Sep 28 21:27:50.899: attr HA_IKE_ATT_STATE_MASK (13) len 4
*Sep 28 21:27:50.899: 00 00 27 FF
.
.
.

```

---

**Related Commands**

Command	Description
<b>debug crypto ipsec ha</b>	Enables debugging messages for IPsec high availability.
<b>debug crypto isakmp ha</b>	Enables debugging messages for ISAKMP high availability.

# debug crypto ipsec

To display IP Security (IPSec) events, use the **debug crypto ipsec** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug crypto ipsec**

**no debug crypto ipsec**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

**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.
```

Internet Key Exchange (IKE) asks for Service Provider Interfaces (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 from 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 about voice control messages that have been captured by the Voice DSP Control Message Logger, use the **debug crypto ipsec client ezvpn** command in privileged EXEC mode. To disable debugging output, 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.

**Defaults** No default behavior or values

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.2(4)YA	This command was introduced on Cisco 806, Cisco 826, Cisco 827, and Cisco 828 routers; Cisco 1700 series routers; and Cisco uBR905 and Cisco uBR925 cable access routers.
	12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
	12.3(4)T	This command was expanded to support the Easy VPN Remote feature.

**Usage Guidelines** To force the Voice DSP Control Message Logger to reestablish the virtual private network (VPN) connections, use the **clear crypto sa** and **clear crypto isakmp** commands to delete the IPsec security associations and Internet Key Exchange (IKE) connections, respectively.

**Examples** The following example shows debugging messages when the Voice DSP Control Message Logger is turned on and typical debugging messages that appear when the VPN tunnel is created:

```
Router# debug crypto ipsec client ezvpn

EzVPN debugging is on
router#
00:02:28: EZVPN(hw1): Current State: IPSEC_ACTIVE
00:02:28: EZVPN(hw1): Event: RESET
00:02:28: EZVPN(hw1): ezvpn_close
00:02:28: EZVPN(hw1): New State: CONNECT_REQUIRED
00:02:28: EZVPN(hw1): Current State: CONNECT_REQUIRED
00:02:28: EZVPN(hw1): Event: CONNECT
00:02:28: EZVPN(hw1): ezvpn_connect_request
00:02:28: EZVPN(hw1): New State: READY
00:02:29: EZVPN(hw1): Current State: READY
00:02:29: EZVPN(hw1): Event: MODE_CONFIG_REPLY
00:02:29: EZVPN(hw1): ezvpn_mode_config
00:02:29: EZVPN(hw1): ezvpn_parse_mode_config_msg
```

```

00:02:29: EZVPN: Attributes sent in message:
00:02:29: Address: 10.0.0.5
00:02:29: Default Domain: cisco.com
00:02:29: EZVPN(hw1): ezvpn_nat_config
00:02:29: EZVPN(hw1): New State: SS_OPEN
00:02:29: EZVPN(hw1): Current State: SS_OPEN
00:02:29: EZVPN(hw1): Event: SOCKET_READY
00:02:29: EZVPN(hw1): No state change
00:02:30: EZVPN(hw1): Current State: SS_OPEN
00:02:30: EZVPN(hw1): Event: MTU_CHANGED
00:02:30: EZVPN(hw1): No state change
00:02:30: EZVPN(hw1): Current State: SS_OPEN
00:02:30: EZVPN(hw1): Event: SOCKET_UP
00:02:30: ezvpn_socket_up
00:02:30: EZVPN(hw1): New State: IPSEC_ACTIVE

```

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

```

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:

```
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
<b>debug crypto ipsec</b>	Displays debugging messages for generic IPsec events.
<b>debug crypto isakmp</b>	Displays debugging messages for IKE events.

# debug crypto ipsec ha

To enable debugging messages for IP Security (IPSec) high availability, use the **debug crypto ipsec ha** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug crypto ipsec ha** [**detail**] [**update**]

**no debug crypto ipsec ha** [**detail**] [**update**]

## Syntax Description

<b>detail</b>	(Optional) Displays detailed debug information.
<b>update</b>	(Optional) Displays updates for inbound and outbound related data.

## Command Modes

Privileged EXEC

## Command History

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

## Examples

The following example is sample output of the **debug crypto ipsec ha** command for both the active and standby router:

```
Active Router
Router# debug crypto ipsec ha

Crypto IPSEC High Availability debugging is on

*Sep 29 17:03:01.851:IPSec HA (crypto_ha_ipsec_notify_add_sa):called

*Sep 29 17:03:01.851:IPSec HA (crypto_ha_ipsec_notify_add_sa):New IPsec SA added...
notifying HA Mgr

Standby Router
Router# debug crypto ipsec ha

Crypto IPSEC High Availability debugging is on
vrf-lite-R1#
*Sep 29 17:03:01.031:IPSec HA (crypto_ha_ipsec_mgr_rcv_add_sas):HA mgr wants to insert
the following bundle
*Sep 29 17:03:01.031:IPSec HA (crypto_ha_ipsec_mgr_rcv_add_sas):This SA Supports DPD
*Sep 29 17:03:01.031:IPSec HA (crypto_ha_ipsec_gen_sa):Sending Kei to IPSec num_kei 2
*Sep 29 17:03:01.039:IPSec HA (crypto_ha_ipsec_notify_add_sa):called

*Sep 29 17:03:01.039:IPSec HA (crypto_ha_ipsec_notify_add_sa):operation not performed as
standby ip 4.0.0.3
```

The following example is sample debug output with the **detail** keyword:

```
Active Router
*Sep 29 17:05:48.803:IPSec HA (crypto_ha_ipsec_mgr_set_state_common):called for vip
4.0.0.3
```

```
*Sep 29 17:06:11.655:IPSec HA (crypto_ha_ipsec_mgr_bulk_sync_sas):Bulk sync request from
standby for local addr 4.0.0.3
*Sep 29 17:06:44.059:IPSec HA (crypto_ha_ipsec_notify_add_sa):called

*Sep 29 17:06:44.059:IPSec HA (crypto_ha_ipsec_notify_add_sa):New IPsec SA added...
notifying HA Mgr
```

Standby Router

Router# **debug crypto ipsec ha detail**

Crypto IPSEC High Availability Detail debugging is on

vrf-lite-R1#

```
*Sep 29 17:06:44.063:IPSec HA (crypto_ha_ipsec_mgr_rcv_add_sas):HA mgr wants to insert
the following bundle
```

```
*Sep 29 17:06:44.063:IPSec HA (crypto_ha_ipsec_mgr_rcv_add_sas):This SA Supports DPD
```

```
*Sep 29 17:06:44.063:IPSec HA (crypto_ha_ipsec_gen_sa):Sending Kei to IPsec num_kei 2
```

```
*Sep 29 17:06:44.071:IPSec HA (crypto_ha_ipsec_notify_add_sa):called
```

```
*Sep 29 17:06:44.071:IPSec HA (crypto_ha_ipsec_notify_add_sa):operation not performed as
standby ip 4.0.0.3
```

The following example is sample debug output with the **update** keyword:

Active Router

```
*Sep 29 17:27:30.839:IPSec HA(check_and_send_replay_update):Replay triggered update
seq_num 1000 last-sent 0 dir inbound
```

```
*Sep 29 17:27:30.839:IPSec HA(create_update_struct):Sending inbound update
```

```
*Sep 29 17:27:30.839:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998772, New replay 1000
```

```
*Sep 29 17:29:30.883:IPSec HA(check_and_send_replay_update):Replay triggered update
seq_num 2000 last-sent 1000 dir inbound
```

```
*Sep 29 17:29:30.883:IPSec HA(create_update_struct):Sending inbound update
```

```
*Sep 29 17:29:30.883:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998624, New replay 2000
```

```
*Sep 29 17:30:30.899:IPSec HA(check_and_send_replay_update):Replay triggered update
seq_num 3000 last-sent 2000 dir inbound
```

```
*Sep 29 17:30:30.899:IPSec HA(create_update_struct):Sending inbound update
```

```
*Sep 29 17:30:30.899:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998476, New replay 3000
```

```
*Sep 29 17:32:30.943:IPSec HA(check_and_send_replay_update):Replay triggered update
seq_num 4000 last-sent 3000 dir inbound
```

```
*Sep 29 17:32:30.943:IPSec HA(create_update_struct):Sending inbound update
```

```
*Sep 29 17:32:30.943:IPSec HA(send_update_struct):
  Outbound - New KB 0, New replay 0
  Inbound - New KB 3998327, New replay 4000
```

Standby Router

```
*Sep 29 17:27:28.887:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):called
```

```
*Sep 29 17:27:28.887:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):UPDATING INBOUND SA:ip =
4.0.0.3, protocol = 50, spi = B8A47EC9,
  NEW KB LIFE = 3998772,
  NEW REPLAY WINDOW START = 1000,
```

```
*Sep 29 17:29:28.915:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):called

*Sep 29 17:29:28.915:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):UPDATING INBOUND SA:ip =
4.0.0.3, protocol = 50, spi = B8A47EC9,
    NEW KB LIFE = 3998624,
    NEW REPLAY WINDOW START = 2000,

*Sep 29 17:30:28.939:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):called

*Sep 29 17:30:28.939:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):UPDATING INBOUND SA:ip =
4.0.0.3, protocol = 50, spi = B8A47EC9,
    NEW KB LIFE = 3998476,
    NEW REPLAY WINDOW START = 3000,

*Sep 29 17:32:28.955:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):called

*Sep 29 17:32:28.955:IPSec HA(crypto_ha_ipsec_mgr_rcv_update_sa):UPDATING INBOUND SA:ip =
4.0.0.3, protocol = 50, spi = B8A47EC9,
    NEW KB LIFE = 3998327,
    NEW REPLAY WINDOW START = 4000,
```

**Related Commands**

Command	Description
<b>debug crypto ha</b>	Displays crypto high availability debugging information.
<b>debug crypto isakmp ha</b>	Enables debugging messages for ISAKMP high availability.

# debug crypto ipv6 ipsec

To display IP Security (IPSec) events for IPv6 networks, use the **debug crypto ipv6 ipsec** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug crypto ipv6 ipsec**

**no debug crypto ipv6 ipsec**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debugging for IPv6 IPSec events is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.3(4)T	This command was introduced.

**Usage Guidelines** Use this command to display IPSec events while setting up or removing policy definitions during OSPF configuration.

Related Commands	Command	Description
	<b>debug crypto engine</b>	Displays debugging messages about crypto engines, which perform encryption and decryption.
	<b>debug crypto ipv6 packet</b>	Displays debug messages for IPv6 packets allowing you to see the contents of packets outbound from a Cisco router when the remote node is not a Cisco node.
	<b>debug crypto socket</b>	Displays communication between the client and IPSec during policy setup and removal processes.
	<b>debug ipv6 ospf authentication</b>	Shows the interaction between OSPF and IPSec, including creation or removal of policies.

# debug crypto ipv6 packet

To display the contents of IPv6 packets, use the **debug crypto ipv6 packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug crypto ipv6 packet**

**no debug crypto ipv6 packet**

**Syntax Description** This command has no arguments or keywords.

**Defaults** Debugging for IPv6 IPsec packets is not enabled.

**Command Modes** Privileged EXEC

Command History	Release	Modification
	12.3(4)T	This command was introduced.

**Usage Guidelines** Consult Cisco Technical Support before using this command.

Use this command to display the contents of IPv6 packets. This command is useful when the remote node is not a Cisco device and communication between the Cisco and non-Cisco router cannot be established. This command enables you to look at the contents of the packets outbound from the Cisco router.



**Caution**

This command examines the content of every IPv6 packet and will affect network performance.

**Examples** This example shows the output of each packet when the **debug crypto ipv6 packet** command is enabled.

```
Router# debug crypto ipv6 packet
```

```
Crypto IPv6 IPSEC packet debugging is on
```

```
Router#
*Oct 30 16:57:06.330:
IPSECv6:before Encapsulation of IPv6 packet:
0E37A7C0:                6E000000 00285901                n....(Y.
0E37A7D0:FE800000 00000000 020A8BFF FED42C1D  ~.....~T,.
0E37A7E0:FF020000 00000000 00000000 00000005  .....
0E37A7F0:03010028 01010104 00000001 8AD80000  ...(.X...
0E37A800:00000006 01000013 000A0028 0A0250CF  .....(..PO
0E37A810:01010104 0A0250CF  .....PO
*Oct 30 16:57:06.330:
IPSECv6:Encapsulated IPv6 packet
:
0E37A7B0:6E000000 00403301 FE800000 00000000  n....@3.~.....
0E37A7C0:020A8BFF FED42C1D FF020000 00000000  ....~T,.....
```

```

0E37A7D0:00000000 00000005 59040000 000022B8 .....Y....."8
0E37A7E0:0000001A 38AB1ED8 04C1C6FB FF1248CF ....8+.X.AF{.HO
0E37A7F0:03010028 01010104 00000001 8AD80000 ...(.X..
0E37A800:00000006 01000013 000A0028 0A0250CF .....(..PO
0E37A810:01010104 0A0250CF .....PO
*Oct 30 16:57:11.914:
IPSECv6:Before Decapsulation of IPv6 packet
:
0E004A50:                6E000000 00403301                n....@3.
0E004A60:FE800000 00000000 023071FF FE7FE81D ~.....0q.~.h.
0E004A70:FF020000 00000000 00000000 00000005 .....
0E004A80:59040000 000022B8 00001D88 F5AC68EE Y....."8....u,hn
0E004A90:1AC00088 947C6BF2 03010028 0A0250CF .@...|kr...(..PO
0E004AA0:00000001 E9080000 00000004 01000013 ...i.....
0E004AB0:000A0028 0A0250CF 01010104 01010104 ...(..PO.....
0E004AC0:
*Oct 30 16:57:11.914:
IPSECv6:Decapsulated IPv6 packet
:
0E004A70:6E000000 00285901 FE800000 00000000 n....(Y.~.....
0E004A80:023071FF FE7FE81D FF020000 00000000 .0q.~.h.....
0E004A90:00000000 00000005 03010028 0A0250CF .....(..PO
0E004AA0:00000001 E9080000 00000004 01000013 ...i.....
0E004AB0:000A0028 0A0250CF 01010104 01010104 ...(..PO.....
0E004AC0:
*Oct 30 16:57:16.330:
IPSECv6:before Encapsulation of IPv6 packet:
0E003DC0:                6E000000 00285901                n....(Y.
0E003DD0:FE800000 00000000 020A8BFF FED42C1D ~.....~T,.
0E003DE0:FF020000 00000000 00000000 00000005 .....
0E003DF0:03010028 01010104 00000001 8AD80000 ...(.X..
0E003E00:00000006 01000013 000A0028 0A0250CF .....(..PO
0E003E10:01010104 0A0250CF .....PO
*Oct 30 16:57:16.330:
IPSECv6:Encapsulated IPv6 packet
:
0E003DB0:6E000000 00403301 FE800000 00000000 n....@3.~.....
0E003DC0:020A8BFF FED42C1D FF020000 00000000 ...~T,.
0E003DD0:00000000 00000005 59040000 000022B8 .....Y....."8
0E003DE0:0000001B F8E3C4E2 4CC4B690 DDF32B5C ...xcDbLD6.]s\
0E003DF0:03010028 01010104 00000001 8AD80000 ...(.X..
0E003E00:00000006 01000013 000A0028 0A0250CF .....(..PO
0E003E10:01010104 0A0250CF .....PO

```

**Related Commands**

Command	Description
<b>debug crypto engine</b>	Displays debugging messages about crypto engines, which perform encryption and decryption.
<b>debug crypto ipv6 ipsec</b>	Displays IPSec events for IPv6 networks.
<b>debug crypto socket</b>	Displays communication between the client and IPSec during policy setup and removal processes.
<b>debug ipv6 ospf authentication</b>	Shows the interaction between OSPF and IPSec, including creation or removal of policies.

# 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 <b>aaa</b> 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
<b>crypto isakmp profile</b>	Defines an ISAKMP profile and audits IPSec user sessions.
<b>crypto map (global IPSec)</b>	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 isakmp ha

To enable debugging messages for Internet Security Association and Key Management Protocol (ISAKMP) high availability, use the **debug crypto isakmp ha** command in privileged EXEC mode. To disable debugging messages, use the **no** form of this command.

**debug crypto isakmp ha** [detail]

**no debug crypto isakmp ha** [detail]

## Syntax Description

**detail** (Optional) Displays detailed debug information.

## Command Modes

Privileged EXEC

## Command History

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

## Examples

The following is sample output for a standby router from the **debug crypto isakmp ha** command:

```
Active Router
no debug message
```

```
Standby Router
Router# debug crypto isakmp ha
```

```
Crypto ISAKMP High Availability debugging is on
vrf-lite-R1#
*Sep 28 21:54:41.815:IKE HA:(4.0.0.3) Adding STANDBY IKE SA
```

```
*Sep 28 21:54:41.843:IKE HA:Create peer struct for local 4.0.0.3 remote 4.0.0.5 & locked
*Sep 28 21:54:41.843:IKE HA:IKE SA inserted on standby with src = 4.0.0.5, dst = 4.0.0.3
```

The following sample output is displayed when the **detail** keyword is issued. (Note that debug output without issuing the **detail** keyword is the same as the debug output with the **detail** keyword.)

```
Active Router
Router# debug crypto isakmp ha detail
```

```
Crypto ISAKMP High Availability detailed debugging is on
vrf-lite-R1#
*Sep 29 16:59:15.035:IKE HA:IKE SA is already failed over
```

```
Standby Router
Router# debug crypto isakmp ha detail
```

```
Crypto ISAKMP High Availability detailed debugging is on
vrf-lite-R2#
*Sep 29 16:59:14.371:IKE HA:(4.0.0.3) Adding STANDBY IKE SA
```

```
*Sep 29 16:59:14.411:IKE HA:Create peer struct for local 4.0.0.3 remote 4.0.0.5 & locked
*Sep 29 16:59:14.411:IKE HA:IKE SA inserted on standby with src = 4.0.0.5, dst = 4.0.0.3
```

**Related Commands**

<b>Command</b>	<b>Description</b>
<b>debug crypto ha</b>	Displays crypto high availability debugging information.
<b>debug crypto ipsec ha</b>	Enables debugging messages for IPSec high availability.

# 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. To disable debugging output, use the **no** form of this command.

**debug crypto key-exchange**

**no debug crypto key-exchange**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

**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
<b>debug crypto sesgmt</b>	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 debugging output, 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 Release 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
	<b>show crypto mib ipsec flowmib history failure size</b>	Displays the size of the IPSec failure history table.
	<b>show crypto mib ipsec flowmib history tunnel size</b>	Displays the size of the IPSec tunnel history table.
	<b>show crypto mib ipsec flowmib version</b>	Displays the IPSec Flow MIB version used by the router.

# debug crypto pki messages

To display debugging 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 Modes** Privileged EXEC

Command History	Release	Modification
	12.0	This command was introduced.

**Usage Guidelines** The **debug crypto pki messages** command displays messages about the actual data being sent and received during public key infrastructure (PKI) transactions. This command is internal for use by Cisco support personnel.

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

**Examples** The following is sample output from 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
.
.
.

```

**Related Commands**

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