debug clns esis events through debug dbconn tcp

- debug clns esis events through debug dbconn tcp, page 1

debug clns esis events through debug dbconn tcp

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 **debugclnsesisevents** 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 **debugclnsesisevents** 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 **debugclnsesispackets** 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**debugclnsesispackets** 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.O906.4023.00, 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 **debugclnsevents** 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**debugclnsevents** 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 igrp packets

To display debugging information on all ISO-IGRP routing activity, use the **debugclnsigrppackets** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns igrp packets

no debug clns igrp packets

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
10.0	This command was introduced.
12.2(13)T	This command is no longer supported in Cisco IOS Mainline or Technology-based (T) releases. It may continue to appear in Cisco IOS 12.2S-family releases.

Examples

The following is sample output from the**debugclnsigrppackets** command:

```
Router# debug clns igrp packets
ISO-IGRP: Hello sent on Ethernet3 for DOMAIN_green1
ISO-IGRP: Received hello from 39.0001.3333.3333.3333.00, (Ethernet3), ht 51
ISO-IGRP: Originating level 1 periodic update
ISO-IGRP: Advertise dest: 2222.2222.2222
ISO-IGRP: Sending update on interface: Ethernet3
ISO-IGRP: Originating level 2 periodic update
ISO-IGRP: Advertise dest: 0001
ISO-IGRP: Sending update on interface: Ethernet3
ISO-IGRP: Received update from 3333.3333.3333 (Ethernet3)
ISO-IGRP: Opcode: area
ISO-IGRP: Received level 2 adv for 0001 metric 1100
ISO-IGRP: Opcode: station
ISO-IGRP: Received level 1 adv for 3333.3333.3333 metric 1100
```

The following line indicates that the router is sending a hello packet to advertise its existence in the DOMAIN_green1 domain:

```
ISO-IGRP: Hello sent on Ethernet3 for DOMAIN_green1
```

The following line indicates that the router received a hello packet from a certain network service access point (NSAP) on Ethernet interface 3. The hold time for this information is 51 seconds.

```
ISO-IGRP: Received hello from 39.0001.3333.3333.3333.00, (Ethernet3), ht 51
```

The following lines indicate that the router is generating a Level 1 update to advertise reachability to destination NSAP 2222.2222.2222 and that it is sending that update to all systems that can be reached through Ethernet interface 3:

```
ISO-IGRP: Originating level 1 periodic update
ISO-IGRP: Advertise dest: 2222.2222.2222
ISO-IGRP: Sending update on interface: Ethernet3
```

The following lines indicate that the router is generating a Level 2 update to advertise reachability to destination area 1 and that it is sending that update to all systems that can be reached through Ethernet interface 3:

```
ISO-IGRP: Originating level 2 periodic update
ISO-IGRP: Advertise dest: 0001
ISO-IGRP: Sending update on interface: Ethernet3
```

The following lines indicate that the router received an update from NSAP 3333.3333.3333 on Ethernet interface 3. This update indicated the area that the router at this NSAP could reach.

```
ISO-IGRP: Received update from 3333.3333.3333 (Ethernet3)
ISO-IGRP: Opcode: area
```

The following lines indicate that the router received an update advertising that the source of that update can reach area 1 with a metric of 1100. A station opcode indicates that the update included system addresses.

```
ISO-IGRP: Received level 2 adv for 0001 metric 1100
ISO-IGRP: Opcode: station
```

debug clns packet

To display information about packet receipt and forwarding to the next interface, use the **debugclnspacket** 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**debugclnspacket** 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 **debugclnsrouting** 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 **debugclnsrouting** 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 **debugclsmessage**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 **debugclsmessage**command displays the primitives (state), selector, header length, and data size.

Examples

The following is sample output from the**debugclsmessage**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.Req 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
debug fras error	Displays information about FRAS protocol errors.
debug fras message	Displays general information about FRAS messages.
debug fras state	Displays information about FRAS data-link control state changes.

debug cls vdlc

To display information about Cisco Link Services (CLS) Virtual Data Link Control (VDLC), use the **debugclsvdlc**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 **debugclsmessage** command displays primitive state transitions, selector, and source and destination MAC and service access points (SAPs).

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

Caution Use the **debugclsvdlc** command with caution because it can generate a substantial amount of output.

Examples

The following messages are sample output from the **debugclsvdlc** command. In the following scenario, the systems network architecture (SNA) service point--also called *nativeservicepoint(*NSP*)*--is setting up two connections through VDLC and data-link switching (DLSw): one from NSP to VDLC and one from DLSw to VDLC. VDLC joins the two.

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

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

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

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

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

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

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

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

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

```
VDLC: Req Open Stn Req PSap 0x7B68C0, port 0x7992A0
4000.1060.1000(04)->4000.05d2.0001(0C)
```

```
VDLC: two-way connection established
VDLC: 4000.1060.1000(04)->4000.05d2.0001(0C): VDLC_IDLE->VDLC_OPEN
```

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

```
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: CEP to CEP ID_REQ 4000.05d2.0001(0C)->4000.1060.1000(04)
VDLC: CEP to CEP ID_RSP 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_OPEN->VDLC_XID_RSP_PENDING
VDLC: 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:

```
VDLC: CEP to CEP CONNECT_REQ 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: 4000.05d2.0001(0C)->4000.1060.1000(04): VDLC_XID_RSP_PENDING->VDLC_OPEN
```

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

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

The following messages show the data flow:

```
VDLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)
.
.
.
VDLC: DATA 4000.1060.1000(04)->4000.05d2.0001(0C)
VDLC: DATA 4000.05d2.0001(0C)->4000.1060.1000(04)
```

Related Commands

Command	Description
debug cls message	Displays information about CLS messages.

debug cme-xml

To generate debug messages for the Cisco Unified CallManager Express XML application, use the **debugcme-xml** command in privileged EXEC mode. To disable debugging, use the **no** form of the command.

debug cme-xml

no debug cme-xml

Syntax Description

This command has no keywords or arguments.

Command Modes

Privileged EXEC

Command History

Cisco IOS Release	Modification
12.4(4)XC	This command was introduced.
12.4(9)T	This command was integrated into Cisco IOS Release 12.4(9)T.

Usage Guidelines

The **showfb-its-log** command displays the contents of the XML event table.

Examples

The following example shows the progress of an XML request that has been sent to Cisco Unified CallManager Express:

```
Router# debug cme-xml
*Aug 5 06:27:25.727: CME got a raw XML message.
*Aug 5 06:27:25.727: doc 0x63DB85E8, doc->doc_type 3, req 0x655FDCD0
*Aug 5 06:27:25.727: CME extracted a XML document
*Aug 5 06:27:25.727: Response buffer 0x63DCFD58, len = 4096
*Aug 5 06:27:25.727: First Tag ID SOAP_HEADER_TAG_ID 58720257
*Aug 5 06:27:25.727: First Attribute ID SOAP_ENV_ATTR 50331649
*Aug 5 06:27:25.727: cme_xml_process_soap_header
*Aug 5 06:27:25.727: cme_xml_process_soap_body
*Aug 5 06:27:25.731: cme_xml_process_axl
*Aug 5 06:27:25.731: cme_xml_process_request
*Aug 5 06:27:25.731: cme_xml_process_ISgetGlobal
*Aug 5 06:27:25.731: CME XML sent 811 bytes response.
```

Related Commands

Command	Description
show fb-its-log	Displays Cisco Unified CallManager Express XML API information.

debug cns config

To turn on debugging messages related to the Cisco Networking Services (CNS) configuration agent, use the **debugcnsconfig** 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

agent	Displays debugging messages related to the CNS configuration agent.
all	Displays all debugging messages.
connection	Displays debugging messages related to configuration connections.
notify	Displays debugging messages related to CNS configurations.

Command Default

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.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

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=minna1,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=minna1
00:04:09: Message finished 150 readend
00:04:09: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 82
 00:04:10: %SYS-5-CONFIG_I: Configured from console by console
```

Related Commands

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

debug cns events

To turn on debugging messages related to the Cisco Networking Services (CNS) Event Gateway, use the **debugcnsevent** 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

agent	Displays debugging messages related to the event agent.
all	Displays all debugging messages.
connection	Displays debugging messages related to event connections.
subscriber	Displays debugging messages related to subscribers.

Command Default

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.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.

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

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

debug cns exec

To display debugging messages about Cisco Networking Services (CNS) exec agent services, use the **debugcnsexec** 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

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

Command Default

Debugging output is disabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(1)	This command was introduced.
12.2(25)S	This command was integrated into Cisco IOS Release 12.2(25)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

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
cns exec	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

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

Command Default

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.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.
12.2(33)SRB	This command was integrated into Cisco IOS Release 12.2(33)SRB.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

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

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

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

debug cns xml

To turn on debugging messages related to the Cisco Networking Services (CNS) eXtensible Markup Language (XML) parser, use the **debugcnsxml** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cns xml {**all**| **decode**| **dom**| **parser**}

no debug cns xml {**all**| **decode**| **dom**| **parser**}

Syntax Description

all	Displays all CNS debugging.
decode	Reports usage of common XML decoding library functions by applications and reports the decoded contents.
dom	Displays failures in the Document Object Model (DOM) infrastructure messages tree built by the XML parser.
parser	Displays failures and progress of the parsing of an XML message by the XML parser.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced. This command replaces the **debugcnsxml-parser** command.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.
12.2(33)SXI	This command was integrated into Cisco IOS Release 12.2(33)SXI.

Examples

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

```
Router# debug cns xml parser
```

```
00:12:05: Registering tag <config-server>
00:12:05: Registering tag <server-info>
00:12:05: Registering tag <ip-address>
00:12:05: Registering tag <web-page>
00:12:05: Registering tag <config-event>
00:12:05: Registering tag <identifier>
00:12:05: Registering tag <config-id>
00:12:05: Registering tag <config-data>
00:12:05: Registering tag <cli>
00:12:05: Registering tag <error-info>
00:12:05: Registering tag <error-message>
00:12:05: Registering tag <line-number>
00:12:05: Registering tag <config-write>
00:12:05: Registering tag <exec-cmd-event>
00:12:05: Registering tag <identifier-exec>
00:12:05: Registering tag <event-response>
00:12:05: Registering tag <reply-subject>
00:12:05: Registering tag <server-response>
00:12:05: Registering tag <ip-address-exec>
00:12:05: Registering tag <port-number>
00:12:05: Registering tag <url>
00:12:05: Registering tag <cli-exec>
00:12:05: Registering tag <config-pwd>
00:12:06: Pushing tag <config-data> on to stack
00:12:06: open tag is <config-data>
00:12:06: Pushing tag <config-id> on to stack
00:12:06: open tag is <config-id>
00:12:06: Popping tag <config-id> off stack
00:12:06: close tag is </config-id>
00:12:06: Pushing tag <cli> on to stack
00:12:06: open tag is <cli>
00:12:06: Popping tag <cli> off stack
00:12:06: close tag is </cli>
00:12:06: Popping tag <config-data> off stack
00:12:06: close tag is </config-data>
00:12:06: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 96
```

Related Commands

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

debug cns xml-parser

Note Effective with Cisco IOS Release 12.3(2)T, the **debug cns xml-parser** command is replaced by the **debug cns xml** command. See the **debug cns xml** command for more information.

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.

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.
12.3(2)T	This command was replaced by the **debug cns xml** command.

Examples

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

```
Router# debug cns xml-parser

00:12:05: Registering tag <config-server>
00:12:05: Registering tag <server-info>
00:12:05: Registering tag <ip-address>
00:12:05: Registering tag <web-page>
00:12:05: Registering tag <config-event>
00:12:05: Registering tag <identifier>
00:12:05: Registering tag <config-id>
00:12:05: Registering tag <config-data>
00:12:05: Registering tag <cli>
00:12:05: Registering tag <error-info>
00:12:05: Registering tag <error-message>
00:12:05: Registering tag <line-number>
00:12:05: Registering tag <config-write>
00:12:05: Registering tag <exec-cmd-event>
00:12:05: Registering tag <identifier-exec>
00:12:05: Registering tag <event-response>
```

```
00:12:05: Registering tag <reply-subject>
00:12:05: Registering tag <server-response>
00:12:05: Registering tag <ip-address-exec>
00:12:05: Registering tag <port-number>
00:12:05: Registering tag <url>
00:12:05: Registering tag <cli-exec>
00:12:05: Registering tag <config-pwd>
00:12:06: Pushing tag <config-data> on to stack
00:12:06: open tag is <config-data>
00:12:06: Pushing tag <config-id> on to stack
00:12:06: open tag is <config-id>
00:12:06: Popping tag <config-id> off stack
00:12:06: close tag is </config-id>
00:12:06: Pushing tag <cli> on to stack
00:12:06: open tag is <cli>
00:12:06: Popping tag <cli> off stack
00:12:06: close tag is </cli>
00:12:06: Popping tag <config-data> off stack
00:12:06: close tag is </config-data>
00:12:06: %CNS-4-NOTE: SUCCESSFUL_COMPLETION
-Process= "CNS Initial Configuration Agent", ipl= 0, pid= 96
```

Related Commands

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

debug compress

To debug compression, enter the **debugcompress**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.

Command Default

Disabled.

Command Modes

Privileged EXEC

Command History

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

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 **debugcompress** 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
```

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

Table 1: 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
debug frame-relay	Displays debugging information about the packets that are received on a Frame Relay interface.
debug ppp	Displays information on traffic and exchanges in an internetwork implementing the PPP.
show compress	Displays compression statistics.
show diag	Displays hardware information including DRAM, SRAM, and the revision-level information on the line card.

debug condition

To filter debugging output for certain **debug** commands on the basis of 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 {**called** *dial-string*| **caller** *dial-string*| **calling** *tidimsi string*| **domain** *domain-name*| **interface** *interface-id*| **ip** *ip-address*| **mac-address** *hexadecimal-MAC-address*| **portbundle ip** *ip-address* **bundle** *bundle-number*| **session-id** *session-number*| **username** *username*| **vcid** *vc-id*| **vlan** *vlan-id*}

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

Syntax Description

called *dial-string*	Filters output on the basis of the called party number.
caller *dial-string*	Filters output on the basis of the calling party number.
calling *tidi/imsi-string*	Filters debug messages for 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.
domain *domain-name*	Filters output on the basis of the specified domain.
interface *interface-id*	Filters output on the basis of the specified interface ID.
ip *ip-address*	Filters output on the basis of the specified IP address.
mac-address *hexadecimal-mac-address*	Filters messages on the specified MAC address.
portbundle ip *ip-address*	Filters output on the basis of the port-bundle host key (PBHK) that uniquely identifies the session.
bundle *bundle-number*	Specifies the port bundle.
session-id *session-number*	Filters output on the specified Intelligent Service Architecture (ISA) session identifier.
username *username*	Filters output on the basis of the specified username.
vcid *vc-id*	Filters output on the basis of the specified VC ID.
vlan *vlan-id*	Filters output on the basis of the specified VLAN ID.
condition-id	ID number of the condition to be removed.

all	Removes all debugging conditions, and conditions specified by the **debug condition interface** command. Use this keyword to disable conditional debugging and reenable debugging for all interfaces.

Command Default

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 modified. The **vcid** and **ip** keywords were added 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 modified. Support was added on the GGSN.
12.3(8)T	This command was modified. The **calling** keyword and *tid/imsi-string* argument were added.
12.2(28)SB	This command was modified. The ability to filter output on the following conditions was added: domain, MAC address, PBHK, and ISA session ID.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
15.2(2)T	This command was modified. The **vlan** *vlan-id* keyword and argument and the **interface** *interface-id* keyword and 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 generated only 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 following components are supported for Intelligent Service Architecture (ISA) distributed conditional debugging:

- Authentication, authorization, and accounting (AAA) and RADIUS
- ATM components
- Feature Manager
- Policy Manager
- PPP
- PPP over Ethernet (PPPoE)
- Session Manager
- Virtual Private Dialup Network (VPDN)

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

Examples

In the following example, the router displays debugging messages only for interfaces that use a username of user1. The condition identifier displayed after the command is entered identifies this particular condition.

```
Router# debug condition username user1
Condition 1 set
```

Examples

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
```

The following example enables other debugging commands. These debugging commands will only display information 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 on which VC 1000 is established:

```
Router(config)# interface serial3/1/0
Router(config-if)# shut
```

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

```
01:15:59: AToM MGR [13.13.13.13, 1000]: Event local down, state changed from established
to remote ready
01:15:59: AToM MGR [13.13.13.13, 1000]: Local end down, vc is down
01:15:59: AToM SMGR [13.13.13.13, 1000]: Processing imposition update, vc_handle 6227BCF0,
 update_action 0, remote_vc_label 18
01:15:59: AToM SMGR [13.13.13.13, 1000]: Imposition Disabled
01:15:59: AToM SMGR [13.13.13.13, 1000]: Processing disposition update, vc_handle 6227BCF0,
 update_action 0, local_vc_label 755
01:16:01:%LINK-5-CHANGED: Interface Serial3/1/0, changed state to administratively down
01:16:02:%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial3/1/0, changed state to down
```

Related Commands

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

debug condition application voice

To display debugging messages for only the specified VoiceXML application, use the **debugconditionapplicationvoice**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*

Syntax Description

application-name	Name of the VoiceXML application for which you want to display all enabled debugging messages.

Command Default

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

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

- This command filters debugging output only for the **debugvxml**and **debughttpclient** commands, except that it does not filter output for the **debugvxmlerror**, **debugvxmlbackground**, **debughttpclienterror**, or **debughttpclientbackground** commands. It does not filter messages for any other debug commands such as the **debugvoipivr** command or the **debugvoiceivr** command.
- This command filters debugging output for all VoiceXML applications except the application named in the command. When this command is configured, the gateway displays debugging messages only for the specified VoiceXML application.
- To filter debugging output with this command, the <cisco-debug> element must be enabled in the VoiceXML document. For more information about the <cisco-debug> element, refer to the Cisco VoiceXML Programmer's Guide.
- To see debugging output for VoiceXML applications, you must first configure global **debug** commands such as the **debugvxml**command or the **debughttpclient**command. If no global **debug** commands are turned on, you do not see debugging messages even if the **debugconditionapplicationvoice** command is configured and the <cisco-debug> element is enabled in the VoiceXML document.
- This command can be configured multiple times to display output for more than one application.
- To see which debug conditions have been set, use the **showdebugcondition** command.

Examples

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

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

Related Commands

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

debug condition glbp

To display debugging messages about Gateway Load Balancing Protocol (GLBP) conditions, use the **debugconditionglbp** 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

interface-type	Interface type for which output is displayed.
interface-number	Interface number for which output is displayed.
group	GLBP group number in the range from 0 to 1023.
forwarder	(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.
12.2(17b)SXA	This command was integrated into Cisco IOS Release 12.2(17b)SXA.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following is sample output from the **debugconditionglbp**command:

```
Router# debug condition glbp fastethernet 0/0 10 1
Condition 1 set
5d23h: Fa0/0 GLBP10.1 Debug: Condition 1, glbp Fa0/0 GLBP10.1 triggered, count 1
```

Related Commands

Command	Description
debug glbp errors	Displays debugging messages about GLBP errors.
debug glbp events	Displays debugging messages about GLBP events.

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

debug condition interface

To limit output for some debug commands on the basis of the interface, virtual circuit (VC), or VLAN, use the **debugconditioninterface** 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*| *vpivci*}] [**vlan-id** *vlan-id*]

no debug condition interface *interface-type interface-number* [**dlci** *dlci*] [**vc** {*vci*| *vpivci*}] [**vlan-id** *vlan-id*]

Syntax Description

interface-type interface-number	Interface type and number. For more information, use the question mark (?) online help function.
dlci *dlci*	(Optional) Specifies the data-link connection identifier (DLCI), if the interface to be debugged is a Frame Relay-encapsulated interface.
vc {*vci* | *vpi*/*vci*}	(Optional) Specifies the virtual channel identifier (VCI) or virtual path identifier/virtual channel identifier (VPI/VCI) pair, if the interface to be debugged is an ATM-encapsulated interface.
vlan-id *vlan-id*	(Optional) Specifies the VLAN ID, if the interface to be debugged is ATM, Ethernet, Fast Ethernet, or Gigabit Ethernet interface.

Command Default

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

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(11)T	This command was introduced.
12.0(28)S	This command was modified. The **dlci** and **vc** keywords were added for additional Frame Relay and ATM functionality.
12.2(25)S	This command was modified. It was integrated into Cisco IOS Release 12.2(25)S.
12.2(27)SBC	This command was modified. It was integrated into Cisco IOS Release 12.2(27)SBC.

Release	Modification
12.2(28)SB	This command was modified. It was integrated into Cisco IOS Release 12.2(28)SB, and the ability to filter debug output on the basis of VLAN ID was added.
12.4(9)T	This command was modified. It was integrated into Cisco IOS Release 12.4(9)T.
12.2(33)SRE	This command was modified. It was integrated into Cisco IOS Release 12.2(33)SRE.
Cisco IOS XE 2.5	This command was modified. It was integrated into Cisco IOS XE Release 2.5.

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 disbaled 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 this command performs the following functions:

- Disables the **debugconditioninterface** 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 **debugcondition** command, debugging output will still be generated for the interface.
- If some other **debugcondition** 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 **debugcondition** command. Some commands generate output whenever they are enabled, regardless of whether they meet any conditions. The commands that are affected by the **debugcondition** 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 **debugcondition**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 shows how to enable 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 209.165.201.2 255.255.255.0
Router(config-if)# pvc 255/62610
Router(config-if-atm-vc)# no shutdown
Router(config-if)# exit
Router(config)# exit
2w3d: %SYS-5-CONFIG_I: Configured from console by console
Router# debug condition interface atm 2/0 vc 255/62610
Condition 1 set
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
```

The following example shows how to enable Frame Relay conditional debugging on Frame Relay DLCI 105:

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

The following example shows how to disable the conditional debugging on VC. A warning message is displayed when the last condition is removed.

```
Router> enable
Router# no debug condition interface atm 1/0 vc 4335
```

```
This condition is the last interface 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
debug condition	Limits output for some debug commands on the basis of specific conditions.
debug frame-relay	Displays debugging information about the packets received on a Frame Relay interface.
show debug condition	Displays the debugging filters that have been enabled for VoiceXML. applications, ATM-enabled interfaces, or Frame Relay interfaces

debug condition match-list

To run a filtered debug on a voice call, use the **debugconditionmatch-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

number	Numeric label that uniquely identifies the match list. Range is 1 to 16. The number for the match list is set using the **callfiltermatch-list** command.
exact-match	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.
partial-match	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.

Command Default

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 **debugconditionmatch-list1** 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_cond_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_cond_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_stream_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_stream_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_ind:
 Call ID=98357, guid=3C0B9468-15C8-11D4-8013-000A8A389BA8
```

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

Table 2: 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
call filter match-list voice	Creates a call filter match list for debugging voice calls.
debug call filter inout	Displays the debug trace inside the GCFM.
show call filter match-list	Displays call filter match lists.

debug condition standby

To filter the output of the **debugstandby**command on the basis of interface and Hot Standby Router Protocol (HSRP) group number, use the**debugconditionstandby**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

interface	Filters output on the basis of the interface.
group-number	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.

Command Default

All debugging messages for the **debugstandby**command aregenerated.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(2)	This command was introduced.

Usage Guidelines

Use the **debugconditionstandby**command to restrict the debug output for the **debugstandby** command. If the **debugconditionstandby**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 **debugconditionstandby** command after entering the **debugstandby**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
debug condition interface	Limits output for some debugging commands based on the interfaces.
debug standby	Displays HSRP state changes.
debug standby errors	Displays error messages related to HSRP.
debug standby events	Displays events related to HSRP.
debug standby events icmp	Displays debugging messages for the HSRP ICMP redirects filter.
debug standby packets	Displays debugging information for packets related to HSRP.

debug condition voice-port

To display debug output for a specified port, use the **debugconditionvoice-port** command in privileged EXEC mode. To enable debugging messages for all voice ports, use the **no** form of this command.

debug condition voice-port *port-number*

no debug condition voice-port *port-number*

Syntax Description

port-number	Voice port for which you want to display all enabled debugging messages. **Note** Syntax for the *port-number*argument is platform-dependent; type ? to determine available options for argument syntax.

Command Default

Debugging messages are enabled for all voice ports.

Command Modes

Privileged EXEC(#)

Command History

Release	Modification
12.4(6)T	This command was introduced.

Usage Guidelines

This command filters out debugging output for all voice ports except the port specified in the command. When this command is configured, the gateway displays debugging messages only for the specified port.

To display debugging output, you must first enable the **debugvoipapplicationstcappall**command. If no **debug** commands are turned on, no debugging messages are displayed even if the **debugconditionvoice-port** command is enabled.

The **debugconditionvoice-port** command can be configured multiple times to display output for more than one voice port. This command differs from the **debugvoipapplicationstcappport**command, which can be configured to display output for only one voice port.

To display which debug conditions have been set, use the **showdebug** command.

Before disabling conditions, first disable any debugging commands; otherwise output for all ports could flood the logging buffer.

Examples

The following example filters debugging output so that output only for ports 2/1 and 2/3 is displayed:

```
Router# debug condition voice-port 2/1
Condition 1 set
*Mar  1 22:24:15.102:  Debug: Condition 1, voice-port 2/1 triggered, count 1
```

```
Router# debug condition voice-port 2/3
Condition 2 set
*Mar  1 22:24:24.794:  Debug: Condition 2, voice-port 2/3 triggered, count 2
Router# show debug
Condition 1: voice-port 2/1 (1 flags triggered)
        Flags: voice-port condition
Condition 2: voice-port 2/3 (1 flags triggered)
        Flags: voice-port condition
```

Related Commands

Command	Description
debug sccp all	Displays debugging information for SCCP.
debug voip application stcapp all	Displays debugging information for the components of the STCAPP.
debug voip application stcapp port	Enables STCAPP debugging for a specific port.
show debug	Displays the types of debugging and the debugging conditions that are enabled on your router.

debug condition vrf

To limit debug output to a specific Virtual Routing and Forwarding (VRF) instance, use the **debugconditionvrf**command in privileged EXEC mode. To remove the debug condition, use the **undebug** version of the command .

debug condition vrf *vrf-name*

undebug condition vrf *vrf-name*

Syntax Description

vrf-name	Name of a VRF.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.2S	This command was introduced.

Usage Guidelines

Use this command to limit debug output to a single VRF.

Note EIGRP does not support the **debugconditionvrf** command.

Examples

The following example shows how to limit debugging output to VRF red:

```
Router# debug condition vrf red
```

Related Commands

Command	Description
vrf definition	Defines a virtual routing and forwarding instance.

debug condition xconnect

To conditionally filter debug messages related to xconnect configurations, use the **debugconditionxconnect** command in privileged EXEC configuration mode. To disable the filtering of xconnect debug messages, use the **no** form of this command.

debug condition xconnect {**fib** *type*| **interface** *type number* [*dlci*| **vp** *number*| **vc** *number*]| **peer** *ip-address* **vcid** *vcid*| **segment** *segment-id*}

no debug condition xconnect {**fib** *type*| **interface** *type number* [*dlci*| **vp** *number*| **vc** *number*]| **peer** *ip-address* **vcid** *vcid*| **segment** *segment-id*}

Syntax Description

fib *type*	Filters control-plane and data-plane debug messages for the xconnect segment pair specified by matching against the Forwarding Information Base (FIB) Interface Descriptor Block (IDB) information associated with a particular interface on a line card.
interface *type number*	Filters control-plane and data-plane debug messages for the xconnect segment pair specified by the interface type and number on a Route Processor.
dlci	(Optional) The Frame Relay data-link connection identifier (DLCI) for the xconnect segment pair associated with a Frame Relay segment.
vp *number*	(Optional) The ATM virtual path (VP) number for the xconnect segment pair associated with an ATM segment.
vc *number*	(Optional) The ATM virtual circuit (VC) number for the xconnect segment pair associated with an ATM segment.
peer	Filters control-plane and data-plane debug messages for the xconnect segment pair specified by the remote peer IP address and the pseudowire virtual circuit ID (VCID).
ip-address	The IP address of the remote peer router.
vcid *vcid*	The VCID of the xconnect pseudowire.
segment	Filters data-plane debug messages for the xconnect segment pair specified by a segment ID.
segment-id	The segment ID. The segment ID value can be found in the output of the **showssmid** command.

Command Default Debug messages are not filtered.

Command Modes Privileged EXEC

Command History

Release	Modification
12.2(28)SB	This command was introduced.
15.3(1)S	This command was integrated in Cisco IOS Release 15.3(1)S.

Usage Guidelines Use the **debugconditionxconnect** command to specify conditions for filtering the debug messages displayed by related subscriber service switch (SSS), xconnect, and attachment circuit **debug** commands.

Examples The following example sets filter conditions that allow related **debug** commands to display debug messages for only the xconnect segment pair specified by the remote peer IP address and the pseudowire VCID:

```
debug condition xconnect peer 10.0.0.1 vcid 100
```

The following example sets filter conditions that allow related **debug** commands to display debug messages for only the xconnect segment pair specified by the serial interface number and DLCI:

```
debug condition xconnect interface serial 0/0 100
```

The following example sets filter conditions that allow related **debug** commands to display debug messages for only the xconnect segment pair specified by the port mode ATM interface number:

```
debug condition xconnect interface atm 0/0
```

The following example sets filter conditions that allow related **debug** commands to display debug messages for only the xconnect segment pair specified by the VP mode ATM interface number:

```
debug condition xconnect interface atm 0/0 vp 1
```

The following example sets filter conditions that allow related **debug** commands to display debug messages for only the xconnect segment pair specified by the VC mode ATM interface number:

```
debug condition xconnect interface atm 0/0 vc 1/40
```

The following example finds the segment ID associated with an L2TPv3 xconnect segment pair and sets filter conditions that allow related **debug** commands to display debug messages for only that xconnect segment pair:

```
Router# show ssm id
!
Segment-ID: 8193 Type: L2TPv3[8]
!
Router# debug conditional xconnect segment 8193
```

Related Commands

Command	Description
debug acircuit	Displays errors and events that occur on the attachment circuits.
debug sss aaa authorization event	Displays messages about AAA authorization events that are part of normal call establishment.
debug sss aaa authorization fsm	Displays information about AAA authorization state changes.
debug sss error	Displays diagnostic information about errors that may occur during SSS call setup.
debug sss event	Displays diagnostic information about SSS call setup events.
debug sss fsm	Displays diagnostic information about the SSS call setup state.
debug xconnect	Displays debug messages related to an xconnect configuration.
show ssm id	Displays SSM information for switched Layer 2 segments.

debug configuration lock

To enable debugging of the Cisco IOS configuration lock, use the **debugconfigurationlock**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug configuration lock

no debug command lock

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.
12.0(31)S	This command was integrated into Cisco IOS Release 12.0(31)S.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.
12.2(33)SB	This command was integrated into Cisco IOS Release 12.2(33)SB.

Examples

The following is sample output with **debugconfigurationlock** enabled (assuming that the feature is enabled using the **configurationmodeexclusivemanual** command in global configuration mode):

```
Router# debug configuration lock
Session1 from console
=========================
Router# configure terminal lock
Configuration mode locked exclusively. The lock will be cleared once you exit out of
configuration mode using end/exit
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
Parser : LOCK REQUEST in EXCLUSIVE mode
Parser: <configure terminal lock> - Config. Lock requested by process <3> client <PARSER
Client>
Parser: <configure terminal lock> - Config. Lock acquired successfully !
Router(config)#
Session2 VTY (User from session2 is trying to enter single user config (exclusive) config
mode)
================================
Router# c
onfigure terminal lock

Configuration mode locked exclusively by user 'unknown' process '3' from terminal '0'.
Please try later.
```

```
Router#
Session1 from console
======================
Router(config)#
Parser : LOCK REQUEST in EXCLUSIVE mode
Parser: <configure terminal lock> - Config. Lock requested by process <104> client <PARSER
 Client>
Parser: <configure terminal lock> - NON_BLOCKING : Config mode locked <EXCLUSIVE> owner <3>
Router(config)#
Router(config)# end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Parser: <Configure terminal> - Config. EXC UnLock requested by user<3>
Parser: <Configure terminal> - Config UNLOCKED !
Router#
```

Related Commands

Command	Description
configuration mode exclusive	Enables single-user (exclusive) access functionality for the Cisco IOS CLI.
show configuration lock	Displays information about the lock status of the running configuration file during a configuration replace operation.

debug confmodem

To display information associated with the discovery and configuration of the modem attached to the router, use the **debugconfmodem**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 **debugconfmodem** command is used in debugging configurations that use the **modemautoconfig** command.

Examples

The following is sample output from the**debugconfmodem**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 **debugconn**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.

Command Default

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 content-scan

To enable content scan debugging, use the **debug content-scan** command in privileged EXEC mode. To disable content scan debugging, use the **no** form of this command.

debug content-scan {**access-list** {*access-list-number* | *extended-access-list-number* | *access-list-name*}| **control-plane| errors | events | function-trace | packet-path**}

no debug content-scan {**access-list** {*access-list-number* | *extended-access-list-number* | *access-list-name*}| **control-plane | errors | events | function-trace | packet-path**}

Syntax Description

access-list	Enables debugging of content scan access control lists (ACLs).
access-list-number	Access list number. The range is from 1 to 199.
extended-access-list-number	Extended access list number. The range is from 1300 to 2699.
access-list-number	Access list name.
control-plane	Enables debugging of content scan control plane messages.
errors	Enables debugging of content scan errors.
events	Enables debugging of content scan events.
function-trace	Enables debugging of content scan function trace.
packet-path	Enables debugging of content scan packet path.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.2(1)T1	This command was introduced.
15.2(4)M	This command was modified. The **access-list** and **control-plane** keywords and the *access-list-number*, *extended-access-list-number*, and *access-list-name* arguments were added.

Usage Guidelines

The content scanning process redirects client web traffic to the ScanSafe web server.

You must configure ACLs by using the **ip access-list extended** command before you enable the debugging of content-scan ACLs. The **debug content-scan access-list** command enables the conditional debugging for

content scan. The conditional debugging does not work for existing sessions when you enable content scan; the ACL match occurs after the first packet is received.

Examples

The following example shows how to enable extended ACL 149 and the debugging of content scan ACL 149:

```
Device(config)# ip access-list extended 149
Device(config-ext-nacl)# permit ip host 10.1.0.1
Device(config-ext-nacl)# end
Device# debug content-scan access-list 149
 Content-scan ACL based conditional debugging is on
```

The following is sample output from the **debug content-scan access-list 149** command:

```
Device# debug content-scan access-list 149

Content-scan ACL based conditional debugging is on

Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Checking if it's a web trafficSrc IP:10.1.0.1(3646),
 Dst IP: 198.51.100.195(80),index:4
Feb 19 19:01:02.887:
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH: Populating user/usergroup infoSrc IP:10.1.0.1(3646),
 Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-EVE:Username Received: defpmuser Src IP:10.1.0.1(E3E) Dst
IP: 198.51.100.195(50) Pak:2033BCA0
Feb 19 19:01:02.887: CONT-SCAN-EVE:Usergroup from pmap Src IP:10.1.0.1(E3E) Dst IP:
198.51.100.195(50) Pak:2033BCA0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Protocol not configuredSrc IP:10.1.0.1(3646), Dst
IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Adding session into HashSrc IP:10.1.0.1(3646), Dst
 IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Free port equals source port for src_ip 10.1.0.1(3646)
 dst_ip 198.51.100.195(80) 2033BCA0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:session enqueued 21ED1DE0 for src_ip 10.1.0.1(3646)
 dst_ip 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Inserting Ingress session 21ED1DE0 in hash table
at 1646 Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:Inserting Egress session 21ED1DE0 in hash table at
 1629 Src IP:10.1.0.1(3646), Dst IP: 198.51.100.195(80)
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:timer wheel started for 21ED1DE0 for src_ip
10.1.0.1(3646)
Feb 19 19:01:02.887: CONT-SCAN-EVE:1 Usergroups from auth proxy Src IP:10.1.0.1(E3E), Dst
IP: 198.51.100.195(50), cs_entry:21ED1DE0
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:conx 21EE2024 ingress_fd 1073741847
Feb 19 19:01:02.887: CONT-SCAN-PAK-PATH:CONT SCAN: received fd1: 1073741847 and tcp flags
= 0x2, payload_len = 0
Feb 19 19:01:02.887: CONT-SCAN-EVE:1 Usergroups found Src IP:10.1.0.1(E3E), Dst IP:
198.51.100.195(50), cs_entry:21ED1DE0
!
!
!
```

Related Commands

Command	Description
content-scan out	Enables content scanning on an egress interface.
content-scan whitelisting	Enables whitelisting of incoming traffic and enters content-scan whitelisting configuration mode.
show content-scan	Displays content scan information.

debug control-plane

To display debugging output from the control-plane routines, use the **debugcontrolplane** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug control-plane [**all**| **host**| **port-filtering**| **queue-thresholding**| **log**]

no debug control-plane [**all**| **host**| **port-filtering**| **queue-thresholding**| **log**]

Syntax Description

all	(Optional) Displays all events on all control-plane interfaces.
host	(Optional) Displays all events on the control-plane host interface.
port-filtering	(Optional) Displays TCP/IP protocol port-filtering events.
queue-thresholding	(Optional) Displays TCP/IP protocol queue-thresholding events.
log	(Optional) Displays control-plane logging events.

Command Default

The default is debugging off.

Command Modes

Privileged EXEC

Command History

Release	Moditication
12.4(4)T	This command was introduced.
12.4(6)T	The **log** keyword was added to support control plane logging.

Usage Guidelines

Use the **debugcontrolplane** command if you want to display the debugging output from the control-plane routines.

Examples

The following example show a display from the **debugcontrol-plane** command:

```
Router# debug control-plane
Control-plane infrastructure events debugging is on
Router# cp_receive_classify - marking pak host
 ingress pak marked cef-exception
```

The following example shows a display from the **debugcontrol-plane** command using the port-filtering option:

```
Router# debug control-plane port-filtering

TCP/IP Port filtering events debugging is on
Dropped UDP dport 1243 sport 62134 saddr 209.165.200.225
```

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

Table 3: debug control plane field descriptions

Field	Description
dport	UPD destination port.
sport	UPD source port.
saddr	Source address of the IP packets.

Related Commands

Command	Description
clear control-plane	Clears packet counters for control-plane interfaces and subinterfaces.
control-plane	Enters control-plane configuration mode, which allows users to associate or modify attributes or parameters that are associated with the control plane of the device.
show control-plane cef-exception counters	Displays the control plane packet counters for the control plane CEF-exception subinterface.
show control-plane cef-exception features	Displays the configured features for the control plane CEF-exception subinterface.
show control-plane counters	Displays the control-plane packet counters for the aggregate control-plane interface.
show control-plane features	Displays the configured features for the aggregate control-plane interface.
show control-plane host counters	Displays the control plane packet counters for the control plane host subinterface.
show control-plane host features	Displays the configured features for the control plane host subinterface.
show control-plane host open-ports	Displays a list of open TCP/UDP ports that are registered with the port-filter database.

show control-plane transit counters	Displays the control plane packet counters for the control plane transit subinterface.

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 **debugcops** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cops [detail]

no debug cops [detail]

Syntax Description

detail	(Optional) Displays additional debug information, including the contents of COPS and Resource Reservation Protocol (RSVP) messages.

Command Default

COPS process debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

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
```

Examples

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)#
```

Examples 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)#
```

Examples 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)#
```

Examples 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)#
```

Examples 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)#
```

Examples 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 **debugcops** command is used along with the **debugiprsvppolicy** command, refer to the second example of the **debugiprsvppolicy** command.

Related Commands

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

debug cot

To display information about the Continuity Test (COT) functionality, use the **debugcot**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

api	Displays information about the COT application programming interface (API).
dsp	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.
queue	Display information related to the COT internal queue.
detail	Display information about COT internal detail; summary of the **debugcotapi**, **debugcotdsp**, and **debugcotqueue** commands.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(7)	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following is sample output from the **debugcotapi** 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 0xFFF1 duration 60000
```

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

Table 4: debug cot api Field Descriptions

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

The following is sample output from the **debugcotdsp** 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
```

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

Table 5: 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 **debugcotqueue** 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
```

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

Table 6: 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 **debugcotdetail** 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 0xFFF1 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 0xFFF1, 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 **debugcotdetail**command is a summary of the **debugcotapi**, **debugcotdsp**, and **debugcotqueue**commands, the field descriptions are the same.

debug cpp event

Note Effective with Release 12.3(4)T, the **debugcppevent** command is no longer available in Cisco IOS 12.3(T) releases.

To display general Combinet Proprietary Protocol (CPP) events, use the **debugcppevent**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 is no longer available in Cisco IOS 12.(3)T releases.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

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

The **debugcppevent**command displays events such as CPP sequencing, group creation, and keepalives.

Examples

One or more of the messages in the table below appear when you use the**debugcppevent** 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 7: 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.

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

Related Commands

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

debug cpp negotiation

Note Effective with Release 12.3(4)T, the **debugcppnegotiation** command is no longer available in Cisco IOS 12.3T releases.

To display Combinet Proprietary Protocol (CPP) negotiation events, use the **debugcppnegotiation**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 is no longer available in Cisco IOS 12.3T releases.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

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

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

Examples

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

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

The table below describes the significant fields in the display.

Table 8: debug CPP negotiation Field Descriptions

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

Related Commands

Command	Description
debug cot	Displays information about the COT functionality.
debug cpp packet	Displays CPP packets.

debug cpp packet

Note Effective with Release 12.3(4)T, the **debugcpppacket**command is no longer available in Cisco IOS 12.3T Releases.

To display Combinet Proprietary Protocol (CPP) packets, use the **debugcpppacket**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 is no longer available in Cisco IOS 12.3T releases.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

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

The **debugcpppacket**command displays the hexadecimal values of the packets.

Examples

The following is sample output from the **debugcpppacket** 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
debug cot	Displays information about the COT functionality.
debug cpp negotiation	Displays CPP negotiation events.

debug credentials

To set debugging on the credentials service that runs between the Cisco Unified Survivable Site Remote Telephony (Cisco Unified SRST) router and Cisco CallManager, use the **debugcredentials** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug credentials

no debug credentials

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Cisco IOS Release	Modification
12.3(14)T	This command was introduced for Cisco SRST 3.3.

Usage Guidelines

Use this command to monitor Cisco CallManager while it requests certificates from the Cisco Unified SRST router.

Examples

The following is sample output showing the credentials service that runs between the Cisco Unified SRST router and Cisco CallManager. The credentials service provides Cisco CallManager with the certificate from the Cisco Unified SRST router.

```
Router# debug credentials
Credentials server debugging is enabled
Router#
May 25 12:08:17.944: Credentials service: Start TLS Handshake 1 10.5.43.174 4374
May 25 12:08:17.948: Credentials service: TLS Handshake returns OPSSLReadWouldBlockErr
May 25 12:08:18.948: Credentials service: TLS Handshake returns OPSSLReadWouldBlockErr
May 25 12:08:19.948: Credentials service: TLS Handshake returns OPSSLReadWouldBlockErr
May 25 12:08:20.964: Credentials service: TLS Handshake completes
```

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

Table 9: debug credentials Field Descriptions

Field	Description
Start TLS Handshake 1 10.5.43.174 4374	Indicates the beginning of the TLS handshake between the secure SRST router and Cisco CallManager. In this example, 1 indicates the socket, 10.5.43.174 is the IP address, and 4374 is the port of Cisco CallManager.
TLS Handshake returns OPSSLReadWouldBlockErr	Indicates that the handshake is in process.

Field	Description
TLS Handshake completes	Indicates that the TLS handshake has finished and that the Cisco CallManager has received the secure SRST device certificate.

Related Commands

Command	Description
credentials (SRST)	Provides the Cisco Unified SRST router certificate to Cisco CallManager and enters credentials configuration mode.
ip source-address (credentials)	Enables the Cisco Unified SRST router to receive messages from Cisco CallManager through the specified IP address and port.
show credentials	Displays the credentials settings on the Cisco Unified SRST router that are supplied to Cisco CallManager for use during secure SRST fallback.
show debugging	Displays information about the types of debugging that are enabled for your router.
trustpoint (credentials)	Specifies the name of the trustpoint that is to be associated with the Cisco Unified SRST router certificate.

debug crm

To troubleshoot the Carrier Resource Manager (CRM), use the **debugcrm** 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

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

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was introduced.

Release	Modification
12.3(8)T	The **all**, **default**, **detail**, **error**, **call**, **informational**, **software**, **function**, and **inout**keywords were added.

Usage Guidelines

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

Examples

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

```
Router# debug crm all

01:21:23: //-1/xxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:21:23: //-1/xxxxxxxxxxxx/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/xxxxxxxxxxxx/CRM/crm_call_update:
    Increment the call count
    CarrierID=2023, TrunkGroupLabel=2023
    Update is for TrunkGroupLabel, Mask=0x00000001
01:21:23: //-1/xxxxxxxxxxxx/CRM/crm_call_update:
    IncomingVoiceCalls=1
Router#
01:21:48: //-1/xxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:21:48: //-1/xxxxxxxxxxxx/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/xxxxxxxxxxxx/CRM/crm_send_periodic_update:
01:22:13: //-1/xxxxxxxxxxxx/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/xxxxxxxxxxxx/CRM/crm_call_update:
    Decrement the call count
    CarrierID=2023, TrunkGroupLabel=2023
    Update is for TrunkGroupLabel, Mask=0x00000001
01:22:18: //-1/xxxxxxxxxxxx/CRM/crm_call_update:
    IncomingVoiceCalls=0
```

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

Table 10: debug crm all Field Descriptions

Field	Description
//-1/xxxxxxxxxxxx/CRM/print_event:	The format of this message is //callid/GUID/CRM/function name: • CallEntry ID is -1. This indicates that the CallEntry ID is unavailable. • GUID is xxxxxxxxxxxxxxxx. This indicates that the GUID is unavailable. • CRM is the module name. • Theprint_event:field shows that the CRM is showing the print event function.
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

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

debug crypto ace b2b

To enable IPSec VPN SPA debugging for a Blade Failure Group, use the debug crypto ace b2b command in privileged EXEC mode.

debug crypto ace b2b

Command Default

No default behavior or values.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(18)SXE2	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Examples

The following example enables IPSec VPN SPA debugging for a Blade Failure Group:

```
Router# debug crypto ace b2b
ACE B2B Failover debugging is on
```

Related Commands

Command	Description
linecard-group feature card	Assigns a group ID to a Blade Failure Group.
show crypto ace redundancy	Displays information about a Blade Failure Group.
show redundancy linecard-group	Displays the components of a Blade Failure Group.

debug crypto ace boot

To enable ACE boot API debugging, use the debug crypto ace boot command in privileged EXEC mode. To disable ACE boot API debugging, use the **no** form of this command.

debug crypto ace boot

no debug crypto ace boot

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE boot API debugging:

```
Router# debug crypto ace boot
ACE Boot debugging is on
```

debug crypto ace cfgmon

To enable ACE CFGMON transactions debugging, use the debug crypto ace cfgmon command in privileged EXEC mode. To disable ACE CFGMON transactions debugging, use the **no** form of this command.

debug crypto ace cfgmon

no debug crypto ace cfgmon

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE CFGMON transaction debugging:

```
Router# debug crypto ace cfgmon
ACE CFGMON Transactions debugging is on
```

debug crypto ace congestion-mgr

To enable ACE Crypto engine congestion manager debugging, use the debug crypto ace congestion-mgr command in privileged EXEC mode. To disable ACE Crypto engine congestion manager debugging, use the **no** form of this command.

debug crypto ace congestion-mgr

no debug crypto ace congestion-mgr

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE Crypto engine congestion manager debugging:

```
Router# debug crypto ace congestion-mgr
ACE Crypto Engine Congestion Manager debugging is on
```

debug crypto ace dpd

To enable ACE DPD debugging, use the debug crypto ace dpd command in privileged EXEC mode. To disable ACE DPD debugging, use the **no** form of this command.

debug crypto ace dpd

no debug crypto ace dpd

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE DPD debugging:

```
Router# debug crypto ace dpd
ACE-DPD debugging is on
```

debug crypto ace error

To enable error logging debugging, use the debug crypto ace error command in privileged EXEC mode. To disable ACE error logging debugging, use the **no** form of this command.

debug crypto ace error

no debug crypto ace error

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables error log debugging:

```
Router# debug crypto ace error
ACE Error logging debugging is on
```

debug crypto ace hapi

To enable ACE HAPI transactions debugging, use the debug crypto ace hapi command in privileged EXEC mode. To disable ACE HAPI transactions debugging, use the **no** form of this command.

debug crypto ace hapi

no debug crypto ace hapi

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE HAPI transaction debugging:

```
Router# debug crypto ace hapi
ACE HAPI transactions debugging is on
```

debug crypto ace ikea

To enable ACE IKEA transactions debugging, use the debug crypto ace ikea command in privileged EXEC mode. To disable ACE IKEA transactions debugging, use the **no** form of this command.

debug crypto ace ikea

no debug crypto ace ikea

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE IKEA transaction debugging:

```
Router# debug crypto ace ikea
ACE IKEA transactions debugging is on
```

debug crypto ace invspi

To enable ACE invalid SPI debugging, use the debug crypto ace invspi command in privileged EXEC mode. To disable ACE invalid SPI debugging, use the **no** form of this command.

debug crypto ace invspi

no debug crypto ace invspi

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE invalid SPI debugging:

```
Router# debug crypto ace invspi
ACE Invalid SPI debugging is on
```

debug crypto ace ipd

To enable ACE IPD debugging, use the debug crypto ace ipd command in privileged EXEC mode. To disable ACE IPD debugging, use the **no** form of this command.

debug crypto ace ipd

no debug crypto ace ipd

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE IPD debugging:

```
Router# debug crypto ace ipd
ACE IPD debugging is on
```

debug crypto ace mace

To enable multi-ACE messages debugging, use the debug crypto ace mace command in privileged EXEC mode. To disable multi-ACE messages debugging, use the **no** form of this command.

debug crypto ace mace

no debug crypto ace mace

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables multi-ACE debugging:

```
Router# debug crypto ace mace
MULTI-ACE messages debugging is on
```

debug crypto ace pcp

To enable ACE PCP transactions debugging, use the debug crypto ace pcp command in privileged EXEC mode. To disable ACE PCP transaction debugging, use the **no** form of this command.

debug crypto ace pcp

no debug crypto ace pcp

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE PCP transactions debugging:

```
Router# debug crypto ace pcp
ACE PCP Transactions debugging is on
```

debug crypto ace polo

To enable ACE policy loader debugging, use the debug crypto ace polo command in privileged EXEC mode. To disable ACE policy loader debugging, use the **no** form of this command.

debug crypto ace polo

no debug crypto ace polo

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE policy loader debugging:

```
Router# debug crypto ace polo
ACE Policy Loader debugging is on
```

debug crypto ace propcfg

To enable ACE propagate configuration debugging, use the debug crypto ace propcfg command in privileged EXEC mode. To disable ACE propagate configuration debugging, use the **no** form of this command.

debug crypto ace propcfg

no debug crypto ace propcfg

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE propagate configuration debugging:

```
Router# debug crypto ace propcfg
ACE Propagate Config debugging is on
```

debug crypto ace qos

To enable ACE QoS debugging, use the debug crypto ace qos command in privileged EXEC mode. To disable ACE QoS debugging, use the **no** form of this command.

debug crypto ace qos

no debug crypto ace qos

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE QoS debugging:

```
Router# debug crypto ace qos
ACE QoS debugging is on
```

debug crypto ace rcon

To enable ACE RCON messages debugging, use the debug crypto ace rcon command in privileged EXEC mode. To disable ACE RCON messages debugging, use the **no** form of this command.

debug crypto ace rcon

no debug crypto ace rcon

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE RCON messages debugging:

```
Router# debug crypto ace rcon
ACE RCON messages debugging is on
```

debug crypto ace redundancy

To enable ACE HA debugging, use the debug crypto ace redundancy command in privileged EXEC mode. To disable ACE HA debugging, use the **no** form of this command.

debug crypto ace redundancy

no debug crypto ace redundancy

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE HA debugging:

```
Router# debug crypto ace redundancy
ACE HA debugging is on
```

debug crypto ace spi

To enable ACE SPI debugging, use the debug crypto ace spi command in privileged EXEC mode. To disable ACE SPI debugging, use the **no** form of this command.

debug crypto ace spi

no debug crypto ace spi

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE SPI debugging:

```
Router# debug crypto ace spi
ACE SPI debugging is on
```

debug crypto ace stats

To enable ACE stats module debugging, use the debug crypto ace stats command in privileged EXEC mode. To disable ACE stats module debugging, use the **no** form of this command.

debug crypto ace stats

no debug crypto ace stats

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE stats module debugging:

```
Router# debug crypto ace stats
ACE Stats Module debugging is on
```

debug crypto ace syslog

To enable ACE syslog messages debugging, use the debug crypto ace syslog command in privileged EXEC mode. To disable ACE syslog messages debugging, use the **no** form of this command.

debug crypto ace syslog

no debug crypto ace syslog

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE syslog messages debugging:

```
Router# debug crypto ace syslog
ACE Syslog messages debugging is on
```

debug crypto ace tftp

To enable ACE TFTP boot debugging, use the debug crypto ace tftp command in privileged EXEC mode. To disable ACE TFTP boot debugging, use the **no** form of this command.

debug crypto ace tftp

no debug crypto ace tftp

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE TFTP boot debugging:

```
Router# debug crypto ace tftp
ACE TFTP Boot debugging is on
```

debug crypto ace topn

To enable ACE-TOPN-HELPER debugging, use the debug crypto ace topn command in privileged EXEC mode. To disable ACE-TOPN-HELPER debugging, use the **no** form of this command.

debug crypto ace topn

no debug crypto ace topn

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE-TOPN-HELPER debugging:

```
Router# debug crypto ace topn
ACE-TOPN-HELPER debugging is on
```

debug crypto ace warning

To enable ACE warning log debugging, use the debug crypto ace warning command in privileged EXEC mode. To disable ACE warning logging debugging, use the **no** form of this command.

debug crypto ace warning

no debug crypto ace warning

Syntax Description

This command has no arguments or keywords.

Command Default

None

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(33)SXI5	This command was introduced.

Examples

The following example enables ACE warning log debugging:

```
Router# debug crypto ace warning
ACE Warning logging debugging is on
```

debug crypto condition unmatched

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

debug crypto condition unmatched [**isakmp| ipsec| ikev2| engine| gdoi-group**]

no debug crypto condition unmatched [**isakmp| ipsec| ikev2| engine| gdoi-group**]

Syntax Description

isakmp | **ipsec** | **ikev2** | **engine** | **gdoi-group**	(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.

Command Default

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.
12.2(18)SXD	This command was integrated into Cisco IOS Release 12.2(18)SXD.
15.1(3)T	This command was modified. The **gdoi-group** keyword was replaced by the **group**keyword of the**debugcryptogdoicondition**command.
Cisco IOS XE Release 3.8S	This command was integrated into Cisco IOS XE Release 3.8S.

Usage Guidelines

After the **debugcryptocondition** command has been enabled, you can use the **debugcryptoconditionunmatched** 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 **debugcryptoconditionunmatched** 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
debug crypto gdoi condition	Defines conditional debug filters for GDOI.
debug crypto condition	Defines conditional debug filters.
show crypto debug-condition	Displays crypto debug conditions that have already been enabled in the router.
show crypto ipsec sa	Displays the settings used by current SAs.
show crypto isakmp sa	Displays all current IKE SAs at a peer.

debug crypto ctcp

To display information about a Cisco Tunnel Control Protocol (cTCP) session, use the **debugcryptoctcp**command in privileged EXEC mode. To turn off debugging, use the **no** form of this command.

debug crypto ctcp

no debug crypto ctcp

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is turned off.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(9)T	This command was introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

You can use this command if a cTCP session fails to come up.

Examples

The following example shows that debugging has been turned on for a cTCP session:

```
Router# debug crypto ctcp
```

Related Commands

Command	Description
crypto ctcp	Configures cTCP encapsulation for Easy VPN.

debug crypto engine

To display debugging messages about crypto engines, which perform encryption and decryption, use the **debugcryptoengine** 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.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

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 **debugcryptoengine** command. The first sample output shows messages from a router that successfully generates Rivest, Shamir, and Adelma (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 **debugcryptoengineacceleratorlogs**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.

Command Default

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.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

Use the **debugcryptoengineacceleratorlogs**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 **debugcryptoengineacceleratorlogs** 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
crypto engine accelerator	Enables or disables the crypto engine accelerator if it exists.
show crypto engine accelerator logs	Prints information about the last 32 CGX Library packet processing commands, and associated parameters sent from the VPN module driver to the VPN module hardware.

Command	Description
show crypto engine accelerator sa-database	Prints active (in-use) entries in the platform-specific VPN module database.
show crypto engine configuration	Displays the Cisco IOS crypto engine of your router.

debug crypto engine ism-vpn

To enable debugging for a Cisco VPN Internal Service Module (ISM), use the **debug crypto engine ism-vpn** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug crypto engine ism-vpn[**init**| **interrupt**| **polo**[**detail**| **dump**]| **shim**[**detail**]| **tftp**]

no debug crypto engine ism-vpn[**init**| **interrupt**| **polo**[**detail**| **dump**]| **shim**[**detail**]| **tftp**]

Syntax Description

init	(Optional) Enables initial state (INIT state) debugging for an Application Control Engine (ACE) appliance in Cisco VPN ISM.
interrupt	(Optional) Enables interrupt service debugging.
polo	(Optional) Enables VPN ISM polo debugging.
detail	(Optional) Enables detailed debugging.
dump	(Optional) Enables packet dump polo debugging.
shim	(Optional) Enables VPN ISM shim layer debugging.
tftp	(Optional) Enables debugging for TFTP download or upload on the Cisco VPN ISM.

Command Default

Cisco VPN ISM debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(2)T	This command was introduced.

Examples

The following example shows how to enable initial state (INIT state) debugging.

```
Device# debug crypto engine ism-vpn init
INIT debugging is on
```

Related Commands

Command	Description
debug crypto engine ism-vpn ssl	Enables debugging for Cisco VPN ISM SSL.

Command	Description
debug crypto engine ism-vpn traffic	Enables debugging for Cisco VPN ISM traffic.
debug crypto engine ism-vpn traffic selector	Enables debugging for specific traffic in a Cisco VPN ISM.
show crypto engine accelerator statistic	Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or the Cisco VPN ISM.

debug crypto engine ism-vpn ssl

To enable debugging for Secure Sockets Layer (SSL) in a Cisco VPN Internal Service Module (VPN ISM), use the **debug crypto engine ism-vpn ssl** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug crypto engine ism-vpn ssl{**context** *rule-number session-id*| **direction** *number*}

no debug crypto engine ism-vpn ssl{**context** *rule-number session-id*| **direction** *number*}

Syntax Description

context *rule-number session-id*	Enables debugging for a specific SSL depending on the rule or the session. The range for the rule number is from 1 to 10. The range for the session ID is 0 to 4294967294.
direction *number*	Enables debugging for packets that are in the inbound or outbound direction. The range for the direction number is from 0 to 3, where 0 indicates the inbound and outbound direction, 1 indicates encryption , 2 indicates decryption, and 3 indicates exceptions.

Command Default

Cisco VPN ISM SSL debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(2)T	This command was introduced.

Examples

The following example shows how to enable debugging of rule 3 and context 1:

```
Device# debug crypto engine ism-vpn ssl context 3 1
ISM-VPN Enable rule 3, ctx: 1

ISM-VPN SSL Context debugging is on
```

Related Commands

Command	Description
debug crypto engine ism-vpn	Enables debugging for a Cisco VPN ISM.
debug crypto engine ism-vpn traffic	Enables debugging for Cisco VPN ISM traffic.
debug crypto engine ism-vpn traffic selector	Enables debugging for specific traffic in a Cisco VPN ISM.

Command	Description
show crypto engine accelerator statistic	Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or Cisco VPN ISM.

debug crypto engine ism-vpn traffic

To enable debugging for Cisco VPN Internal Service Module (ISM) traffic, use the **debug crypto engine ism-vpn traffic** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug crypto engine ism-vpn traffic {**all**| **detail**| **exception** *number*| **inbound**| **outbound**| **selective**| **vam**}

no debug crypto engine ism-vpn traffic {**all**| **detail**| **exception** *number*| **inbound**| **outbound**| **selective**| **vam**}

Syntax Description

all	Enables debugging for all traffic that flows to the Cisco VPN ISM.
detail	Enables detailed debugging for all traffic that flows to the Cisco VPN ISM.
exception *number*	Enables debugging for specified exceptions in the traffic. The range is from 0 to 54. 0 enables debugging for all exceptions.
inbound	Enables debugging for inbound traffic.
outbound	Enables debugging for outbound traffic.
selective	Enables selective rules for the traffic on the Cisco VPN ISM.
vam	Enables debugging for the Cisco VPN Acceleration Module (VAM) in the Cisco VPN ISM.

Command Default

Cisco VPN ISM traffic debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(2)T	This command was introduced.

Examples

The following shows how to enable debugging for all exceptions in the traffic:

```
Device# debug crypto engine ism-vpn traffic exception 0
ISM-VPN Traffic Detail debugging is on
```

Related Commands

Command	Description
debug crypto engine ism-vpn	Enables debugging for a Cisco VPN ISM.

Command	Description
debug crypto engine ism-vpn traffic selector	Enables debugging for specific traffic in a Cisco VPN ISM.
debug crypto engine ism-vpn ssl	Enables debugging for Cisco VPN ISM SSL.
show crypto engine accelerator statistic	Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or the Cisco VPN ISM.

debug crypto engine ism-vpn traffic selector

To enable debug rules for specific traffic in Cisco VPN Internal Service Module (ISM), use the **debug crypto engine ism-vpn traffic selector** command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug crypto engine ism-vpn traffic selector {**ipv4| ipv6**} {**disable** *rule-number*| **enable** *rule-number source-ip-address source-address-mask destination-ip-address destination-address-mask protocols*}

no debug crypto engine ism-vpn traffic selector {**ipv4| ipv6**} {**disable** *rule-number*| **enable** *rule-number source-ip-address source-address-mask destination-ip-address destination-address-mask protocols*}

Syntax Description

ipv4	Enables debugging rules for IPv4 addresses.
ipv6	Enables debugging rules for IPv6 addresses.
disable	Enables debugging rules are disabled.
enable	Enables debugging rules are enabled.
rule-number	A specific rule. The range is from 1 to 10.
source-ip-address source-address-mask	Source IP address and network mask of the traffic for which rules must be enabled or disabled.
destination-ip-address destination-address-mask	Destination IP address and network mask of the traffic for which rules must be enabled or disabled.
protocols	Protocols in the source and destination IP addresses of the traffic for which rules must be enabled or disabled. The range is from 0 to 155. 0 denotes all protocols.

Command Default

Cisco VPN ISM selective traffic debugging is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.3(2)T	This command was introduced.

Examples

The following example shows how to enable the debugging of rule 0 for all protocols from source address 10.0.0.1 255.255.255.0 and destination address 10.0.0.2 255.255.255.0:

```
Device# debug crypto engine ism-vpn traffic selector enable 0 10.0.0.1 255.255.255.0 10.0.0.2
 255.255.255.0 1
 ISM-VPN Enable rule 0, s: 10.0.0.1 d: 10.0.0.2 p: 1
```

Related Commands

Command	Description
debug crypto engine ism-vpn	Enables debugging for a Cisco VPN ISM.
debug crypto engine ism-vpn traffic	Enables debugging for Cisco VPN ISM traffic.
debug crypto engine ism-vpn ssl	Enables debugging for Cisco VPN ISM SSL.
show crypto engine accelerator statistic	Displays IPsec encryption statistics and error counters for the onboard hardware accelerator of a device, the IPsec VPN SPA, or Cisco VPN ISM.

debug crypto error

To enable error debugging for a crypto area, use the **debugcryptoerror**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

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

Command Default

Crypto error debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(2)T	This command was introduced.
12.2(18)SXD	This command was integrated into Cisco IOS Release 12.2(!8)SXD.

Usage Guidelines

The**debugcryptoerror**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 **debugcryptoisakmp**, **debugcryptoipsec**, and **debugcryptoengine** commands) will override the **debugcryptoerror** 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 gdoi

To display information about a Group Domain of Interpretation (GDOI) configuration, use the **debugcryptogdoi** command in privileged EXEC mode. To disable crypto GDOI debugging, use the **no** form of this command.

debug crypto gdoi [**all-features [all-levels]**| **detail**| **error**| **event**| **gm**| **infrastructure**| **ks** [**acls**| **coop**]| **packet**| **replay**| **terse**]

no debug crypto gdoi [**all-features [all-levels]**| **detail**| **error**| **event**| **gm**| **infrastructure**| **ks** [**acls**| **coop**]| **packet**| **replay**| **terse**]

Syntax Description

all-features	(Optional) Displays debug data for all features in GDOI. This consists of rekey, cooperative key server, replay, and registration information (for KSs) and rekey, registration, and replay information (for GMs).
all-levels	(Optional) Displays all debug levels (detail, error, event, packet, and terse).
detail	(Optional) Displays detailed debug information.
error	(Optional) Displays information about error debugs.
event	(Optional) Displays user-level information.
gm	(Optional) Displays information about group members.
infrastructure	(Optional) Displays information about the GDOI infrastructure.
ks	(Optional) Displays information about key servers.
acls	(Optional) Displays information about implementation of ACLs.
coop	(Optional) Displays information about cooperative key servers.
packet	(Optional) Displays information about packet-level debugs (administrator-level information).
replay	(Optional) Displays information about the pseudotime stamp that is contained in a packet.
terse	(Optional) Displays lowest-level debugs (message-level information).

Note	The **detail**, **error**, **event**, **packet**, and **terse** keywords can be used with the feature keywords (for example, **gmerror**, **infrastructureerror**, **kscoopevent,replayerror**).

Command Default

Debugging is turned off.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(6)T	This command was introduced.
12.4(11)T	The **detail**, **error**, **event**, **gm**, **infrastructure**, **ks**, **coop**, **packet**, **replay**, and**terse**keywords were added.
Cisco IOS XE Release 2.3	This command was integrated into Cisco IOS XE Release 2.3.
15.1(3)T	This command was modified. The **acls** and **replay** keywords were removed. The **all-features** and **all-levels** keywords were added.
Cisco IOS XE Release 3.8S	This command was modified. The **acls** and **replay** keywords were removed. The **all-features** and **all-levels** keywords were added.

Usage Guidelines

Using this command displays various GDOI debugs. For debugging information for cooperative key servers, use the **debugcryptogdoikscoop** command.

If you do not specify a feature (using the **all-features**, **registration**, **rekey**, **replay**, **coop**, or **infrastructure** keywords), then messages for all features are displayed. If you do not specify a level (using the **detail**, **error**, **event**, **packet**, and **terse** keywords), then the terse level (which also includes the error level) is used.

You can use this command in conjunction with the **debugcryptogdoicondition**command. When these two commands are used together, only those messages that pass through any debug level or feature (specified by the **debugcryptogdoi**command) and pass through any condition (specified by the **debugcryptogdoicondition**command) are displayed.

Examples

The following example shows group member registration debug output:

```
Router# debug crypto gdoi
00:00:40: GDOI:(0:0:N/A:0):GDOI group diffint
00:00:40: %CRYPTO-5-GM_REGSTER: Start registration for group diffint using address 10.0.3.1
00:00:40: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
00:00:40: GDOI:(0:1001:HW:0:3333):beginning GDOI exchange, M-ID of 1167145075
00:00:40: GDOI: Group Number is 3333
00:00:40: GDOI:(0:1001:HW:0:3333):GDOI: GDOI ID sent successfully
```

```
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI SA Payload, message ID + 1167145075
00:00:40: GDOI:(0:1001:HW:0):processing GDOI SA KEK Payload
00:00:40: GDOI:(0:0:N/A:0):    KEK_ALGORITHM 5
00:00:40: GDOI:(0:0:N/A:0):    KEY_LENGTH 24
00:00:40: GDOI:(0:0:N/A:0):    KEY_LIFETIME 299
00:00:40: GDOI:(0:0:N/A:0):    SIG_HASH_ALG 2
00:00:40: GDOI:(0:0:N/A:0):    SIG_ALG 1
00:00:40: GDOI:(0:0:N/A:0):    SIG_KEY_LEN 94
00:00:40: GDOI:(0:0:N/A:0): Completed KEK Processing
00:00:40: GDOI:(0:1001:HW:0):processing GDOI SA TEK Payload
00:00:40: GDOI:(0:1001:HW:0:3333): Completed TEK Processing
00:00:40: GDOI:(0:1001:HW:0):processing GDOI SA TEK Payload
00:00:40: GDOI:(0:1001:HW:0:3333): Completed TEK Processing
00:00:40: GDOI:(0:1001:HW:0:3333):GDOI ACK sent successfully by GM
00:00:40: GDOI:received payload type 18
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI Seq Payload, message_id  1167145075
00:00:40: GDOI:(0:1001:HW:0:3333):Completed SEQ Processing for seq 0
00:00:40: GDOI:received payload type 17
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI KD Payload, message_id  1167145075
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI Key Packet, message_id  38649336
00:00:40: GDOI:(0:1001:HW:0:3333):procesing TEK KD: spi is 56165461, spi
00:00:40: GDOI:(0:1001:HW:0:3333):TEK Integrity Key 20 bytes
00:00:40: GDOI:(0:1001:HW:0:3333):Completed KeyPkt Processing
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI Key Packet, message_id  38649336
00:00:40: GDOI:(0:1001:HW:0:3333):procesing TEK KD: spi is 56165522, spi
00:00:40: GDOI:(0:1001:HW:0:3333):TEK Integrity Key 20 bytes
00:00:40: GDOI:(0:1001:HW:0:3333):Completed KeyPkt Processing
00:00:40: GDOI:(0:1001:HW:0:3333):processing GDOI Key Packet, message_id  38649336
00:00:40: GDOI:(0:1001:HW:0:3333): Processing KEK KD
00:00:40: GDOI:(0:1001:HW:0:3333):KEK Alg Key 32 bytes
00:00:40: GDOI:(0:1001:HW:0:3333):KEK Sig Key 94 bytes
00:00:40: GDOI:(0:1001:HW:0:3333):Completed KeyPkt Processing
00:00:40: %GDOI-5-GM_REGS_COMPL: Registration complete for group diffint using address
10.0.3.1
enc(config-if)#
00:00:40: GDOI:(0:0:N/A:0):Registration installed 2 new ipsec SA(s) for group diffint.
```

The following output example shows key server registration debugs:

```
Router# debug crypto gdoi
00:00:40: GDOI:(0:1001:HW:0):processing GDOI ID payload, message ID = 1167145075
00:00:40: GDOI:(0:1001:HW:0):The GDOI ID is a Number: 3333
00:00:40: GDOI:(0:0:N/A:0): Adding KEK Policy to the current ks_group
00:00:40: GDOI:(0:0:N/A:0):Setting MULTICAST TEK rekey lifetime 30
00:00:40: GDOI:(0:0:N/A:0):Setting MULTICAST TEK rekey lifetime 30
00:00:40: GDOI:(0:1001:HW:0:3333):GDOI SA sent successfully by KS
00:00:40: GDOI:(0:1001:HW:0:3333):GDOI KD sent successfully by KS
```

The following output example shows group member rekey debugs:

```
Router# debug crypto gdoi
00:02:00: GDOI:(0:1002:HW:0):Received Rekey Message!
00:02:00: GDOI:(0:1002:HW:0):Signature Valid!
00:02:00: GDOI:received payload type 18
00:02:00: GDOI:(0:1002:HW:0):processing GDOI Seq Payload, message_id  0
00:02:00: GDOI:(0:1002:HW:0):Completed SEQ Processing for seq 8
00:02:00: GDOI:(0:1002:HW:0):processing GDOI SA Payload, message ID + 0
00:02:00: GDOI:(0:1002:HW:0):processing GDOI SA KEK Payload
00:02:00: GDOI:(0:1002:HW:0):    KEK_ALGORITHM 5
00:02:00: GDOI:(0:1002:HW:0):    KEY_LENGTH 24
00:02:00: GDOI:(0:1002:HW:0):    KEY_LIFETIME 219
00:02:00: GDOI:(0:1002:HW:0):    SIG_HASH_ALG 2
00:02:00: GDOI:(0:1002:HW:0):    SIG_ALG 1
00:02:00: GDOI:(0:1002:HW:0): Completed KEK Processing
00:02:00: GDOI:(0:1002:HW:0):processing GDOI SA TEK Payload
00:02:00: GDOI:(0:1002:HW:0): Completed TEK Processing
00:02:00: GDOI:(0:1002:HW:0):processing GDOI SA TEK Payload
00:02:00: GDOI:(0:1002:HW:0): Completed TEK Processing
00:02:00: GDOI:received payload type 17
00:02:00: GDOI:(0:1002:HW:0):processing GDOI KD Payload, message_id  0
00:02:00: GDOI:(0:1002:HW:0):processing GDOI Key Packet, message_id  38649336
00:02:00: GDOI:(0:1002:HW:0):procesing TEK KD: spi is 49193284, spi
```

```
00:02:00: GDOI:(0:1002:HW:0):TEK Integrity Key 20 bytes
00:02:00: GDOI:(0:1002:HW:0):Completed KeyPkt Processing
enc(config-if)#
00:02:00: GDOI:(0:1002:HW:0):processing GDOI Key Packet, message_id  38649336
00:02:00: GDOI:(0:1002:HW:0):procesing TEK KD: spi is 49193345, spi
00:02:00: GDOI:(0:1002:HW:0):TEK Integrity Key 20 bytes
00:02:00: GDOI:(0:1002:HW:0):Completed KeyPkt Processing
00:02:00: GDOI:(0:1002:HW:0):processing GDOI Key Packet, message_id  38649336
00:02:00: GDOI:(0:1002:HW:0): Processing KEK KD
00:02:00: GDOI:(0:1002:HW:0):Completed KeyPkt Processing
```

debug crypto gdoi condition

To configure conditional filters (based on groups and peers) for GDOI debugging, use the **debugcryptogdoicondition**command in privileged EXEC mode. To disable conditional filters, use the **no** form of this command.

debug crypto gdoi condition {[**group** *group-name*] [**peer** {**address ipv4** *ipv4-address-of-peer*| **hostname ipv4** *ipv4-hostname*}]| **reset**| **unmatched**}

no debug crypto gdoi condition {[**group** *group-name*] [**peer** {**address ipv4** *ipv4-address-of-peer*| **hostname ipv4** *ipv4-hostname*}]| **reset**| **unmatched**}

Syntax Description

group *group-name*	(Optional) Displays information for a specific group.
address ipv4 *ipv4-address-of-peer*	(Optional) Displays information for a specific IPv4 peer.
hostname ipv4 *ipv4-hostname*	(Optional) Displays information for a specific IPv4 host.
reset	(Optional) Removes all debugging conditions.
unmatched	(Optional) Displays debug messages if no context is available.

Command Default

Conditional bugging is turned off.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(3)T	This command was introduced.
Cisco IOS XE Release 3.8S	This command was integrated into Cisco IOS XE Release 3.8S.

Usage Guidelines

This command lets you filter the number of debug messages to make it easier to identify the problem of a particular group member (GM). This command is useful when many (such as thousands of) GMs are registered to a key server on which debugging is enabled.

You can use this command in conjunction with the **debugcryptogdoi**command. When these two commands are used together, only those messages that pass through any debug level or feature (specified by the

debugcryptogdoicommand) and pass through any condition (specified by the **debugcryptogdoicondition**command) are displayed.

You can use this command multiple times to configure conditional debugging for any number of GMs. You can have more than one group configured and more than one GM registered to each group.

If no GM is specified, debug messages will appear for all GMs (this occurs automatically when all conditional filtering is removed).

Also, this command lets you filter per cooperative key server on each key server. This command is useful when there are many key servers in a system.

The **unmatched** keyword displays messages when there is insufficient information to perform conditional debugging. The **unmatched** keyword is enabled for the error, terse, and event debug levels.

To remove the debug messages for a GM or a key server, use the **no** form of the same command.

Examples

The following example shows how to configure a conditional filter for a GDOI group named group1:

```
Router# debug crypto gdoi condition group group1
```

The following output example shows how to configure a conditional filter for IPv4 peers:

```
Router# debug crypto gdoi condition peer
```

Related Commands

Command	Description
debug crypto condition	Defines conditional debug filters for IP Security (IPSec) tunnels.

debug crypto ha

To display crypto high availability debugging information, use the **debugcryptoha**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 **debugcryptoha** 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
debug crypto ipsec ha	Enables debugging messages for IPSec high availability.
debug crypto isakmp ha	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.

Command Default

Debugging for IPv6 IPSec events is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)T	This command was introduced.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

Usage Guidelines

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

Examples

The following example enables the display of IPSec events for IPv6 networks:

```
Router# debug crypto ipv6 ipsec
```

Related Commands

Command	Description
debug crypto engine	Displays debugging messages about crypto engines, which perform encryption and decryption.
debug crypto ipv6 packet	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.
debug crypto socket	Displays communication between the client and IPSec during policy setup and removal processes.

Command	Description
debug ipv6 ospf authentication	Displays 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.

Command Default

Debugging for IPv6 IPSec packets is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)T	This command was introduced.
Cisco IOS XE Release 2.1	This command was introduced on Cisco ASR 1000 Series Routers.

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.

This command examines the content of every IPv6 packet and may slow network performance.

Examples

This example shows the ouptut 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
debug crypto engine	Displays debugging messages about crypto engines, which perform encryption and decryption.
debug crypto ipv6 ipsec	Displays IPSec events for IPv6 networks.
debug crypto socket	Displays communication between the client and IPSec during policy setup and removal processes.

debug crypto ikev2

To enable Internet Key Exchange Version 2(IKEv2) debug messages, use the **debugcryptoikev2**command in privileged EXEC mode.

debug crypto ikev2 [**error**| **terse**| **event**| **packet**| **detail**]

no debug crypto ikev2 [**error**| **terse**| **event**| **packet**| **detail**]

Syntax Description

error	(Optional) Enables debug messages capturing IKEv2 errors.
terse	(Optional) Enables debug messages capturing IKEv2 message exchanges.
event	(Optional) Enables debug messages capturing IKEv2 packet description, contents, and policy matching.
packet	(Optional) Enables debug messages capturing IKEv2 packet dump.
detail	(Optional) Enables debug messages capturing IKEv2 state machine events.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.1(1)T	This command was introduced.
Cisco IOS XE Release 3.3S	This command was integrated into Cisco IOS XE Release 3.3S.

Usage Guidelines

Use this command to enable IKEv2 debug messages. IKEv2 uses following debug levels.

- Level 1--error
- Level 2--terse
- Level 3--event
- Level 4--packet
- Level 5--detail

Note Level 1 is the lowest level and is least verbose and level 5 is highest level. Enabling debug at a higher level enables debug at all lower levels. You can selectively disable a debug level and enable conditional debug using the **debugcryptocondition** command.

Examples

The following example shows that the IKEv2 debugging is enabled:

```
Router# debug crypto ikev2
debugging is on
```

Related Commands

Command	Description
debug crypto condition	Enables conditional debugging.

debug crypto ipsec

To display IP security (IPsec) events, use the **debugcryptoipsec**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 (#)

Command History

Release	Modifications
11.3 T	This command was introduced.
Cisco IOS XE Release 2.6	This command was integrated into Cisco IOS XE Release 2.6.

Examples

The following is sample output from the **debugcryptoipsec** 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 **permitiphost172.21.114.123host172.21.114.67** command. IPsec is configured to first use the set esp-des w/esp-md5-hmac, but it will also use ah-sha-hmac on a secondary basis.

```
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) prompts 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 525075940ld for SA
        from 172.21.114.67   to 172.21.114.123  for prot 2
```

IKE will verify whether IPsec accepts the SA proposal. In this example, it is the SA proposal sent by the local IPsec that is accepted first.

```
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 then 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
 sa_lifetime(k/sec)= (4445923/500)
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
 sa_lifetime(k/sec)= (4445923/500)
```

The following is sample output from the **debugcryptoipsec** command as seen on the peer router. In this example, IKE verifies whether IPsec 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 prompts for SPIs.

```
00:26:15: IPSEC(key_engine): got a queue event...
00:26:15: IPSEC(spi_response): getting spi 59315364ld 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. IKE then notifies 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 then 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
 sa_lifetime(k/sec)= (4445923/500)
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
 sa_lifetime(k/sec)= (4445923/500)
```

debug crypto ipsec client ezvpn

To display information about voice control messages that have been captured by the Voice DSP Control Message Logger and about Cisco Easy VPN remote connections, use the **debugcryptoipsecclientezvpn**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.

Command Default

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.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2SX	This command is supported in the Cisco IOS 12.2SX family of releases. Support in a specific 12.2SX release is dependent on your feature set, platform, and platform hardware.

Usage Guidelines

To force the Voice DSP Contol Message Logger to reestablish the virtual private network (VPN) connections, use the **clearcryptosaandclearcryptoisakmp** 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 Contol 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 **clearcryptoipsecclientezvpn** 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: 172.16.0.250
3d17h:         DNS Secondary: 172.16.0.251
3d17h:         NBMS/WINS Primary: 172.16.0.252
3d17h:         NBMS/WINS Secondary: 172.16.0.253
3d17h:         Split Tunnel List: 1
3d17h:               Address    : 172.16.0.128
3d17h:               Mask       : 255.255.255.128
3d17h:               Protocol   : 0x0
3d17h:               Source Port: 0
3d17h:               Dest Port  : 0
3d17h:         Split Tunnel List: 2
3d17h:               Address    : 172.16.1.128
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 **nocryptoipsecclientezvpn** 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
debug crypto ipsec	Displays debugging messages for generic IPSec events.
debug crypto isakmp	Displays debugging messages for IKE events.

debug crypto ipsec ha

To enable debugging messages for IP Security (IPSec) high availability, use the **debugcryptoipsecha**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

detail	(Optional) Displays detailed debug information.
update	(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 **debugcryptoipsecha** command for both the active and stanby 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_recv_add_sas):HA mgr wants to insert the
 following bundle
*Sep 29 17:03:01.031:IPSec HA (crypto_ha_ipsec_mgr_recv_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_recv_add_sas):HA mgr wants to insert the
 following bundle
*Sep 29 17:06:44.063:IPSec HA (crypto_ha_ipsec_mgr_recv_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_recv_update_sa):called
*Sep 29 17:27:28.887:IPSec HA(crypto_ha_ipsec_mgr_recv_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_recv_update_sa):called
*Sep 29 17:29:28.915:IPSec HA(crypto_ha_ipsec_mgr_recv_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_recv_update_sa):called
*Sep 29 17:30:28.939:IPSec HA(crypto_ha_ipsec_mgr_recv_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_recv_update_sa):called
*Sep 29 17:32:28.955:IPSec HA(crypto_ha_ipsec_mgr_recv_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
debug crypto ha	Displays crypto high availability debugging information.

Command	Description
debug crypto isakmp ha	Enables debugging messages for ISAKMP high availability.

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**| **errors**| **kmi** *number*]

no debug crypto isakmp [**aaa**| **errors**| **kmi**]

Syntax Description

aaa	(Optional) Specifies accounting events.
errors	(Optional) Enables error events.
kmi *number*	(Optional) Captures key management interface (KMI) messages.

Command Modes

Privileged EXEC (#)

Command History

Release	Modifications
11.3 T	This command was introduced.
12.2(15)T	The **aaa** keyword was added.
12.2(18)SXD	This command was integrated into Cisco IOS Release 12.2(18)SXD.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
Cisco IOS XE Release 2.6	This command was integrated into Cisco IOS XE Release 2.6.
15.3(2)T	This command was modified. The **kmi** *number* keyword-argument pair was added.

Usage Guidelines

IKE is a key management protocol standard that is used in conjunction with the IPsec to configure basic IPsec VPNs. IPsec can be configured without IKE, but IKE enhances IPsec by providing additional features, flexibility, and ease of configuration for the IPsec standard. IKE is a hybrid protocol that implements the Oakley key exchange and Skeme key exchange inside the Internet Security Association and Key Management Protocol (ISAKMP) framework.

The **debug crypto isakmp kmi** command must be enabled prior to any traffic that triggers sessions. This command enables capturing KMI sequence events along with timestamps for a particular IKE security association (SA) and corresponding IPsec SA. The *number* argument denotes the number of KMI messages to be captured. The KMI sequence is displayed in the output of the **show crypto isakmp peers detail** command.

Examples

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

IKE negotiates its own SA.

```
Device# 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
```

When IKE finds a matching policy, each peer in the network uses the IKE SA to authenticate another 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 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.
```

After finding a matching IPsec transform set for the two peers, an IPsec SA is 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:

```
Device# debug crypto isakmp aaa

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

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
```

The following example shows how to set the KMI sequence for a value of 10. The KMI sequence is displayed in the output of the **show crypto isakmp peers detail** command.

```
Device# debug crypto isakmp kmi 10
Crypto ISAKMP KMI debugging is on

Device# show crypto isakmp peers detail

Peer: 1.1.1.2 Port: 500 Local: 1.1.1.1
 Phase1 id: 1.1.1.2
  flags:
 NAS Port: 0 (Normal)
  IKE SAs: 1 IPSec SA bundles: 2
  Peer Handle: 0x80000005
  last_locker: 0xD198B32, last_last_locker: 0x0
  last_unlocker: 0x0, last_last_unlocker: 0x0

KMI messages(Time of capture - KMI message)

*Mar  7 12:20:28.124     KEY_ENG_REQUEST_SAS
*Mar  7 12:20:28.165     KEY_MGR_CREATE_IPSEC_SAS
*Mar  7 12:20:28.165     KEY_ENG_NOTIFY_QOS_GROUP
*Mar  7 12:20:28.165     KEY_ENG_NOTIFY_INCR_COUNT
*Mar  7 12:20:58.124     KEY_ENG_REQUEST_SAS
*Mar  7 12:20:58.126     KEY_MGR_CREATE_IPSEC_SAS
*Mar  7 12:20:58.126     KEY_ENG_NOTIFY_QOS_GROUP
```

Related Commands

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

debug crypto isakmp ha

To enable debugging messages for Internet Security Association and Key Management Protocol (ISAKMP) high availability, use the **debugcryptoisakmpha**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 **debugcryptoisakmpha**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

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

debug crypto key-exchange

To show Digital Signature Standard (DSS) public key exchange messages, use the **debugcryptokey-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 *cryptoengine*. 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 **configcryptokey-exchange** command is not successful, try to exchange DSS public keys again after enabling the **debugcryptokey-exchange** command to help you diagnose the problem.

Examples

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

```
Router# debug crypto key-exchange
CRYPTO-KE: Sent 4 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 2 bytes.
CRYPTO-KE: Sent 64 bytes.
Router# debug crypto key-exchange
CRYPTO-KE: Received 4 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 2 bytes.
CRYPTO-KE: Received 49 bytes.
CRYPTO-KE: Received 15 bytes.
```

Related Commands

Command	Description
debug crypto sesmgmt	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 **debugcryptomib** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto mib [**detail**| **error**]

no debug crypto mib [**detail**| **error**]

Syntax Description

detail	(Optional) Displays different events as they occur in the IPsec MIB subsystem. **Note** Because the output for this keyword can be quite long, due consideration should be given to enabling **debugcryptomibdetail**.
error	(Optional) Displays error events in the MIB agent.

Command Default

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.
12.4(4)T	The **detail** and **error** keywords were added.

Examples

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

```
Router# debug crypto mib
Crypto IPSec Mgmt Entity debugging is on
```

The following example shows that detailed information about events that are occurring in the subsystem has been requested:

```
Router# debug crypto mib detail
```

The following example shows that information has been requested about error events in the MIB agent:

Router# **debugcryptomiberror**

Related Commands

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

debug crypto pki messages

To display debugging messages for the details of the interaction (message dump) between the certification authority (CA) and the router, use the **debugcryptopkimessages** 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.

Command Default

Disabled.

Command Modes

Privileged EXEC

Command History

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

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 **debugcryptopkimessagescommand**:

```
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
00:48:23:00 00
00:48:24:Received pki message:1778 types
.
.
.
```

Related Commands

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

debug crypto pki server

To enable debugging for a crypto public key infrastructure (PKI) certificate server, use the **debugcryptopkiserver**command in privileged EXEC mode. To disable debugging, use the **no** form of this command.

debug crypto pki server

no debug crypto pki server

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(4)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following example shows how to enable debugging for a certificate server. This example also contains sample debug messages, which allow users to troubleshoot the various certificate-request-related stages and tasks that are handled by the certificate server.

```
Router# debug crypto pki server
Crypto PKI Certificate Server debugging is on
Oct 15 19:50:41.003:CRYPTO_CS:old RA cert flag 0x4
Oct 15 19:50:41.003:CRYPTO_CS:new RA cert flag 0x1000C
Oct 15 19:50:41.003:CRYPTO_CS:nvram filesystem
Oct 15 19:50:41.279:CRYPTO_CS:serial number 0x1 written.
Oct 15 19:50:53.383:CRYPTO_CS:created a new serial file.
Oct 15 19:50:53.383:CRYPTO_CS:SCEP server started
Oct 15 19:50:53.419:%SYS-5-CONFIG_I:Configured from console by console
Oct 15 19:50:53.731:CRYPTO_CS:received a SCEP GetCACert request
Oct 15 19:50:53.739:CRYPTO_CS:CA certificate sent
Oct 15 19:50:54.355:CRYPTO_CS:received a SCEP GetCACert request
Oct 15 19:50:54.363:CRYPTO_CS:CA certificate sent
Oct 15 19:50:57.791:CRYPTO_CS:received a SCEP request
Oct 15 19:50:57.795:CRYPTO_CS:read SCEP:registered and bound service
SCEP_READ_DB_8
Oct 15 19:50:57.947:CRYPTO_CS:scep msg type - 19
Oct 15 19:50:57.947:CRYPTO_CS:trans id -
3673CE2FF0235A4AE6F26242B00A4BB4
Oct 15 19:50:58.679:CRYPTO_CS:read SCEP:unregistered and unbound
service SCEP_READ_DB_8
Oct 15 19:50:58.683:CRYPTO_CS:received an enrollment request
Oct 15 19:50:58.691:CRYPTO_CS:request has been authorized, transaction
id=3673CE2FF0235A4AE6F26242B00A4BB4
Oct 15 19:50:58.699:CRYPTO_CS:byte 2 in key usage in PKCS#10 is 0x7
Oct 15 19:50:58.699:CRYPTO_CS:signature
Oct 15 19:50:58.699:CRYPTO_CS:key_usage is 1
Oct 15 19:50:58.703:CRYPTO_CS:enrollment request with pendingID 1 sent
to the CA
Oct 15 19:50:58.707:CRYPTO_CS:write SCEP:registered and bound service
```

```
SCEP_WRTE_DB_8
Oct 15 19:50:59.531:CRYPTO_CS:write SCEP:unregistered and unbound
service SCEP_WRTE_DB_8
.....
Oct 15 19:53:08.403:CRYPTO_CS:CS_RA_REQUEST:save cert in dbase,
pending id = 2
Oct 15 19:53:08.403:CRYPTO_CS:enrollment request 2 granted
Oct 15 19:53:08.403:CRYPTO_PKI:All enrollment requests completed for
trustpoint ra.
Oct 15 19:53:08.403:%CRYPTO-6-CERTRET:Certificate received from
Certificate Authority
Oct 15 19:53:08.403:CRYPTO_PKI:All enrollment requests completed for
trustpoint ra.
Oct 15 19:53:08.403:CRYPTO_PKI:All enrollment requests completed for
trustpoint ra.
Oct 15 19:53:08.407:CRYPTO_PKI:All enrollment requests completed for
trustpoint ra.
Oct 15 19:53:19.623:IPSEC(key_engine):major = 1
Oct 15 19:53:19.623:IPSEC(key_engine):expired_timer:skip ...
Oct 15 19:53:35.707:CRYPTO_CS:received a SCEP request
Oct 15 19:53:35.711:CRYPTO_CS:read SCEP:registered and bound service
SCEP_READ_DB_14
Oct 15 19:53:35.859:CRYPTO_CS:scep msg type - 20
Oct 15 19:53:35.859:CRYPTO_CS:trans id -
4D774FFE2F7CA9991A7F6A785E803E77
Oct 15 19:53:36.591:CRYPTO_CS:read SCEP:unregistered and unbound
service SCEP_READ_DB_14
Oct 15 19:53:36.595:CRYPTO_CS:received an enrollment request
Oct 15 19:53:36.595:CRYPTO_CS:write SCEP:registered and bound service
SCEP_WRTE_DB_14
Oct 15 19:53:37.623:CRYPTO_CS:write SCEP:unregistered and unbound
service SCEP_WRTE_DB_14
Oct 15 19:53:37.631:CRYPTO_CS:Certificate sent to requestor
```

Related Commands

Command	Description
crypto pki server	Enables a Cisco IOS certificate server and enters certificate server configuration mode.

debug crypto pki transactions

To display debugging messages for the trace of interaction (message type) between the certification authority (CA) and the router, use the **debugcryptopkitransactions** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto pki transactions

no debug crypto pki transactions

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

Use the debug crypto pki transactions command to display debugging messages pertaining to public key infrastructure (PKI) certificates. The messages will show status information during certificate enrollment and verification.

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

Examples

The following example, which authenticates and enrolls a CA, contains sample output for the **debugcryptopkitransactionscommand**:

```
Router(config)# crypto ca authenticate msca
Certificate has the following attributes:
Fingerprint:A5DE3C51 AD8B0207 B60BED6D 9356FB00
% Do you accept this certificate? [yes/no]:y
Router# debug crypto pki transactions
00:44:00:CRYPTO_PKI:Sending CA Certificate Request:
GET /certsrv/mscep/mscep.dll/pkiclient.exe?operation=GetCACert&message=msca HTTP/1.0
00:44:00:CRYPTO_PKI:http connection opened
00:44:01:CRYPTO_PKI:HTTP response header:
 HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:50:59 GMT
Content-Length:2693
Content-Type:application/x-x509-ca-ra-cert

Content-Type indicates we have received CA and RA certificates.

00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
```

```
certificate status

00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status

00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:transaction GetCACert completed
00:44:01:CRYPTO_PKI:CA certificate received.
00:44:01:CRYPTO_PKI:CA certificate received.
Router(config)# crypto ca enroll msca
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.
Password:
Re-enter password:
% The subject name in the certificate will be:Router.cisco.com
% Include the router serial number in the subject name? [yes/no]:n
% Include an IP address in the subject name? [yes/no]:n
Request certificate from CA? [yes/no]:y
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.
Router(config)#
00:44:29:CRYPTO_PKI:transaction PKCSReq completed
00:44:29:CRYPTO_PKI:status:
00:44:29:CRYPTO_PKI:http connection opened
00:44:29:CRYPTO_PKI: received msg of 1924 bytes
00:44:29:CRYPTO_PKI:HTTP response header:
 HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:51:28 GMT
Content-Length:1778
Content-Type:application/x-pki-message
00:44:29:CRYPTO_PKI:signed attr:pki-message-type:
00:44:29:13 01 33
00:44:29:CRYPTO_PKI:signed attr:pki-status:
00:44:29:13 01 30
00:44:29:CRYPTO_PKI:signed attr:pki-recipient-nonce:
00:44:29:04 10 B4 C8 2A 12 9C 8A 2A 4A E1 E5 15 DE 22 C2 B4 FD
00:44:29:CRYPTO_PKI:signed attr:pki-transaction-id:
00:44:29:13 20 34 45 45 41 44 42 36 33 38 43 33 42 42 45 44 45 39 46
00:44:29:34 38 44 33 45 36 39 33 45 33 43 37 45 39
00:44:29:CRYPTO_PKI:status = 100:certificate is granted
00:44:29:CRYPTO__PKI:All enrollment requests completed.
00:44:29:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

Related Commands

Command	Description
crypto ca authenticate	Authenticates the CA (by getting the certificate of the CA).
crypto ca enroll	Obtains the certificate of your router from the CA.
debug crypto pki messages	Displays debugging messages for details of the interaction (message dump) between the CA and the router.
show crypto ca certificates	Displays information about your certificate, the certificate of the CA, and any RA certificates.

debug crypto provisioning

To display information about an easy secure device provisioning (SDP) operation, use the **debugcryptoprovisioning**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto provisioning

no debug crypto provisioning

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.3(14)T	This command replaced the **debugcryptowui**command.

Usage Guidelines

For more detailed information about authentication, authorization, and accounting (AAA), use the **debugaaaauthorization** command.

Examples

The following is sample output from the **debugcryptoprovisioning**command. The output includes explanations of the process.

```
Router# debug crypto provisioning
Petitioner device
! The user starts the Welcome phase.
Nov  7 03:15:48.171: CRYPTO_WUI_TTI: received welcome get request.
! The router generates a Rivest, Shamir, and Adelman (RSA) keypair for future enrollment.
Nov  7 03:15:48.279: CRYPTO_WUI_TTI: keyhash 'A506BE3B83C6F4B4A6EFCEB3D584AACA'
! The TTI transaction is completed.
Nov  7 03:16:10.607: CRYPTO_WUI_TTI: received completion post request.
Registrar device
!. During the introduction phase, the browser prompts for login information.
06:39:18: CRYPTO_WUI_TTI: received introduction post request.
06:39:18: CRYPTO_WUI_TTI: checking AAA authentication (ipsecca_script_aaalist, ttiuser)
! This happens if the user types in the wrong username or password.
06:39:19: CRYPTO_WUI_TTI: authentication declined by AAA, or AAA server not found - 0x3
06:39:19: CRYPTO_WUI_TTI: aaa query fails!
! The user re-enters login information.
06:39:19: CRYPTO_WUI_TTI: received introduction post request.
06:39:19: CRYPTO_WUI_TTI: checking AAA authentication (ipsecca_script_aaalist, ttiuser)
06:39:20: CRYPTO_WUI_TTI: checking AAA authorization (ipsecca_script_aaalist, ttiuser)
! The login attempt succeeds and authorization information is retrieved from the AAA database.
06:39:21: CRYPTO_WUI_TTI: aaa query ok!
```

```
! These attributes are inserted into the configuration template.
06:39:21: CRYPTO_WUI_TTI: building TTI av pairs from AAA attributes
06:39:21: CRYPTO_WUI_TTI: "subjectname" = "CN=user, O=company, C=country"
06:39:21: CRYPTO_WUI_TTI: "$1" = "ntp server 192.0.2.1"
06:39:21: CRYPTO_WUI_TTI: "$2" = "hostname user-vpn"
! The registrar stores this subject name and overrides the subject name in the subsequent
enrollment request.
06:39:21: CRYPTO_WUI_TTI: subjectname=CN=user, O=company, C=country
! The registrar stores this key information so that it may be used to automatically grant
the subsequent enrollment request.
06:39:21: CRYPTO_WUI_TTI: key_hash=A506BE3B83C6F4B4A6EFCEB3D584AACA
```

Related Commands

Command	Description
authentication list (tti-registrar)	Authenticates the introducer in an SDP operation.
authorization list (tti-registrar)	Specifies the appropriate authorized fields for the certificate subject name and list of template variables to be expanded into the Cisco IOS CLI snippet that is sent back to the petitioner in an SDP operation.
debug aaa authorization	Displays information on AAA TACACS+ authorization.
template config	Specifies a remote URL for a Cisco IOS CLI configuration template.
template username	Establishes a template username and password to access the configuration template on the file system.

debug crypto sesmgmt

To show connection setup messages and their flow through the router, use the **debugcryptosesmgmt**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug crypto sesmgmt

no debug crypto sesmgmt

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 *cryptoengine*. The crypto engine performs authentication through Digital Signature Standard (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 receives a message with a DSS signature knows exactly who sent the message.

When connections are not completing, use the **debugcryptosesmgmt** command to follow the progress of connection messages as a first step in diagnosing the problem. You see a record of each connection message as the router discovers it, and can track its progress through the necessary signing, verifying, and encryption session setup operations. Other significant connection setup events, such as the pregeneration of Diffie-Hellman public numbers, are also shown. For information on Diffie-Hellman public numbers, refer to the *Cisco IOS Security Configuration Guide* .

Also use the **showcryptoconnections** command to display additional information on connections.

Examples

The following is sample output from the **debugcryptosesmgmt** command. The first shows messages from a router that initiates a successful connection. The second shows messages from a router that receives a connection.

```
Router# debug crypto sesmgmt
CRYPTO: Dequeued a message: Inititate_Connection
CRYPTO: DH gen phase 1 status for conn_id 2 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: send_nnc_req:   NNC Echo Request sent
CRYPTO: Dequeued a message: CRM
CRYPTO: DH gen phase 2 status for conn_id 2 slot 0:OK
CRYPTO: Verify done. Status=OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.163, d=172.21.114.162
CRYPTO-SDU: recv_nnc_rpy:   NNC Echo Confirm sent
CRYPTO: Create encryption key for conn_id 2 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 2 (slot 0)
Router# debug crypto sesmgmt
CRYPTO: Dequeued a message: CIM
CRYPTO: Verify done. Status=OK
CRYPTO: DH gen phase 1 status for conn_id 1 slot 0:OK
CRYPTO: DH gen phase 2 status for conn_id 1 slot 0:OK
CRYPTO: Signing done. Status:OK
CRYPTO: ICMP message sent: s=172.21.114.162, d=172.21.114.163
CRYPTO-SDU: act_on_nnc_req: NNC Echo Reply sent
```

```
CRYPTO: Create encryption key for conn_id 1 slot 0:OK
CRYPTO: Replacing -2 in crypto maps with 1 (slot 0)
CRYPTO: Dequeued a message: CCM
CRYPTO: Verify done. Status=OK
```

Related Commands

Command	Description
debug crypto key-exchange	Displays DSS public key exchange messages.

debug csm neat

To turn on debugging for all Call Switching Module (CSM) Voice over IP (VoIP) calls, use the **debugcsmneat**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug csm neat [*slot*| *dspm*| *dsp*| *dsp-channel*]

no debug csm neat [*slot*| *dspm*| *dsp*| *dsp-channel*]

Syntax Description

slot | *dspm* | *dsp* | *dsp-channel*	(Optional) Identifies the location of a particular digital signal processor (DSP) channel.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(9)T	This command was introduced.
12.2SX	This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.

Usage Guidelines

The **debugcsmneat**command turns on debugging for all CSM VoIP calls. If no arguments are specified, debugging is enabled for all voice calls.

Note The **debugcsmneat** command does not display any information if you try to debug ISDN voice calls. To view debugging information about ISDN calls, use the **debugcdapi** command.

The **nodebugcmsneat** command turns off debugging information for all voice calls.

If the *slot,dspm,dsp,*or*dsp-channel* arguments are specified (if the specified DSP channel is engaged in a CSM call), CSM call-related debugging information is turned on for this channel. The **no**form of this command turns off debugging for that particular channel.

Examples

The following examples show sample output from the **debugcsmneat** command. The following shows that CSM has received an event from ISDN.

```
Router# debug csm neat
March 18 04:05:07.052: EVENT_FROM_ISDN::dchan_idb=0x60D7B6B8, call_id=0xCF, ces=0x1
bchan=0x0, event=0x1, cause=0x0
```

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

Table 11: debug csm neat Field Descriptions

Field	Description
dchan_idb	Indicates the address of the hardware interface description block (IDB) for the D channel.
call_id	Indicates the call ID assigned by ISDN.
call-id	Indicates the call ID assigned by ISDN
bchan	Indicates the number of the B channel assigned for this call
cause	Indicates the ISDN event cause

The following example shows that CSM has allocated the CSM voice control block for the DSP device on slot 1 port 10 for this call.

```
March 18 04:05:07.052: VDEV_ALLOCATE: slot 1 and port 10 is allocated.
```

In this example, CSM must first allocate the CSM voice control block to initiate the state machine. After the voice control block has been allocated, CSM obtains from the DSP Resource Manager the actual DSP channel that will be used for the call. At that point, CSM will switch to the actual logical port number. The slot number in this example refers to the physical slot on the Cisco AS5400 access server. The port number is the logical DSP number interpreted as listed in the table below.

Table 12: Logical DSP Numbers

Logical Port	Physical DSP Channel		
0	DSPRM 1	DSP 1	DSP channel 1
1	DSPRM 1	DSP 1	DSP channel 2
2	DSPRM 1	DSP 2	DSP channel 1
3	DSPRM 1	DSP 2	DSP channel 2
4	DSPRM 1	DSP 3	DSP channel 1
5	DSPRM 1	DSP 3	DSP channel 2
6	DSPRM 1	DSP 4	DSP channel 1
7	DSPRM 1	DSP 4	DSP channel 2
8	DSPRM 1	DSP 5	DSP channel 1
9	DSPRM 1	DSP 5	DSP channel 2

Logical Port	Physical DSP Channel		
10	DSPRM 1	DSP 6	DSP channel 1
11	DSPRM 1	DSP 6	DSP channel 2
12	DSPRM 2	DSP 1	DSP channel 1
13	DSPRM 2	DSP 1	DSP channel 2
14	DSPRM 2	DSP 2	DSP channel 1
15	DSPRM 2	DSP 2	DSP channel 2
16	DSPRM 2	DSP 3	DSP channel 1
17	DSPRM 2	DSP 3	DSP channel 2
18	DSPRM 2	DSP 4	DSP channel 1
19	DSPRM 2	DSP 4	DSP channel 2
20	DSPRM 2	DSP 5	DSP channel 1
21	DSPRM 2	DSP 5	DSP channel 2
22	DSPRM 2	DSP 6	DSP channel 1
23	DSPRM 2	DSP 6	DSP channel 2
48	DSPRM 5	DSP 1	DSP channel 1
49	DSPRM 5	DSP 1	DSP channel 2
50	DSPRM 5	DSP 2	DSP channel 1
51	DSPRM 5	DSP 2	DSP channel 2
52	DSPRM 5	DSP 3	DSP channel 1
53	DSPRM 5	DSP 3	DSP channel 2
54	DSPRM 5	DSP 4	DSP channel 1
55	DSPRM 5	DSP 4	DSP channel 2
56	DSPRM 5	DSP 5	DSP channel 1
57	DSPRM 5	DSP 5	DSP channel 2

Logical Port	Physical DSP Channel		
58	DSPRM 5	DSP 6	DSP channel 1
59	DSPRM 5	DSP 6	DSP channel 2

The following example shows that the function csm_vtsp_init_tdm() has been called with a voice control block of address 0x60B8562C. This function will be called only when the call is treated as a voice call.

```
March 18 04:05:07.052: csm_vtsp_init_tdm (voice_vdev=0x60B8562C)
```

The following example shows that CSM has obtained a DSP channel from the DSP Resource Manager:

```
March 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 2, dsp 5,
dsp_channel 1csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 9, bank 0,
bp_channel 10
```

The table below describes the significant fields shown in the DSP channel initialized TDM display.

Table 13: debug csm neat TDM Channel Field Descriptions

Field	Description
TDM slot 1, dspm 2, dsp 5, dsp_channel 1	Indicates the physical DSP channel that will be used for this call.
TDM stream 5, channel 9, bank 0, bp_channel 10	Indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 9 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 10 means that the backplane stream 0 and backplane channel #1 are assigned to this DSP.

The following shows that CSM received an incoming call event from ISDN:

```
March 18 04:05:07.052: EVENT_FROM_ISDN:(00CF): DEV_INCALL at slot 1 and port 20
```

Slot 1, port 20 means the logical DSP channel 20 (mapped to DSPRM 2, DSP 5, DSP channel 1).

The following shows that the DEV_INCALL message been translated into a CSM_EVENT_ISDN_CALL message:

```
March 18 04:05:07.052: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 20
```

This message is passed to the CSM central state machine while it is in the CSM_IDLE state and is in the CSM_PROC_IDLE procedure. The logical DSP channel port 20 on slot 1 is used to handle this call.

The following shows that CSM has invoked the vtsp_ic_notify() function with a CSM voice call control block 0x60B8562C.

```
March 18 04:05:07.052: vtsp_ic_notify : (voice_vdev= 0x60B8562C)
```

Inside this function, CSM will send a SETUP INDICATION message to the VTSP. This function will be invoked only if the call is a voice call.

The following shows that CSM received a SETUP INDICATION RESPONSE message from the VTSP as an acknowledgment.

```
March 18 04:05:07.056: csm_vtsp_call_setup_resp (vdev_info=0x60B8562C, vtsp_cdb=0x60FCA114)
```

This means that the VTSP received the CALL SETUP INDICATION message previously sent and has proceeded to process the call.

- vdev_info--Contains the address of the CSM voice data block.
- vtsp_cdb--Contains the address of the VTSP call control block.

The following shows that CSM received a CALL CONNECT message from the VTSP:

```
March 18 04:05:07.596: csm_vtsp_call_connect (vtsp_cdb=0x60FCA114, voice_vdev=0x60B8562C)
```

This indicates that the VTSP received a CONNECT message for the call leg initiated to the Internet side.

- vtsp_cdb--Contains the address of the VTSP call control block.
- voice_vdev--Contains the address of the CSM voice data block.

The following shows that while CSM is in the CSM_IC2_RING state, it receives a SETUP INDICATION RESPONSE from the VTSP. This message is translated into CSM_EVENT_MODEM_OFFHOOK and passed to the CSM central state machine.

```
March 18 04:05:07.596: CSM_PROC_IC2_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 20
```

The following shows that CSM received a CONNECT message from ISDN for the call using the logical DSP channel on slot 1 and port 20:

```
March 18 04:05:07.616: EVENT_FROM_ISDN:(00CF): DEV_CONNECTED at slot 1 and port 20
```

The following shows that CSM translated the CONNECT event from ISDN into the CSM_EVENT_ISDN_CONNECTED message, which is then passed to the CSM central state machine:

```
March 18 04:05:07.616: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 20
```

The following shows that CSM received a CALL SETUP REQUEST from the VTSP:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request (vtsp_cdb=0x60FCFA20, vtsp_sdb=0x60DFB608)
```

This represents a request to make an outgoing call to the PSTN.

- vtsp_cdb--Contains the address of the VTSP call control block.
- vtsp_sdb--Contains the address of the signalling data block for the signalling interface to be used to send the outgoing call.

The following shows that the physical DSP channel has been allocated for this outgoing call:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm slot 1, dspm 5, dsp 4, dsp_channel 1
```

The following shows the on-chip and backplane TDM channel assigned to this DSP channel:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm stream 5, channel 25, bank 0, bp_channel
 27
```

In this sample output, tdm stream 5, channel 25, bank 0, bp_channel 27 indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 25 gives the on-chip TDM channel mapped

to the DSP; bank 0, bp_channel 27 means that the backplane stream 0 and backplane channel 1 are assigned to this DSP.

The following shows the calling number and the called number for this call.

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: calling number: 10001, called number:
30001
```

The following shows that the CALL SETUP REQUEST from the VTSP has been translated into the ' CSM_EVENT_MODEM_OFFHOOK message and is passed to the CSM central state machine:

```
May 16 12:22:27.580: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 54
```

The logical DSP channel number for the DSP (slot 1, port 54) is now displayed, which maps to the physical DSP channel slot 1, dspm 5, dsp 4, dsp_channel 1.

The following shows that CSM collected all the digits for dialing out:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS at slot 1,
port 54
```

For PRI and for applications that do not require digit collection of outdialing digits (for example, voice calls), the intermediate digit collection states are omitted and the CSM state machine moves to this state directly, pretending that the digit collection has been done.

The following shows an information message:

```
March 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num: (30001) at slot
1, port 54
```

The following shows that CSM attempts to find a free signalling D channel to direct the outgoing call:

```
March 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
March 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0,
dchan_idb=0x60E8ACF0)
March 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
March 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0,
dchan_idb=0x60D7ACB0)
```

In the case of voice calls, the free signaling D channel must match the voice interface specified inside the signalling data block (vtsp_sdb) passed from the VTSP.

The following shows that CSM has received an event from ISDN:

```
March 16 12:22:27.624: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
bchan=0x1E, event=0x3, cause=0x0
```

In this sample output:

- dchan_idb--Indicates the address of the hardware IDB for the D channel
- call_id--Indicates the call id assigned by ISDN
- bchan--Indicates the number of the B channel assigned for this call
- cause--Indicates the ISDN event cause

The following shows that CSM has received a CALL PROCEEDING message from ISDN.

```
March 16 12:22:27.624: EVENT_FROM_ISDN:(A121): DEV_CALL_PROC at slot 1 and port 54
```

The following shows that the CALL PROCEEDING event received from ISDN has been interpreted as a CSM_EVENT_ISDN_BCHAN_ASSIGNED message:

```
March 16 12:22:27.624: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port
 54
```

ISDN has assigned a B channel for this outgoing call. This B channel must be on the same PRI span as the signalling D channel allocated previously.

The following shows that the csm_vtsp_setup_for_oc function is called:

```
March 16 12:22:27.624: csm_vtsp_setup_for_oc (voice_vdev=0x60B8562C)
```

This is invoked when an outgoing call initiated by the VTSP receives a response from the ISDN stack.

The following shows that ISDN has sent a CONNECT message to CSM indicating that the call leg to the PSTN side has been established:

```
March 16 12:22:28.084: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
   bchan=0x1E, event=0x4, cause=0x0
March 16 12:22:28.084: EVENT_FROM_ISDN:(A121): DEV_CONNECTED at slot 1 and port 54
```

The following shows that while CSM is in the OC5_WAIT_FOR_CARRIER state, it has received the 'CONNECT' message from ISDN and has translated it into the CSM_EVENT_ISDN_CONNECTED message, which is passed to the CSM central state machine:

```
March 16 12:22:28.084: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 54
```

The following shows that the function vtsp_confirm_oc() has been called:

```
March 16 12:22:28.084: vtsp_confirm_oc : (voice_vdev= 0x60B8562C)
```

This is invoked after CSM received the CONNECT message from ISDN. CSM sends a confirmation of the CONNECT to the VTSP.

debug csm tgrm

To view Call Switching Module (CSM) trunk group resource manager information, use the **debugcsmtgrm**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug csm tgrm

no debug csm tgrm

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Usage Guidelines

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

Examples

The following is sample output from the **debug** csm tgrm command. The output shows that the call type is voice, the direction is incoming, and the call is accepted by the CSM.

```
Router# debug csm tgrm

Router#
00:02:25:CSM-TGRM:csm_rx_cas_event_from_neat(EVENT_DIAL_IN) - c(T1 7/1:1:3) call_type=VOICE,
 dir=INCOMING
Router#
00:02:30:CSM-TGRM:csm_proc_ic3_wait_for_res_resp() c(T1 7/1:1:3) VOICE <ACCEPTED !!>
```

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

Table 14: debug csm tgrm Field Descriptions

Field	Description
call_type	Type of call: VOICE or MODEM.
dir	Direction of the call: INCOMING or OUTGOING.

debug csm voice

To turn on debugging for all Call Switching Module (CSM) Voice over IP (VoIP) calls, use the **debugcsmvoice**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug csm voice [*slot*| *dspm*| *dsp*| *dsp-channel*]

no debug csm voice [*slot*| *dspm*| *dsp*| *dsp-channel*]

Syntax Description

slot | *dspm* | *dsp* | *dsp-channel*	(Optional) Identifies the location of a particular digital signal processor (DSP) channel.

Command Modes

Privileged EXEC

Usage Guidelines

The **debugcsmvoice**command turns on debugging for all CSM VoIP calls. If this command has no keyword specified, then debugging is enabled for all voice calls.

Note The **debugcsmvoice** command does not display any information if you try to debug isdn voice calls. To view debugging information about isdn calls, use the **debugcdapi** command.

The **nodebugcmsvoice** command turns off debugging information for all voice calls.

If the *slot*|*dspm*|*dsp*|*dsp-channel* argument is specified, then (if the specified DSP channel is engaged in a CSM call) CSM call-related debugging information will be turned on for this channel. The **no**form of this command turns off debugging for that particular channel.

Examples

The following examples show sample output from the **debugcsmvoice** command. The following shows that CSM has received an event from ISDN.

```
Router# debug csm voice
Oct 18 04:05:07.052: EVENT_FROM_ISDN::dchan_idb=0x60D7B6B8, call_id=0xCF, ces=0x1
bchan=0x0, event=0x1, cause=0x0
```

In this example, terms are explained as follows:

- dchan_idb--Indicates the address of the hardware interface description block (IDB) for the D channel
- call_id--Indicates the call ID assigned by ISDN
- bchan--Indicates the number of the B channel assigned for this call
- cause--Indicates the ISDN event cause

The following shows that CSM has allocated the CSM voice control block for the DSP device on slot 1 port 10 for this call.

```
Oct 18 04:05:07.052: VDEV_ALLOCATE: slot 1 and port 10 is allocated.
```

This AS5300 access server might not be actually used to handle this call. CSM must first allocate the CSM voice control block to initiate the state machine. After the voice control block has been allocated, CSM obtains from the DSP Resource Manager the actual DSP channel that will be used for the call. At that point, CSM will switch to the actual logical port number. The slot number refers to the physical slot on the AS5300 access server. The port number is the logical DSP number interpreted as listed in the table below.

Table 15: Logical DSP Numbers

Logical Port Number	Physical DSP Channel
Port 0	DSPRM 1, DSP 1, DSP channel 1
Port 1	DSPRM 1, DSP 1, DSP channel 2
Port 2	DSPRM 1, DSP 2, DSP channel 1
Port 3	DSPRM 1, DSP 2, DSP channel 2
Port 4	DSPRM 1, DSP 3, DSP channel 1
Port 5	DSPRM 1, DSP 3, DSP channel 2
Port 6	DSPRM 1, DSP 4, DSP channel 1
Port 7	DSPRM 1, DSP 4, DSP channel 2
Port 8	DSPRM 1, DSP 5, DSP channel 1
Port 9	DSPRM 1, DSP 5, DSP channel 2
Port 10	DSPRM 1, DSP 6, DSP channel 1
Port 11	DSPRM 1, DSP 6, DSP channel 2
Port 12	DSPRM 2, DSP 1, DSP channel 1
Port 13	DSPRM 2, DSP 1, DSP channel 2
Port 14	DSPRM 2, DSP 2, DSP channel 1
Port 15	DSPRM 2, DSP 2, DSP channel 2
Port 16	DSPRM 2, DSP 3, DSP channel 1
Port 17	DSPRM 2, DSP 3, DSP channel 2
Port 18	DSPRM 2, DSP 4, DSP channel 1

Logical Port Number	Physical DSP Channel
Port 19	DSPRM 2, DSP 4, DSP channel 2
Port 20	DSPRM 2, DSP 5, DSP channel 1
Port 21	DSPRM 2, DSP 5, DSP channel 2
Port 22	DSPRM 2, DSP 6, DSP channel 1
Port 23	DSPRM 2, DSP 6, DSP channel 2
Port 48	DSPRM 5, DSP 1, DSP channel 1
Port 49	DSPRM 5, DSP 1, DSP channel 2
Port 50	DSPRM 5, DSP 2, DSP channel 1
Port 51	DSPRM 5, DSP 2, DSP channel 2
Port 52	DSPRM 5, DSP 3, DSP channel 1
Port 53	DSPRM 5, DSP 3, DSP channel 2
Port 54	DSPRM 5, DSP 4, DSP channel 1
Port 55	DSPRM 5, DSP 4, DSP channel 2
Port 56	DSPRM 5, DSP 5, DSP channel 1
Port 57	DSPRM 5, DSP 5, DSP channel 2
Port 58	DSPRM 5, DSP 6, DSP channel 1
Port 59	DSPRM 5, DSP 6, DSP channel 2

The following shows that the function csm_vtsp_init_tdm() has been called with a voice control block of address 0x60B8562C. This function will be called only when the call is treated as a voice call.

```
Oct 18 04:05:07.052: csm_vtsp_init_tdm (voice_vdev=0x60B8562C)
```

The following shows that CSM has obtained a DSP channel from the DSP Resource Manager:

```
Oct 18 04:05:07.052: csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm slot 1, dspm 2, dsp 5,
dsp_channel 1csm_vtsp_init_tdm: dsprm_tdm_allocate: tdm stream 5, channel 9, bank 0,
bp_channel 10
```

The DSP channel has the following initialized TDM channel information:

- TDM slot 1, dspm 2, dsp 5, dsp_channel 1--Indicates the physical DSP channel that will be used for this call.

- TDM stream 5, channel 9, bank 0, bp_channel 10--Indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 9 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 10 means that the backplane stream 0 and backplane channel #1 are assigned to this DSP.

The following shows that CSM has received an incoming call event from ISDN:

```
Oct 18 04:05:07.052: EVENT_FROM_ISDN:(00CF): DEV_INCALL at slot 1 and port 20
```

Slot 1, port 20 means the logical DSP channel 20 (mapped to DSPRM 2, DSP 5, DSP channel 1).

The following shows that the DEV_INCALL message has been translated into a CSM_EVENT_ISDN_CALL message:

```
Oct 18 04:05:07.052: CSM_PROC_IDLE: CSM_EVENT_ISDN_CALL at slot 1, port 20
```

This message is passed to the CSM central state machine while it is in the CSM_IDLE state and is in the CSM_PROC_IDLE procedure. The logical DSP channel port 20 on slot 1 is used to handle this call.

The following shows that CSM has invoked the vtsp_ic_notify() function with a CSM voice call control block 0x60B8562C.

```
Oct 18 04:05:07.052: vtsp_ic_notify : (voice_vdev= 0x60B8562C)
```

Inside this function, CSM will send a SETUP INDICATION message to the VTSP. This function will be invoked only if the call is a voice call.

The following shows that CSM has received a SETUP INDICATION RESPONSE message from the VTSP as an acknowledgment.

```
Oct 18 04:05:07.056: csm_vtsp_call_setup_resp (vdev_info=0x60B8562C, vtsp_cdb=0x60FCA114)
```

This means that the VTSP has received the CALL SETUP INDICATION message previously sent and has proceeded to process the call.

- vdev_info--Contains the address of the CSM voice data block.
- vtsp_cdb--Contains the address of the VTSP call control block.

The following shows that CSM has received a CALL CONNECT message from the VTSP:

```
Oct 18 04:05:07.596: csm_vtsp_call_connect (vtsp_cdb=0x60FCA114, voice_vdev=0x60B8562C)
```

This indicates that the VTSP has received a CONNECT message for the call leg initiated to the Internet side.

- vtsp_cdb--Contains the address of the VTSP call control block.
- voice_vdev--Contains the address of the CSM voice data block.

The following shows that while CSM is in the CSM_IC2_RING state, it receives a SETUP INDICATION RESPONSE from the VTSP. This message is translated into CSM_EVENT_MODEM_OFFHOOK and passed to the CSM central state machine.

```
Oct 18 04:05:07.596: CSM_PROC_IC2_RING: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 20
```

The following shows that CSM has received a CONNECT message from ISDN for the call using the logical DSP channel on slot 1 and port 20:

```
Oct 18 04:05:07.616: EVENT_FROM_ISDN:(00CF): DEV_CONNECTED at slot 1 and port 20
```

The following shows that CSM has translated the CONNECT event from ISDN into the CSM_EVENT_ISDN_CONNECTED message, which is then passed to the CSM central state machine:

```
Oct 18 04:05:07.616: CSM_PROC_IC4_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 20
```

The following shows that CSM has received a CALL SETUP REQUEST from the VTSP:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request (vtsp_cdb=0x60FCFA20, vtsp_sdb=0x60DFB608)
```

This represents a request to make an outgoing call to the PSTN.

- vtsp_cdb--Contains the address of the VTSP call control block.
- vtsp_sdb--Contains the address of the signalling data block for the signalling interface to be used to send the outgoing call.

The following shows that the physical DSP channel has been allocated for this outgoing call:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm slot 1, dspm 5, dsp 4, dsp_channel 1
```

The following shows the on-chip and backplane TDM channel assigned to this DSP channel:

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: tdm stream 5, channel 25, bank 0, bp_channel
 27
```

In this sample output, tdm stream 5, channel 25, bank 0, bp_channel 27 indicates the on-chip and backplane TDM channel assigned to this DSP channel. Stream 5, channel 25 gives the on-chip TDM channel mapped to the DSP; bank 0, bp_channel 27 means that the backplane stream 0 and backplane channel 1 are assigned to this DSP.

The following shows the calling number and the called number for this call.

```
May 16 12:22:27.580: csm_vtsp_call_setup_request: calling number: 10001, called number:
30001
```

The following shows that the CALL SETUP REQUEST from the VTSP has been translated into the ' CSM_EVENT_MODEM_OFFHOOK message and is passed to the CSM central state machine:

```
May 16 12:22:27.580: CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 54
```

The logical DSP channel number for the DSP (slot 1, port 54) is now displayed, which maps to the physical DSP channel slot 1, dspm 5, dsp 4, dsp_channel 1.

The following shows that CSM has collected all the digits for dialing out:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: CSM_EVENT_GET_ALL_DIGITS at slot 1,
port 54
```

For PRI and for applications that do not require digit collection of outdialing digits (for example, voice calls), the intermediate digit collection states are omitted and the CSM state machine moves to this state directly, pretending that the digit collection has been done.

The following shows an information message:

```
May 16 12:22:27.580: CSM_PROC_OC3_COLLECT_ALL_DIGIT: called party num: (30001) at slot 1,
port 54
```

The following shows that CSM attempts to find a free signalling D channel to direct the outgoing call:

```
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0,
dchan_idb=0x60E8ACF0)
May 16 12:22:27.580: csm_vtsp_check_dchan (voice_vdev=0x60B8562C)
May 16 12:22:27.580: csm_vtsp_check_dchan (vtsp requested dchan=0x60D7ACB0,
dchan_idb=0x60D7ACB0)
```

In the case of voice calls, the free signaling D channel must match the voice interface specified inside the signalling data block (vtsp_sdb) passed from the VTSP.

The following shows that CSM has received an event from ISDN:

```
May 16 12:22:27.624: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
bchan=0x1E, event=0x3, cause=0x0
```

In this sample output:

- dchan_idb--indicates the address of the hardware IDB for the D channel
- call_id--Indicates the call id assigned by ISDN
- bchan--Indicates the number of the B channel assigned for this call
- cause--Indicates the ISDN event cause

The following shows that CSM has received a CALL PROCEEDING message from ISDN.

```
May 16 12:22:27.624: EVENT_FROM_ISDN:(A121): DEV_CALL_PROC at slot 1 and port 54
```

The following shows that the CALL PROCEEDING event received from ISDN has been interpreted as a CSM_EVENT_ISDN_BCHAN_ASSIGNED message:

```
*May 16 12:22:27.624: CSM_PROC_OC4_DIALING: CSM_EVENT_ISDN_BCHAN_ASSIGNED at slot 1, port
54
```

ISDN has assigned a B channel for this outgoing call. This B channel must be on the same PRI span as the signalling D channel allocated previously.

The following shows that the csm_vtsp_setup_for_oc function is called:

```
May 16 12:22:27.624: csm_vtsp_setup_for_oc (voice_vdev=0x60B8562C)
```

This is invoked when an outgoing call initiated by the VTSP receives a response from the ISDN stack.

The following shows that ISDN has sent a CONNECT message to CSM indicating that the call leg to the PSTN side has been established:

```
May 16 12:22:28.084: EVENT_FROM_ISDN::dchan_idb=0x60D7ACB0, call_id=0xA121, ces=0x1
   bchan=0x1E, event=0x4, cause=0x0
May 16 12:22:28.084: EVENT_FROM_ISDN:(A121): DEV_CONNECTED at slot 1 and port 54
```

The following shows that while CSM is in the OC5_WAIT_FOR_CARRIER state, it has received the 'CONNECT' message from ISDN and has translated it into the CSM_EVENT_ISDN_CONNECTED message, which is passed to the CSM central state machine:

```
May 16 12:22:28.084: CSM_PROC_OC5_WAIT_FOR_CARRIER: CSM_EVENT_ISDN_CONNECTED at slot 1,
port 54
```

The following shows that the function vtsp_confirm_oc() has been called:

```
May 16 12:22:28.084: vtsp_confirm_oc : (voice_vdev= 0x60B8562C)
```

This is invoked after CSM received the CONNECT message from ISDN. CSM sends a confirmation of the CONNECT to the VTSP.

debug ctl-client

To collect debug information about the CTL client, use the **debugctl-client** command in privileged EXEC configuration mode. To disable collection of debug information, use the **no** form of this command.

debug ctl-client

no debug ctl-client

Syntax Description

This command has no arguments or keywords.

Command Default

Collection of CTL client debug information is disabled.

Command Modes

Privileged EXEC

Command History

Cisco IOS Release	Modification
12.4(4)XC	This command was introduced.
12.4(9)T	This command was integrated into Cisco IOS Release 12.4(9)T.

Usage Guidelines

This command is used with Cisco Unified CME phone authentication.

Examples

The following example shows debug messages for the CTL client:

```
Router# debug ctl-client

001954: .Jul 21 18:23:02.136: ctl_client_create_ctlfile:
001955: .Jul 21 18:23:02.272: create_ctl_record: Function 0 Trustpoint cisco1
001956: .Jul 21 18:23:02.276: create_ctl_record: record added for function 0
001957: .Jul 21 18:23:02.276: create_ctl_record: Function 0 Trustpoint sast2
001958: .Jul 21 18:23:02.280: create_ctl_record: record added for function 0
001959: .Jul 21 18:23:02.280: create_ctl_record: Function 1 Trustpoint cisco1
001960: .Jul 21 18:23:02.284: create_ctl_record: record added for function 1
001961: .Jul 21 18:23:02.284: create_ctl_record: Function 3 Trustpoint cisco1
001962: .Jul 21 18:23:02.288: create_ctl_record: record added for function 3
001963: .Jul 21 18:23:02.288: create_ctl_record: Function 4 Trustpoint cisco1
001964: .Jul 21 18:23:02.292: create_ctl_record: record added for function 4
001965: .Jul 21 18:23:02.424: ctl_client_create_ctlfile: Signature length 128
001966: .Jul 21 18:23:02.640: CTL File Created Successfully
```

debug ctunnel

To display debugging messages for the IP over a Connectionless Network Service (CLNS) Tunnel feature, use the **debugctunnel** command in privileged EXEC mode. To disable debugging messages, use the **no** form of this command.

debug ctunnel

no debug ctunnel

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(5)	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

As packets are sent over the virtual interface, the following type of output will appear on the console when the **debugctunnel** command is used:

```
Router# debug ctunnel

4d21h: CTunnel1: IPCLNP encapsulated 49.0001.1111.1111.1111.00->49.0001.2222.2222.2222.00
(linktype=7, len=89)
```

debug custom-queue

To enable custom queueing output, use the **debugcustom-queue** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug custom-queue

no debug custom-queue

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Examples

The following is an example of enabling custom queueing output:

```
Router# debug custom-queue
Custom output queueing debugging is on
```

The following is sample output from the debug custom-queue command:

```
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 2
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 2 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
00:27:38: CQ: Serial0 output (Pk size/Q: 232/1) Q # was 1 now 1
```

Related Commands

Command	Description
debug priority	Enables priority queueing output.

debug cwmp

To debug the TR-069 Agent, use the **debugcwmp** command in privileged EXEC mode. To disable, use the **no** form of this command.

debug cwmp {**all**| **error**| **profile**| **trace**}

no debug cwmp {**all**| **error**| **profile**| **trace**}

Syntax Description

all	Specifies all debug messages.
error	Specifies error messages.
profile	Specifies the error and trace messages in the Cisco WAN Management Protocol (CWMP) profiles.
trace	Specifies trace message.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(20)T	This command was introduced.

Examples

The following is sample output from the **debugcwmperror** command:

```
Device# debug cwmp error
CWMP ERROR: cwmp_session_response_data -> HTTPC Response failed with httpc error
```

The following is sample trace message output from the **debugcwmpprofile** command:

```
Device# debug cwmp profile
CWMP PROFILE: cwmp_generate_connection_request_URL -> ConnectionRequestURL =
http://172.27.116.170/00000C/FTX1101A1XH/cwmp
```

The following is sample error message output from the **debugcwmpprofile** command:

```
Device# debug cwmp profile
CWMP PROFILE ERROR: validate_dhcp_pool_min_max_address -> Missing
InternetGatewayDevice.LANDevice.5.LANHostConfigManagement.MaxAddress
```

The following is sample output from the **debugcwmptrace** command:

```
Device# debug cwmp trace
CWMP: cwmp_process -> CWMP Engine started
```

debug dampening

To display debug trace information messages for interface dampening, use the **debugdampening** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dampening [**all**| **interface**]

no debug dampening [**all**| **interface**]

Syntax Description

all	(Optional) Enables trace debugging for all dampening features.
interface	(Optional) Enables trace debugging for IP event dampening.

Command Default

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(22)S	This command was introduced.
12.2(13)T	This command was integrated into Cisco IOS Release 12.2(13)T.
12.2(14)S	This command was integrated into Cisco IOS Release 12.2(14)S.
12.2(18)SXD	This command was integrated into Cisco IOS Release 12.2(18)SXD.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.
12.2(31)SB2	This command was integrated into Cisco IOS Release 12.2(31)SB2.

Examples

The following sample output is similar to the output that will be displayed when the **debugdampening** command is entered with the **interface** keyword. The sample output shows the following information:

- Ethernet interface 1/1 is configured with the IP Event Dampening feature. The half-life period is set to 30 seconds, the reuse threshold to 500, the suppress threshold to 1000, and the restart penalty to 90.
- The **shutdown** command and then the **noshutdown** command was entered on Ethernet interface 1/1. The interface was suppressed and then reused by the IP Event Dampening feature.

```
Router# debug dampening interface
```

```
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1):charge penalty 1000, new accum. penalty 1000, flap count 1
00:07:17:EvD(Ethernet1/1):accum. penalty 1000, now suppressed with a reuse intervals of 30
00:07:17:IF-EvD(Ethernet1/1):update CLNS Routing state to DOWN, interface is suppressed
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from DOWN to DOWN
00:07:17:IF-EvD(Ethernet1/1):update IP Routing state to DOWN, interface is suppressed
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1):accum. penalty decayed to 1000 after 0 second(s)
00:07:17:EvD(Ethernet1/1):charge penalty 1000, new accum. penalty 2000, flap count 2
00:07:17:EvD(Ethernet1/1):accum. penalty 2000, now suppressed with a reuse intervals of 60
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to DOWN
00:07:17:EvD(Ethernet1/1):accum. penalty decayed to 2000 after 0 second(s)
00:07:17:EvD(Ethernet1/1):charge penalty 1000, new accum. penalty 3000, flap count 3
00:07:17:EvD(Ethernet1/1):accum. penalty 3000, now suppressed with a reuse intervals of 78
00:07:17:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from DOWN to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:17:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:20:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to UP
00:07:20:IF-EvD(Ethernet1/1):IP Routing reports state transition from UP to UP
00:07:47:IF-EvD:unsuppress interfaces
00:08:36:IF-EvD:unsuppress interfaces
00:08:36:EvD(Ethernet1/1):accum. penalty decayed to 483 after 79 second(s)
00:08:36:EvD(Ethernet1/1):accum. penalty 483, now unsuppressed
00:08:36:IF-EvD(Ethernet1/1):update IP Routing state to UP, interface is not suppressed
00:08:36:IF-EvD(Ethernet1/1):update CLNS Routing state to UP, interface is not suppressed
00:08:36:IF-EvD(Ethernet1/1):CLNS Routing reports state transition from UP to UP
```

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

Table 16: debug dampening Field Descriptions

Field	Description
... Routing reports state transition from UP to DOWN	Displays the status of the specified interface from the perspective of the specified protocol. Interface state changes are displayed. The interface is specified within parentheses. The protocol is specified at the beginning of the message.
charge penalty 1000, new accum. penalty 1000, flap count 1	Displays the penalty assigned to the flapping interface and amount of penalty that is added to the accumulated penalty. The interface flap count is also displayed.
accum. penalty 1000, now suppressed with a reuse intervals of 30	Displays the status of the interface, accumulated penalty, and configured reuse threshold.
update CLNS Routing state to DOWN, interface is suppressed	Displays the status of the specified interface. Interface state changes and suppression status are displayed.
accum. penalty decayed to 1000 after 0 second(s)	Displays the decay rate of the accumulated penalty.
unsuppress interfaces	Indicates that dampened interfaces have been unsuppressed.

debug data-store

To display persistant storage device (PSD)-related debugging messages for the gateway GPRS support node (GGSN), use the **debugdata-store**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug data-store

no debug data-store

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)YU	This command was introduced.
12.4(2)XB	This command was integrated into Cisco IOS Release 12.4(2)XB.
12.4(15)T	This command was integrated into Cisco IOS Release 12.4(15)T.

Usage Guidelines

This command displays PSD-related debugging messages for the GGSN.

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

Examples

The following example configures a debugging session to check PSD-related parameters:

```
Router# debug data-store
```

debug data-store detail

To display extended details for persistent storage device (PSD)-related debugging information, use the **debugdata-storedetail**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug data-store detail

no debug data-store detail

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)YU	This command was introduced.
12.4(2)XB	This command was integrated into Cisco IOS Release 12.4(2)XB.
12.4(15)T	This command was integrated into Cisco IOS Release 12.4(15)T.

Usage Guidelines

This command displays PSD-related debugging messages for the GGSN.

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

Examples

The following example configures a detailed PSD-related debugging session:

```
Router# debug data-store details
```

Related Commands

Command	Description
auto-retrieve	Configures the GGSN to automatically initiate a retrieval of G-CDRs from PSDs defined in a PSD server group.
clear data-store statistics	Clears PSD-related statistics.
show data-store	Displays the status of the PSD client and PSD server-related information.
show data-store statistics	Displays statistics related to the PSD client.

debug dbconn all

To turn on all debug flags for Database Connection, use the **debugdbconnall**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dbconn all

no debug dbconn all

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled for Database Connection.

Command Modes

Privileged EXEC

Usage Guidelines

The **debugdbconnall** command displays debug output for Advanced Program-to-Program Communication (APPC), Database Connection configuration, Distributed Relational Database Architecture (DRDA), error messages, event traces, and TCP. The Database Connection debug flags are **appc**, **config**, **drda**, **event**, and **tcp**.

Examples

See the sample output provided for the **debug dbconn appc** , **debug dbconn config** , **debug dbconn drda** , **debug dbconn event** , and **debug dbconn tcp** commands.

Related Commands

Command	Description
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn appc

To display Advanced Program-to-Program Communication (APPC)-related trace or error messages, use the **debugdbconnappc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dbconn appc

no debug dbconn appc

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

In a router with stable Database Connection, the alias_cp_name field in the trace message should not be blank. There should be no other APPC error message. You can use Advanced Peer-to-Peer Networking (APPN) debug commands with this debug command to track APPN-related errors.

Examples

The following is sample output from the **debugdbconnappc**command. In a normal situation, only the following message is displayed:

```
DBCONN-APPC: alias_cp_name is "ASH"
```

The following error messages are displayed if there is a network configuration error or other APPN-related problem:

```
DBCONN-APPC-612C2B28: APPC error: opcode 0x1, primary_rc 0x0003,
secondary_rc 0x00000004
DBCONN-APPC-612C2B28: Verb block =
DBCONN-APPC-612C2B28:   0001 0200 0003 0000  0000 0004 0020 100C
DBCONN-APPC-612C2B28:   610A 828B 0000 0000  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   0000 0000 8014 0003  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   D3E4 F6F2 E2E3 C1D9  C4C2 F240 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  0200 0000 0000 0000
DBCONN-APPC-612C2B28:   0000 0000 D4C5 D9D9  C9C5 4040 4040 D7C5
DBCONN-APPC-612C2B28:   E3C5 D940 4040 4040  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   00E2 E3C1 D9E6 4BE3  D6D9 C3C8 4040 4040
DBCONN-APPC-612C2B28:   4040 0000 0000 0000  0000 0000
DBCONN-APPC-612C2B28: ALLOCATE verb block =
DBCONN-APPC-612C2B28:   0001 0200 0003 0000  0000 0004 0020 100C
DBCONN-APPC-612C2B28:   610A 828B 0000 0000  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   0000 0000 8014 0003  0000 0000 0000 0000
DBCONN-APPC-612C2B28:   D3E4 F6F2 E2E3 C1D9  C4C2 F240 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  4040 4040 4040 4040
DBCONN-APPC-612C2B28:   4040 4040 4040 4040  0200 0000 0000 0000
```

You can use the **debugappn**command to obtain more information.

The following message is displayed if a database connection is manually cleared and an outstanding APPC verb is pending:

```
DBCONN-APPC-%612C2B28: Canceling pending APPC verb 0x1
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn event	Displays trace or error messages for Database Connection events.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn config

To display trace or error messages for Database Connection configuration and control blocks, use the **debugdbconnconfig**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dbconn config

no debug dbconn config

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

Most of the messages for Database Connection and control blocks do not report any errors. If a connection is inactive and cannot be cleared, use this command with the **debugdbconnappc,debugdbconntcp**, and**debugappn** commands to locate the problem. The alias_cp_name field must match the configured APPN cpname.

Examples

The following is sample output from the **debugdbconnconfig**command:

```
Router# debug dbconn config
DBCONN-CONFIG: alias_cp_name is "ASH     "
DBCONN-CONFIG: connection 612BDAAC matching server on 198.147.235.5:0 with
rdbname=STELLA
DBCONN-CONFIG: APPN shutdown; clearing connection 1234abcd
DBCONN-CONFIG: created server 612C2720
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: server 612C2720 (listen 60F72E94) is active
DBCONN-CONFIG: new connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 accepts connection 612BDAAC
DBCONN-CONFIG: server 60F74614 takes connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 releases connection 612BDAAC
DBCONN-CONFIG: server 60F74614 releases connection 612BDAAC
DBCONN-CONFIG: deleting connection 612BDAAC
DBCONN-CONFIG: listen 60F72E94 abandons connection 612BDAAC
DBCONN-CONFIG: server 612C2720 abandons connection 612BDAAC
DBCONN-CONFIG: deleting server 612C2720
DBCONN-CONFIG: daemon 60381738 takes zombie connection 612BDAAC
DBCONN-CONFIG: daemon 60381738  releases zombie connection 612BDAAC
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn drda	Displays error messages and stream traces for DRDA.

Command	Description
debug dbconn event	Displays trace or error messages for Database Connection events.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn drda

To display error messages and stream traces for Distributed Relational Database Architecture (DRDA), use the **debugdbconndrda**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dbconn drda

no debug dbconn drda

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled for the dbconn subsystem.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following is sample output from the **debugdbconndrda**command:

```
Router# debug dbconn drda
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: DSS X'006CD0410001', length 108, in chain,
REQDSS, correlator 1
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'00661041', length 98, code point X'1041'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0020115E' in COLLECTION X'1041', length
 28, code point X'115E'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'000C116D' in COLLECTION X'1041', length
 8, code point X'116D'
*Jun 30 16:09:32.363: DBCONN-DRDA-62008300: OBJECT X'0013115A' in COLLECTION X'1041', length
 15, code point X'115A' (skipping...)
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.

Command	Description
debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
debug dbconn tcp	Displays error messages and traces for TCP.

debug dbconn event

To display trace or error messages for Cisco Transaction Connection (CTRC) events related to DATABASE2 (DB2) communications, use the **debugdbconnevent**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dbconn event

no debug dbconn event

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled for the dbconn subsystem.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following examples display output from the **debugdbconnevent**command in a variety of situations. A normal trace for the **debugdbconnevent** displays as follows:

```
Router# debug dbconn event
DBCONN-EVENT: Dispatch to 60FD6C00, from 0, msg 60F754CC, msgid 6468 'dh',
buffer 0.
DBCONN-EVENT: [*] Post to 61134240(cn), from 60EC5470(tc), msg 611419E4,
msgid 0x6372 'cr', buffer 612BF68C.
DBCONN-EVENT: Flush events called for pto 61182742, pfrom 61239837.
DBCONN-EVENT: Event discarded: to 61182742 (cn), from 61239837(ap), msg
61339273, msgid 0x6372 'cr' buffer 0.
DBCONN-EVENT: == Send to 1234abcd, from 22938acd, msg 72618394, msgid
0x6372 'cr', buffer 0.
```

If the following messages are displayed, contact Cisco technical support personnel:

```
DBCONN-TCPFSM-1234abcd: Cannot occur in state 2 on input 6363 ('cc')
DBCONN-APPCFSM-1234abcd: Cannot occur in state 3 on input 6363 ('cc')
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn appc	Displays APPC-related trace or error messages.

Command	Description
debug dbconn config	Display trace or error messages for Database Connection configuration and control blocks.
debug dbconn drda	Displays error messages and stream traces for DRDA.
debug dbconn tcp	Displays error messages and traces for TCP.
show debugging	Displays the state of each debugging option.

debug dbconn tcp

To display error messages and traces for TCP, use the **debugdbconntcp**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug dbconn tcp

no debug dbconn tcp

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging is not enabled for the dbconn subsystem.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(2)T	This command was introduced.
12.2(33)SRA	This command was integrated into Cisco IOS Release 12.2(33)SRA.

Examples

The following is sample output from the **debugdbconntcp**command:

```
Router# debug dbconn tcp
DBCONN-TCP-63528473: tcpdriver_passive_open returned NULL
DBCONN-TCP-63528473: (no memory) tcp_reset(63829482) returns 4
DBCONN-TCP: tcp_accept(74625348,&error) returns tcb 63829482, error 4
DBCONN-TCP: (no memory) tcp_reset(63829482) returns 4
DBCONN-TCP-63528473: (open) tcp_create returns 63829482, error = 4
DBCONN-TCP-63528473: tcb_connect(63829482,1.2.3.4,2010) returns 4
DBCONN-TCP-63528473: (open error) tcp_reset(63829482) returns 4
DBCONN-TCP-63528473: tcp_create returns 63829482, error = 4
DBCONN-TCP-63528473: tcb_bind(63829482,0.0.0.0,2001) returns 4
DBCONN-TCP-63528473: tcp_listen(63829482,,) returns 4
DBCONN-TCP-63528473: (errors) Calling tcp_close (63829482)
```

Related Commands

Command	Description
debug dbconn all	Turns on all debug flags for Database Connection.
debug dbconn appc	Displays APPC-related trace or error messages.
debug dbconn config	Displays trace or error messages for Database Connection configuration and control blocks.
debug dbconn drda	Displays error messages and stream traces for DRDA.

Command	Description
debug dbconn event	Displays trace or error messages for CTRC events related to DB2 communications.
show debugging	Displays the state of each debugging option.