debug vpdn

To troubleshoot Layer 2 Forwarding (L2F) or Layer 2 Tunnel Protocol (L2TP) virtual private dialup network (VPDN) tunneling events and infrastructure, use the **debug vpdn** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpdn {**error** | **event** [**disconnect**] | **l2tp-sequencing** | **l2x-data** | **l2x-errors** | **l2x-events** | **l2x-packets** | **packet** [**errors**]}

no debug vpdn {**error** | **event** [**disconnect**] | **l2tp-sequencing** | **l2x-data** | **l2x-errors** | **l2x-events** | **l2x-packets** | **packet** [**errors**]}

Syntax Description

error	Displays VPDN errors.
event	Displays VPDN events.
disconnect	(Optional) Displays VPDN disconnect events.
l2tp-sequencing	Displays significant events related to L2TP sequence numbers such as mismatches, resend queue flushes, and drops.
l2x-data	Displays errors that occur in data packets.
l2x-errors	Displays errors that occur in protocol-specific conditions.
l2x-events	Displays events resulting from protocol-specific conditions.
l2x-packets	Displays detailed information about control packets in protocol-specific conditions.
packet	Displays information about VPDN packets.
errors	(Optional) Displays errors that occur in packet processing.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.2	This command was introduced.
12.0(5)T	Support was added for L2TP debugging messages. The **l2tp-sequencing** and **errors** keywords were added. The **l2f-errors**, **l2f-events**, and **l2f-packets** keywords were changed to **l2x-errors**, **l2x-events**, and **l2x-packets**.

Usage Guidelines

Note that the **debug vpdn packet** and **debug vpdn packet detail** commands generate several debug operations per packet. Depending on the L2TP traffic pattern, these commands may cause the CPU load to increase to a high level that impacts performance.

Examples

This section contains the following examples:

- Debugging VPDN Events on a NAS—Normal L2F Operations
- Debugging VPDN Events on the Tunnel Server—Normal L2F Operations

- Debugging VPDN Events on the NAS—Normal L2TP Operations
- Debugging VPDN Events on the Tunnel Server—Normal L2TP Operations
- Debugging Protocol-Specific Events on the NAS—Normal L2F Operations
- Debugging Protocol-Specific Events on the Tunnel Server—Normal L2F Operations
- Debugging Errors on the NAS—L2F Error Conditions
- Debugging L2F Control Packets for Complete Information

Debugging VPDN Events on a NAS—Normal L2F Operations

The network access server (NAS) has the following VPDN configuration:

```
vpdn-group 1
 request-dialin
  protocol l2f
  domain cisco.com
 initiate-to ip 172.17.33.125
username nas1 password nas1
```

The following is sample output from the **debug vpdn event** command on a NAS when an L2F tunnel is brought up and Challenge Handshake Authentication Protocol (CHAP) authentication of the tunnel succeeds:

```
Router# debug vpdn event

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:26:05.537: looking for tunnel -- cisco.com --
*Mar 2 00:26:05.545: Async6 VPN Forwarding...
*Mar 2 00:26:05.545: Async6 VPN Bind interface direction=1
*Mar 2 00:26:05.553: Async6 VPN vpn_forward_user user6@cisco.com is forwarded
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:26:06.289: L2F: Chap authentication succeeded for nas1.
```

The following is sample output from the **debug vpdn event** command on a NAS when the L2F tunnel is brought down normally:

```
Router# debug vpdn event

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:27:18.865: Async6 VPN cleanup
*Mar 2 00:27:18.869: Async6 VPN reset
*Mar 2 00:27:18.873: Async6 VPN Unbind interface
%LINK-3-UPDOWN: Interface Async6, changed state to down
```

Table 203 describes the significant fields shown in the two previous displays. The output describes normal operations when an L2F tunnel is brought up or down on a NAS.

Table 203 debug vpdn event Field Descriptions for the NAS

Field	Description
Asynchronous interface coming up	
%LINK-3-UPDOWN: Interface Async6, changed state to up	Asynchronous interface 6 came up.
looking for tunnel -- cisco.com -- Async6 VPN Forwarding...	Domain name is identified.

Table 203 debug vpdn event Field Descriptions for the NAS (continued)

Field	Description
Async6 VPN Bind interface direction=1	Tunnel is bound to the interface. These are the direction values: • 1—From the NAS to the tunnel server • 2—From the tunnel server to the NAS
Async6 VPN vpn_forward_user user6@cisco.com is forwarded	Tunnel for the specified user and domain name is forwarded.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol is up.
L2F: Chap authentication succeeded for nas1.	Tunnel was authenticated with the tunnel password nas1.
Virtual access interface coming down	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down	Normal operation when the virtual access interface is taken down.
Async6 VPN cleanup Async6 VPN reset Async6 VPN Unbind interface	Normal cleanup operations performed when the line or virtual access interface goes down.

Debugging VPDN Events on the Tunnel Server—Normal L2F Operations

The tunnel server has the following VPDN configuration, which uses nas1 as the tunnel name and the tunnel authentication name. The tunnel authentication name might be entered in a users file on an authentication, authorization, and accounting (AAA) server and used to define authentication requirements for the tunnel.

```
vpdn-group 1
 accept-dialin
  protocol l2f
  virtual-template 1
 terminate-from hostname nas1
```

The following is sample output from the **debug vpdn event** command on the tunnel server when an L2F tunnel is brought up successfully:

```
Router# debug vpdn event

L2F: Chap authentication succeeded for nas1.
Virtual-Access3 VPN Virtual interface created for user6@cisco.com
Virtual-Access3 VPN Set to Async interface
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up
Virtual-Access3 VPN Bind interface direction=2
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up
```

The following is sample output from the **debug vpdn event** command on a tunnel server when an L2F tunnel is brought down normally:

```
Router# debug vpdn event

%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down
Virtual-Access3 VPN cleanup
```

```
Virtual-Access3 VPN reset
Virtual-Access3 VPN Unbind interface
Virtual-Access3 VPN reset
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down
```

Table 204 describes the fields shown in two previous outputs. The output describes normal operations when an L2F tunnel is brought up or down on a tunnel server.

Table 204 debug vpdn event Field Descriptions for the Tunnel Server

Field	Description
Tunnel coming up	
L2F: Chap authentication succeeded for nas1.	PPP CHAP authentication status for the tunnel named nas1.
Virtual-Access3 VPN Virtual interface created for user6@cisco.com	Virtual access interface was set up on the tunnel server for the user user6@cisco.com.
Virtual-Access3 VPN Set to Async interface	Virtual access interface 3 was set to asynchronous for character-by-character transmission.
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0	Virtual template 1 was applied to virtual access interface 3.
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up	Link status is set to up.
Virtual-Access3 VPN Bind interface direction=2	Tunnel is bound to the interface. These are the direction values: • 1—From the NAS to the tunnel server • 2—From the tunnel server to the NAS
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK	PPP link control protocol (LCP) configuration settings (negotiated between the remote client and the NAS) were copied to the tunnel server and acknowledged.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up	Line protocol is up; the line can be used.
Tunnel coming down	
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down	Virtual access interface is coming down.
Virtual-Access3 VPN cleanup Virtual-Access3 VPN reset Virtual-Access3 VPN Unbind interface Virtual-Access3 VPN reset	Router is performing normal cleanup operations when a virtual access interface used for an L2F tunnel comes down.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down	Line protocol is down for virtual access interface 3; the line cannot be used.

Debugging VPDN Events on the NAS—Normal L2TP Operations

The following is sample output from the **debug vpdn event** command on the NAS when an L2TP tunnel is brought up successfully:

```
Router# debug vpdn event

20:19:17: L2TP: I SCCRQ from ts1 tnl 8
20:19:17: L2X: Never heard of ts1
20:19:17: Tnl 7 L2TP: New tunnel created for remote ts1, address 172.21.9.4
20:19:17: Tnl 7 L2TP: Got a challenge in SCCRQ, ts1
20:19:17: Tnl 7 L2TP: Tunnel state change from idle to wait-ctl-reply
20:19:17: Tnl 7 L2TP: Got a Challenge Response in SCCCN from ts1
20:19:17: Tnl 7 L2TP: Tunnel Authentication success
20:19:17: Tnl 7 L2TP: Tunnel state change from wait-ctl-reply to established
20:19:17: Tnl 7 L2TP: SM State established
20:19:17: Tnl/Cl 7/1 L2TP: Session FS enabled
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from idle to wait-for-tunnel
20:19:17: Tnl/Cl 7/1 L2TP: New session created
20:19:17: Tnl/Cl 7/1 L2TP: O ICRP to ts1 8/1
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from wait-for-tunnel to wait-connect
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from wait-connect to established
20:19:17: Vi1 VPDN: Virtual interface created for bum1@cisco.com
20:19:17: Vi1 VPDN: Set to Async interface
20:19:17: Vi1 VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
20:19:18: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
20:19:18: Vi1 VPDN: Bind interface direction=2
20:19:18: Vi1 VPDN: PPP LCP accepting rcv CONFACK
20:19:19: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state
to up
```

Debugging VPDN Events on the Tunnel Server—Normal L2TP Operations

The following is sample output from the **debug vpdn event** command on the tunnel server when an L2TP tunnel is brought up successfully:

```
Router# debug vpdn event

20:47:33: %LINK-3-UPDOWN: Interface Async7, changed state to up
20:47:35: As7 VPDN: Looking for tunnel -- cisco.com --
20:47:35: As7 VPDN: Get tunnel info for cisco.com with NAS nas1, IP 172.21.9.13
20:47:35: As7 VPDN: Forward to address 172.21.9.13
20:47:35: As7 VPDN: Forwarding...
20:47:35: As7 VPDN: Bind interface direction=1
20:47:35: Tnl/Cl 8/1 L2TP: Session FS enabled
20:47:35: Tnl/Cl 8/1 L2TP: Session state change from idle to wait-for-tunnel
20:47:35: As7 8/1 L2TP: Create session
20:47:35: Tnl 8 L2TP: SM State idle
20:47:35: Tnl 8 L2TP: Tunnel state change from idle to wait-ctl-reply
20:47:35: Tnl 8 L2TP: SM State wait-ctl-reply
20:47:35: As7 VPDN: bum1@cisco.com is forwarded
20:47:35: Tnl 8 L2TP: Got a challenge from remote peer, nas1
20:47:35: Tnl 8 L2TP: Got a response from remote peer, nas1
20:47:35: Tnl 8 L2TP: Tunnel Authentication success
20:47:35: Tnl 8 L2TP: Tunnel state change from wait-ctl-reply to established
20:47:35: Tnl 8 L2TP: SM State established
20:47:35: As7 8/1 L2TP: Session state change from wait-for-tunnel to wait-reply
20:47:35: As7 8/1 L2TP: Session state change from wait-reply to established
20:47:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async7, changed state to up
```

Debugging Protocol-Specific Events on the NAS—Normal L2F Operations

The following is sample output from the **debug vpdn l2x-events** command on the NAS when an L2F tunnel is brought up successfully:

```
Router# debug vpdn l2x-events

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:41:17.365: L2F Open UDP socket to 172.21.9.26
*Mar 2 00:41:17.385: L2F_CONF received
*Mar 2 00:41:17.389: L2F Removing resend packet (type 1)
*Mar 2 00:41:17.477: L2F_OPEN received
*Mar 2 00:41:17.489: L2F Removing resend packet (type 2)
*Mar 2 00:41:17.493: L2F building nas2gw_mid0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:41:18.613: L2F_OPEN received
*Mar 2 00:41:18.625: L2F Got a MID management packet
*Mar 2 00:41:18.625: L2F Removing resend packet (type 2)
*Mar 2 00:41:18.629: L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6
```

The following is sample output from the **debug vpdn l2x-events** command on a NAS when an L2F tunnel is brought down normally:

```
Router# debug vpdn l2x-events

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:42:29.213: L2F_CLOSE received
*Mar 2 00:42:29.217: L2F Destroying mid
*Mar 2 00:42:29.217: L2F Removing resend packet (type 3)
*Mar 2 00:42:29.221: L2F Tunnel is going down!
*Mar 2 00:42:29.221: L2F Initiating tunnel shutdown.
*Mar 2 00:42:29.225: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F Got closing for tunnel
*Mar 2 00:42:29.233: L2F Removing resend packet
*Mar 2 00:42:29.233: L2F Closed tunnel structure
%LINK-3-UPDOWN: Interface Async6, changed state to down
*Mar 2 00:42:31.793: L2F Closed tunnel structure
*Mar 2 00:42:31.793: L2F Deleted inactive tunnel
```

Table 205 describes the fields shown in the displays.

Table 205 debug vpdn l2x-events Field Descriptions—NAS

Field	Descriptions
Tunnel coming up	
%LINK-3-UPDOWN: Interface Async6, changed state to up	Asynchronous interface came up normally.
L2F Open UDP socket to 172.21.9.26	L2F opened a User Datagram Protocol (UDP) socket to the tunnel server IP address.
L2F_CONF received	L2F_CONF signal was received. When sent from the tunnel server to the NAS, an L2F_CONF indicates the tunnel server's recognition of the tunnel creation request.

Table 205 debug vpdn l2x-events Field Descriptions—NAS (continued)

Field	Descriptions
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the tunnel server accepted the NAS configuration of an L2F tunnel.
L2F building nas2gw_mid0	L2F is building a tunnel between the NAS and the tunnel server, using the Multiplex ID (MID) MID0.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol came up. Indicates whether the software processes that handle the line protocol regard the interface as usable.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the tunnel server accepted the NAS configuration of an L2F tunnel.
L2F Got a MID management packet	MID management packets are used to communicate between the NAS and the tunnel server.
L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6	L2F synchronized the Client IDs on the NAS and the tunnel server, respectively. A multiplex ID is assigned to identify this connection in the tunnel.
Tunnel coming down	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down	Line protocol came down. Indicates whether the software processes that handle the line protocol regard the interface as usable.
%LINK-5-CHANGED: Interface Async6, changed state to reset	Interface was marked as reset.
L2F_CLOSE received	NAS received a request to close the tunnel.
L2F Destroying mid	Connection identified by the MID is being taken down.
L2F Tunnel is going down!	Advisory message about impending tunnel shutdown.
L2F Initiating tunnel shutdown.	Tunnel shutdown has started.
L2F_CLOSE received	NAS received a request to close the tunnel.
L2F Got closing for tunnel	NAS began tunnel closing operations.
%LINK-3-UPDOWN: Interface Async6, changed state to down	Asynchronous interface was taken down.
L2F Closed tunnel structure	NAS closed the tunnel.
L2F Deleted inactive tunnel	Now-inactivated tunnel was deleted.

Debugging Protocol-Specific Events on the Tunnel Server—Normal L2F Operations

The following is sample output from the **debug vpdn l2x-events** command on a tunnel server when an L2F tunnel is created:

```
Router# debug vpdn l2x-events

L2F_CONF received
L2F Creating new tunnel for nas1
L2F Got a tunnel named nas1, responding
L2F Open UDP socket to 172.21.9.25
L2F_OPEN received
L2F Removing resend packet (type 1)
L2F_OPEN received
L2F Got a MID management packet
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
```

The following is sample output from the **debug vpdn l2x-events** command on a tunnel server when the L2F tunnel is brought down normally:

```
Router# debug vpdn l2x-events

L2F_CLOSE received
L2F Destroying mid
L2F Removing resend packet (type 3)
L2F Tunnel is going down!
L2F Initiating tunnel shutdown.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
L2F_CLOSE received
L2F Got closing for tunnel
L2F Removing resend packet
L2F Removing resend packet
L2F Closed tunnel structure
L2F Closed tunnel structure
L2F Deleted inactive tunnel
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down
```

Table 206 describes the significant fields shown in the displays.

Table 206 debug vpdn l2x-events Field Descriptions—Tunnel Server

Field	Description
Tunnel coming up	
L2F_CONF received	L2F configuration is received from the NAS. When sent from a NAS to a tunnel server, the L2F_CONF is the initial packet in the conversation.
L2F Creating new tunnel for nas1	Tunnel named *nas1* is being created.
L2F Got a tunnel named nas1, responding	Tunnel server is responding.
L2F Open UDP socket to 172.21.9.25	Opening a socket to the NAS IP address.
L2F_OPEN received	L2F_OPEN management message was received, indicating the NAS is opening an L2F tunnel.
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. The two resend packet types have different meanings in different states of the tunnel.

Table 206 debug vpdn l2x-events Field Descriptions—Tunnel Server (continued)

Field	Description
L2F Got a MID management packet	L2F MID management packets are used to communicate between the NAS and the tunnel server.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up	Tunnel server is bringing up virtual access interface 1 for the L2F tunnel.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up	Line protocol is up. The line can be used.
Tunnel coming down	
L2F_CLOSE received	NAS or tunnel server received a request to close the tunnel.
L2F Destroying mid	Connection identified by the MID is being taken down.
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel.
L2F Tunnel is going down! L2F Initiating tunnel shutdown.	Router is performing normal operations when a tunnel is coming down.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down	The virtual access interface is coming down.
L2F_CLOSE received L2F Got closing for tunnel L2F Removing resend packet L2F Removing resend packet L2F Closed tunnel structure L2F Closed tunnel structure L2F Deleted inactive tunnel	Router is performing normal cleanup operations when the tunnel is being brought down.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down	Line protocol is down; virtual access interface 1 cannot be used.

Debugging Errors on the NAS—L2F Error Conditions

The following is sample output from the **debug vpdn errors** command on a NAS when the L2F tunnel is not set up:

```
Router# debug vpdn errors

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down
%LINK-5-CHANGED: Interface Async1, changed state to reset
%LINK-3-UPDOWN: Interface Async1, changed state to down
%LINK-3-UPDOWN: Interface Async1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up
VPDN tunnel management packet failed to authenticate
VPDN tunnel management packet failed to authenticate
```

Table 207 describes the significant fields shown in the display.

Table 207 debug vpdn error Field Descriptions for the NAS

Field	Description
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down	Line protocol on the asynchronous interface went down.
%LINK-5-CHANGED: Interface Async1, changed state to reset	Asynchronous interface 1 was reset.
%LINK-3-UPDOWN: Interface Async1, changed state to down %LINK-3-UPDOWN: Interface Async1, changed state to up	Link from asynchronous interface 1 link went down and then came back up.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up	Line protocol on the asynchronous interface came back up.
VPDN tunnel management packet failed to authenticate	Tunnel authentication failed. This is the most common VPDN error. **Note** Verify the password for the NAS and the tunnel server name. If you store the password on an AAA server, you can use the **debug aaa authentication** command.

The following is sample output from the **debug vpdn l2x-errors** command:

```
Router# debug vpdn l2x-errors

%LINK-3-UPDOWN: Interface Async1, changed state to up
L2F Out of sequence packet 0 (expecting 0)
L2F Tunnel authentication succeeded for cisco.com
 L2F Received a close request for a non-existent mid
 L2F Out of sequence packet 0 (expecting 0)
 L2F packet has bogus1 key 1020868 D248BA0F
L2F packet has bogus1 key 1020868 D248BA0F
```

Table 208 describes the significant fields shown in the display.

Table 208 debug vpdn l2x-errors Field Descriptions

Field	Description
%LINK-3-UPDOWN: Interface Async1, changed state to up	The line protocol on the asynchronous interface came up.
L2F Out of sequence packet 0 (expecting 0)	Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received.
L2F Tunnel authentication succeeded for cisco.com	Tunnel was established from the NAS to the tunnel server, cisco.com.
L2F Received a close request for a non-existent mid	Multiplex ID was not used previously; cannot close the tunnel.
L2F Out of sequence packet 0 (expecting 0)	Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received.

Table 208 debug vpdn l2x-errors Field Descriptions (continued)

Field	Description
L2F packet has bogus1 key 1020868 D248BA0F	Value based on the authentication response given to the peer during tunnel creation. This packet, in which the key does not match the expected value, must be discarded.
L2F packet has bogus1 key 1020868 D248BA0F	Another packet was received with an invalid key value. The packet must be discarded.

Debugging L2F Control Packets for Complete Information

The following is sample output from the **debug vpdn l2x-packets** command on a NAS. This example displays a trace for a **ping** command:

```
Router# debug vpdn l2x-packets

L2F SENDING (17): D0 1 1 10 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 16 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 10 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F SENDING (17): D0 1 1 11 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 17 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 11 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F header flags: 57345 version 57345 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key 1701976070
L2F-IN Output to Async1 (16): FF 3 C0 21 9 F 0 C 0 1D 41 AD FF 11 46 87
L2F-OUT (16): FF 3 C0 21 A F 0 C 0 1A C9 BD FF 11 46 87
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key -2120949344
L2F-OUT (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 3 1 0 0 1 8 0 62 B1
0 0 C A8 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key -2120949344
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key 1701976070
L2F-IN Output to Async1 (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 1 1 0
0 3 0 0 6A B1 0 0 C A8 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
```

Table 209 describes the significant fields shown in the display.

Table 209 debug vpdn l2x-packets Field Descriptions

Field	Description
L2F SENDING (17)	Number of bytes being sent. The first set of "SENDING"..."RECEIVED" lines displays L2F keepalive traffic. The second set displays L2F management data.
L2F header flags:	Version and flags, in decimal.
version 53249	Version.
protocol 1	Protocol for negotiation of the point-to-point link between the NAS and the tunnel server is always 1, indicating L2F management.

Table 209 debug vpdn l2x-packets Field Descriptions (continued)

Field	Description
sequence 16	Sequence numbers start at 0. Each subsequent packet is sent with the next increment of the sequence number. The sequence number is thus a free running counter represented modulo 256. There is a distinct sequence counter for each distinct MID value.
mid 0	Multiplex ID, which identifies a particular connection within the tunnel. Each new connection is assigned a MID currently unused within the tunnel.
cid 4	Client ID used to assist endpoints in demultiplexing tunnels.
length 17	Size in octets of the entire packet, including header, all fields pre-sent, and payload. Length does not reflect the addition of the checksum, if pre-sent.
offset 0	Number of bytes past the L2F header at which the payload data is expected to start. If it is 0, the first byte following the last byte of the L2F header is the first byte of payload data.
key 1701976070	Value based on the authentication response given to the peer during tunnel creation. During the life of a session, the key value serves to resist attacks based on spoofing. If a packet is received in which the key does not match the expected value, the packet must be silently discarded.
L2F RECEIVED (17)	Number of bytes received.
L2F-IN Otput to Async1 (16)	Payload datagram. The data came in to the VPDN code.
L2F-OUT (16):	Payload datagram sent out from the VPDN code to the tunnel.
L2F-OUT (101)	Ping payload datagram. The value 62 in this line is the ping packet size in hexadecimal (98 in decimal). The three lines that follow this line show ping packet data.

Related Commands

Command	Description
debug aaa authentication	Displays information on AAA/TACACS+ authentication.
debug acircuit	Displays events and failures related to attachment circuits.
debug pppoe	Display debugging information for PPPoE sessions.
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established sessions to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.

debug vpm all

Use the **debug vpm all** privileged EXEC command to enable debugging on all VPM areas. The **no** form of this command disables debugging output.

debug vpm all

no debug vpm all

Syntax Description

This command has no arguments or keywords.

Examples

The **debug vpm all** command enables all of the **debug vpm** commands: **debug vpm dsp**, **debug vpm port, debug vpm signal**, and **debug vpm spi**. For more information or sample output, refer to the individual commands in this chapter.

Related Commands

Command	Description
debug vpm dsp	Shows messages from the DSP on the VPM to the router.
debug vpm port	Observes the behavior of the Holst state machine.
debug vpm signal	Collects debug information only for signaling events.
debug vpm spi	Traces how the voice port module SPI interfaces with the call control API.

debug vpm dsp

Use the **debug vpm dsp** privileged EXEC command to show messages from the DSP on the VPM to the router. The **no** form of this command disables debugging output.

debug vpm dsp

no debug vpm dsp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vpm dsp** command shows messages from the DSP on the VPM to the router; this command can be useful if you suspect that the VPM is not functional. It is a simple way to check if the VPM is responding to off-hook indications and to evaluate timing for signaling messages from the interface.

Examples

The following output shows the DSP timestamp and the router timestamp for each event and, for SIG_STATUS, the state value shows the state of the ABCD bits in the signaling message. This sample shows a call coming in on an FXO interface.

The router waits for ringing to terminate before accepting the call. State=0x0 indicates ringing; state 0x4 indicates not ringing:

```
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x0 timestamp=58172 systime=40024
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x4 timestamp=59472 systime=40154
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x4 timestamp=59589 systime=40166
```

The following output shows the digits collected:

```
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=4
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=1
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
```

This shows the disconnect indication and the final call statistics reported by the DSP (which are then populated in the call history table):

```
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0xC timestamp=21214 systime=42882
vcsm_dsp_message: MSG_TX_GET_TX_STAT: num_tx_pkts=1019 num_signaling_pkts=0
num_comfort_noise_pkts=0 transmit_durtation=24150 voice_transmit_duration=20380
fax_transmit_duration=0
```

debug vpm port

To observe the behavior of the Holst state machine, use the **debug vpm port** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug vpm port [*slot-number*| *subunit-number* | *port*]

no debug vpm [*slot-number* | *subunit-number* | *port*]

Syntax Description

slot-number	(Optional) Specifies the slot number in the Cisco router where the voice interface card is installed. Valid entries are from 0 to 3, depending on the router being used and the slot where the voice interface card has been installed.
subunit-number	(Optional) Specifies the subunit on the voice interface card where the voice port is located. Valid entries are 0 or 1.
port	(Optional) Specifies the voice port. Valid entries are 0 or 1.

Command History

Release	Modification
11.3(1)	This command was introduced.

Usage Guidelines

This command is not supported on Cisco 7200 series routers or on the Cisco MC3810.

Use this command to limit the debug output to a particular port. The debug output can be quite voluminous for a single channel. A 12-port box might create problems. Use this debug command with any or all of the other debug modes.

Execution of **no debug vpm all** will turn off all port level debugging. Cisco recommends that you turn off all debugging and then enter the debug commands you are interested in one by one. This process helps to avoid confusion about which ports you are actually debugging.

Examples

The following example shows sample output from the **debug vpm port 1/1/0** command during trunk establishment after the **no shutdown** command has been executed on the voice port:

```
router# debug vpm port 1/1/0
*Mar  1 03:21:39.799: htsp_process_event: [1/1/0, 0.1 , 2]act_down_inserve
*Mar  1 03:21:39.807: htsp_process_event: [1/1/0, 0.0 , 14]
      act_go_trunkhtsp_trunk_createhtsp_trunk_sig_linkfxols_trunk
*Mar  1 03:21:39.807: htsp_process_event: [1/1/0, 1.0 , 1]trunk_offhookfxols_trunk_down
*Mar  1 03:21:39.807: dsp_sig_encap_config: [1/1/0] packet_len=28 channel_id=128
      packet_id=42 transport_protocol=1 playout_delay=100 signaling_mode=0
      t_ssrc=0 r_ssrc=0 t_vpxcc=0 r_vpxcc=0
*Mar  1 03:21:39.811: dsp_set_sig_state: [1/1/0] packet_len=12
      channel_id=128 packet_id=39 state=0xC timestamp=0x0
*Mar  1 03:21:39.811: trunk_offhook: Trunk Retry Timer Enabled
*Mar  1 03:22:13.095: htsp_process_event: [1/1/0, 1.1, 39]act_trunk_setuphtsp_setup_ind
*Mar  1 03:22:13.095: htsp_process_event: [1/1/0, 1.2 , 8]
*Mar  1 03:22:13.099: hdsprm_vtsp_codec_loaded_ok: G726 firmware needs download
*Mar  1 03:22:13.103: dsp_download: p=0x60E73844 size=34182 (t=1213310):39 FA 6D
*Mar  1 03:22:13.103: htsp_process_event: [1/1/0, 1.2 , 6]act_trunk_proc_connect
*Mar  1 03:22:13.191: dsp_receive_packet: MSG_TX_RESTART_INDICATION: code=0 t=1213319
*Mar  1 03:22:13.191: dsp_download: p=0x60EA8924 size=6224 (t=1213319): 8 55 AE
*Mar  1 03:22:13.207: dsp_receive_packet: MSG_TX_RESTART_INDICATION: code=0 t=1213320
```

```
*Mar  1 03:22:13.207: htsp_process_event: [1/1/0, 1.3 , 11] trunk_upfxols_trunk_up
*Mar  1 03:22:13.207: dsp_set_sig_state: [1/1/0] packet_len=12
      channel_id=128 packet_id=39 state=0x4 timestamp=0x0
*Mar  1 03:22:13.207: dsp_sig_encap_config: [1/1/0] packet_len=28 channel_id=128
      packet_id=42 transport_protocol=3 playout_delay=100 headerbytes = 0xA0
```

Note in the above display that “transport_protocol = 3” indicates Voice-over-Frame Relay. Also note that the second line of the display indicates that a **shutdown/no shutdown** command sequence was executed on the voice port.

Related Commands

Command	Description
debug vpm all	Enables debugging of all VPM areas.
debug vpm dsp	Shows messages from the DSP on the VPM to the router.
debug vpm signal	Collects debug information only for signaling events.
debug vpm spi	Displays information about how each network indication and application request is handled.

debug vpm signal

Use the **debug vpm signal** privileged EXEC command to collect debug information only for signaling events. The **no** form of this command disables debugging output.

debug vpm signal

no debug vpm signal

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vpm signal** command collects debug information only for signaling events. This command can also be useful in resolving problems with signaling to a PBX.

Examples

The following output shows that a ring is detected, and that the router waits for the ringing to stop before accepting the call:

```
ssm_process_event: [1/0/1, 0.2, 15] fxols_onhook_ringing
ssm_process_event: [1/0/1, 0.7, 19] fxols_ringing_not
ssm_process_event: [1/0/1, 0.3, 6]
ssm_process_event: [1/0/1, 0.3, 19] fxols_offhook_clear
```

The following output shows that the call is connected:

```
ssm_process_event: [1/0/1, 0.3, 4] fxols_offhook_proc
ssm_process_event: [1/0/1, 0.3, 8] fxols_proc_voice
ssm_process_event: [1/0/1, 0.3, 5] fxols_offhook_connect
```

The following output confirms a disconnect from the switch and release with higher layer code:

```
ssm_process_event: [1/0/1, 0.4, 27] fxols_offhook_disc
ssm_process_event: [1/0/1, 0.4, 33] fxols_disc_confirm
ssm_process_event: [1/0/1, 0.4, 3] fxols_offhook_release
```

debug vpm spi

Use the **debug vpm spi** privileged EXEC command to trace how the voice port module SPI interfaces with the call control API. The **no** form of this command disables debugging output.

debug vpm spi

no debug vpm spi

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vpm spi** command traces how the voice port module SPI interfaces with the call control API. This debug command displays information about how each network indication and application request is handled.

This debug level shows the internal workings of the voice telephony call state machine.

Examples

The following output shows that the call is accepted and presented to a higher layer code:

```
dsp_set_sig_state: [1/0/1] packet_len=14 channel_id=129 packet_id=39 state=0xC
timestamp=0x0
vcsm_process_event: [1/0/1, 0.5, 1] act_up_setup_ind
```

The following output shows that the higher layer code accepts the call, requests addressing information, and starts DTMF and dial-pulse collection. This also shows that the digit timer is started.

```
vcsm_process_event: [1/0/1, 0.6, 11] act_setup_ind_ack
dsp_voice_mode: [1/0/1] packet_len=22 channel_id=1 packet_id=73 coding_type=1
voice_field_size=160 VAD_flag=0 echo_length=128 comfort_noise=1 fax_detect=1
dsp_dtmf_mode: [1/0/1] packet_len=12 channel_id=1 packet_id=65 dtmf_or_mf=0
dsp_CP_tone_on: [1/0/1] packet_len=32 channel_id=1 packet_id=72 tone_id=3 n_freq=2
freq_of_first=350 freq_of_second=440 amp_of_first=4000 amp_of_second=4000 direction=1
on_time_first=65535 off_time_first=0 on_time_second=65535 off_time_second=0
dsp_digit_collect_on: [1/0/1] packet_len=22 channel_id=129 packet_id=35
min_inter_delay=550 max_inter_delay=3200 mim_make_time=18 max_make_time=75
min_brake_time=18 max_brake_time=75
vcsm_timer: 46653
```

The following output shows the collection of digits one by one until the higher level code indicates it has enough. The input timer is restarted with each digit and the device waits in idle mode for connection to proceed.

```
vcsm_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcsm_timer: 47055
vcsm_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcsm_timer: 47079
vcsm_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcsm_timer: 47173
vcsm_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcsm_timer: 47197
vcsm_process_event: [1/0/1, 0.7, 25] act_dcollect_digit
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
```

```
vcsm_timer: 47217
vcsm_process_event: [1/0/1, 0.7, 13] act_dcollect_proc
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
dsp_digit_collect_off: [1/0/1] packet_len=10 channel_id=129 packet_id=36
dsp_idle_mode: [1/0/1] packet_len=10 channel_id=1 packet_id=68
```

The following output shows that the network voice path cuts through:

```
vcsm_process_event: [1/0/1, 0.8, 15] act_bridge
vcsm_process_event: [1/0/1, 0.8, 20] act_caps_ind
vcsm_process_event: [1/0/1, 0.8, 21] act_caps_ack
dsp_voice_mode: [1/0/1] packet_len=22 channel_id=1 packet_id=73 coding_type=6
voice_field_size=20 VAD_flag=1 echo_length=128 comfort_noise=1 fax_detect=1
```

The following output shows that the called-party end of the connection is connected:

```
vcsm_process_event: [1/0/1, 0.8, 8] act_connect
```

The following output shows the voice quality statistics collected periodically:

```
vcsm_process_event: [1/0/1, 0.13, 17]
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=0
vcsm_process_event: [1/0/1, 0.13, 28]
vcsm_process_event: [1/0/1, 0.13, 29]
vcsm_process_event: [1/0/1, 0.13, 32]
vcsm_process_event: [1/0/1, 0.13, 17]
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=0
vcsm_process_event: [1/0/1, 0.13, 28]
vcsm_process_event: [1/0/1, 0.13, 29]
vcsm_process_event: [1/0/1, 0.13, 32]
vcsm_process_event: [1/0/1, 0.13, 17]
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=0
vcsm_process_event: [1/0/1, 0.13, 28]
vcsm_process_event: [1/0/1, 0.13, 29]
vcsm_process_event: [1/0/1, 0.13, 32]
```

The following output shows that the disconnection indication is passed to higher level code. The call connection is torn down, and final call statistics are collected:

```
vcsm_process_event: [1/0/1, 0.13, 4] act_generate_disc
vcsm_process_event: [1/0/1, 0.13, 16] act_bdrop
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
vcsm_process_event: [1/0/1, 0.13, 18] act_disconnect
dsp_get_levels: [1/0/1] packet_len=10 channel_id=1 packet_id=89
vcsm_timer: 48762
vcsm_process_event: [1/0/1, 0.15, 34] act_get_levels
dsp_get_tx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=86 reset_flag=1
vcsm_process_event: [1/0/1, 0.15, 31] act_stats_complete
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
dsp_digit_collect_off: [1/0/1] packet_len=10 channel_id=129 packet_id=36
dsp_idle_mode: [1/0/1] packet_len=10 channel_id=1 packet_id=68
vcsm_timer: 48762
dsp_set_sig_state: [1/0/1] packet_len=14 channel_id=129 packet_id=39 state=0x4
timestamp=0x0
vcsm_process_event: [1/0/1, 0.16, 5] act_wrelease_release
dsp_CP_tone_off: [1/0/1] packet_len=10 channel_id=1 packet_id=71
dsp_idle_mode: [1/0/1] packet_len=10 channel_id=1 packet_id=68
dsp_get_rx_stats: [1/0/1] packet_len=12 channel_id=1 packet_id=87 reset_flag=1
```

debug vsi api

Use the **debug vsi api** privileged EXEC command to display information on events associated with the external ATM API interface to the VSI master. The **no** form of this command disables debugging output.

debug vsi api

no debug vsi api

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.

Usage Guidelines

You can use **debug vsi api** command to monitor the communication between the VSI master and the XTagATM component about interface changes and cross-connect requests.

Examples

The following is an example of the display you see when you enter **debug vsi api**:

```
Router# debug vsi api

VSI_M: vsi_exatm_conn_req: 0x000C0200/1/35 -> 0x000C0100/1/50
       desired state up, status OK
VSI_M: vsi_exatm_conn_resp: 0x000C0200/1/33 -> 0x000C0100/1/49
       curr state up, status OK
```

Table 210 defines the significant fields shown in this display.

Table 210 debug vsi api Command Field Descriptions

Field	Description
vsi_exatm_conn_req	Indicates that a connect or disconnect request was submitted to the VSI Master.
0x000C0200	Logical interface identifier of the primary endpoint, in hexadecimal form.
1/35	VPI and VCI of the primary endpoint.
->	Indicates that the expected traffic flow is unidirectional (from the primary endpoint to the secondary endpoint). The other value for this field is “<->,” which indicates bidirectional traffic flow.
0x000C0100	Logical interface identifier of the secondary endpoint.

Table 210 debug vsi api Command Field Descriptions (continued)

Field	Description
1/50	VPI and VCI of the secondary endpoint.
desired state	Up indicates a connect request; Down indicates a disconnect request.
status (in vsi_exatm_conn_req output)	Mnemonic indicating the success or failure of the initial processing of the request. One of: • OK • INVALID_ARGS • NONEXIST_INTF • TIMEOUT • NO_RESOURCES • FAIL OK means only that the request was successfully queued for transmission to the switch; it does not indicate completion of the request.

debug vsi errors

Use the **debug vsi errors** privileged EXEC command to display information on errors encountered by the VSI Master. The **no** form of this command disables debugging output.

debug vsi errors [**interface** *interface* [**slave** *number*]]

no debug vsi errors [**interface** *interface* [**slave** *number*]]

Syntax Description

interface *interface*	(Optional) Interface number.
slave *number*	(Optional) Slave number (beginning with zero).

Usage Guidelines

You can use the **debug vsi errors** command to display information on errors encountered by the VSI master when parsing received messages, as well as information on unexpected conditions encountered by the VSI Master.

If the interface parameter is specified, output is restricted to errors associated with the indicated VSI control interface. If the slave number is specified, output is further restricted to errors associated with the session with the indicated slave.

Note Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple uses of the form of the command which specifies slave number allows multiple slaves to be debugged at once. For example, the following commands restrict output to that for errors associated with sessions 0 and 1 on control interface atm2/0 (but for no other sessions).

```
Router# debug vsi errors interface atm2/0 slave 0
Router# debug vsi errors interface atm2/0 slave 1
```

Some errors are not associated with any particular control interface or session; messages associated with these errors are printed regardless of the **interface** or **slave** options which are in effect.

Examples

The following is an example of the display you see when you enter **debug vsi errors**:

```
Router# debug vsi errors

VSI Master: parse error (unexpected param-group contents) in GEN ERROR RSP rcvd on
ATM2/0:0/51 (slave 0)
            errored section is at offset 16, for 2 bytes:
 01.01.00.a0 00.00.00.00 00.12.00.38 00.10.00.34
*00.01*00.69 00.2c.00.00 01.01.00.80 00.00.00.08
 00.00.00.00 00.00.00.00 00.00.00.00 0f.a2.00.0a
 00.01.00.00 00.00.00.00 00.00.00.00 00.00.00.00
 00.00.00.00
```

Table 211 lists the significant fields shown in this display.

Table 211 debug vsi errors Command Field Description

Field	Description
parse error	Indicates that an error has been encountered while parsing a message received by the VSI master.
unexpected param-group contents	Indicates the type of parsing error encountered. In this case, a parameter group within the message contained invalid data.
GEN ERROR RSP	Mnemonic for the function code in the header of the errored message.
ATM2/0	Control interface on which the errored message was received.
0/51	VPI/VCI of the VC (on the control interface) on which the errored message was received.
slave	Number of the session on which the errored message was received.
offset <n>	Indicates the number of bytes between the start of the VSI header the start of the errored portion of the message.
<n> bytes	Length of the errored section.
00.01.00.a0 [...]	Entire errored message, as a series of hexadecimal bytes. Note that the errored section is between asterisks (*).

debug vsi events

Use the **debug vsi events** privileged EXEC command to display information on events that affect entire sessions as well as events that affect only individual connections. The **no** form of this command disables debugging output.

debug vsi events [**interface** *interface* [**slave** *number*]]

no debug vsi events [**interface** *interface* [**slave** *number*]]

Syntax Description

interface *interface*	(Optional) Specifies the interface number.
slave *number*	(Optional) Specifies the slave number (beginning with zero).

Command History

Release	Modification
12.0(5)T	This command was introduced.

Usage Guidelines

You can use the **debug vsi events** command to display information on events associated with the per-session state machines of the VSI master, as well as the per-connection state machines. If the interface parameter is specified, output is restricted to events associated with the indicated VSI control interface. If the slave number is specified, output is further restricted to events associated with the session with the indicated slave.

Note Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple uses of the form of the command which specifies slave number allows multiple slaves to be debugged at once. For example, the following commands restrict output to that for events associated with sessions 0 and 1 on control interface atm2/0 (but for no other sessions). Output associated with all per-connection events are displayed regardless of the **interface** or **slave** options which are in effect.

```
Router# debug vsi events interface atm2/0 slave 0
Router# debug vsi events interface atm2/0 slave 1
```

Examples

The following is an example of the display you see when you enter **debug vsi events**:

```
Router# debug vsi events

VSI Master: conn 0xC0200/1/37->0xC0100/1/51:
            CONNECTING -> UP
VSI Master(session 0 on ATM2/0):
    event CONN_CMT_RSP, state ESTABLISHED -> ESTABLISHED
VSI Master(session 0 on ATM2/0):
    event KEEPALIVE_TIMEOUT, state ESTABLISHED -> ESTABLISHED
VSI Master(session 0 on ATM2/0):
    event SW_GET_CNFG_RSP, state ESTABLISHED -> ESTABLISHED
debug vsi packets
```

Table 212 defines the significant fields shown in this display.

Table 212 debug vsi events Command Field Description

Field	Description
conn	Indicates that the event applies to a particular connection.
0xC0200	Logical interface identifier of the primary endpoint, in hexadecimal form.
1/37	VPI or VCI of the primary endpoint.
->	Indicates the expected traffic flow is unidirectional (from the primary endpoint to the secondary endpoint.) The other value for this field is "<->," indicating bidirectional traffic flow.
0xC0100	Logical interface identifier of the secondary endpoint.
1/51	VPI or VCI of the secondary endpoint.
<state1> -> <state2>	• <state1> is a mnemonic for the state of the connection before the event occurred • <state2> represents the state of the connection after the event occurred
session	Indicates the number of the session with which the event is associated.
ATM2/0	Indicates the control interface associated with the session.
event	Mnemonic for the event that has occurred. This includes mnemonics for the function codes of received messages (for example, CONN_CMT_RSP), as well as mnemonics for other sorts of events (for example, KEEPALIVE_TIMEOUT).
state <state1> -> <state2>	Mnemonics for the session states associated with the transition triggered by the event. <state1> is a mnemonic for the state of the session before the event occurred; <state2> is a mnemonic for the state of the session after the event occurred.

debug vsi packets

Use the **debug vsi packets** privileged EXEC command to display a one-line summary of each VSI message sent and received by the LSC. The **no** form of this command disables debugging output.

debug vsi packets [**interface** *interface* [**slave** *number*]]

no debug vsi packets [**interface** *interface* [**slave** *number*]]

Syntax Description

interface *interface*	(Optional) Specifies the interface number.
slave *number*	(Optional) Specifies the slave number (beginning with zero).

Command History

Release	Modification
12.0(5)T	This command was introduced.

Usage Guidelines

If the interface parameter is specified, output is restricted to messages sent and received on the indicated VSI control interface. If the slave number is specified, output is further restricted to messages sent and received on the session with the indicated slave.

Note Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Multiple uses of the form of the command which specifies slave number allows multiple slaves to be debugged at once. For example, the following commands restrict output to that for messages received on atm2/0 for sessions 0 and 1, (but for no other sessions).

`Router#` **debug vsi packets interface atm2/0 slave 0**

`Router#` **debug vsi packets interface atm2/0 slave 1**

Examples

The following is an example of the display you see when you enter **debug vsi packets**:

```
Router# debug vsi packets

VSI master(session 0 on ATM2/0): sent msg SW GET CNFG CMD on 0/51
VSI master(session 0 on ATM2/0): rcvd msg SW GET CNFG RSP on 0/51
VSI master(session 0 on ATM2/0): sent msg SW GET CNFG CMD on 0/51
VSI master(session 0 on ATM2/0): rcvd msg SW GET CNFG RSP on 0/51
```

Table 213 defines the significant fields shown in this display.

Table 213 Debug VSI Packets Command Field Description

Field	Description
session	Session number identifying a particular VSI slave. Numbers begin with zero. See the **show controllers vsi session** command.
ATM2/0	Identifier for the control interface on which the message was sent or received.
sent	Message was sent by the VSI master.
rcvd	Message was received by the VSI master.
msg	Mnemonic for the function code from the message header.
0/51	VPI or VCI of the VC (on the control interface) on which the message was sent or received.

debug vsi param-groups

Use the **debug vsi param-groups** privileged EXEC command to display the first 128 bytes of each VSI message sent and received by the LSC (in hexadecimal form). The **no** form of this command disables debugging output.

debug vsi param-groups [**interface** *interface* [**slave** *number*]]

no debug vsi param-groups [**interface** *interface* [**slave** *number*]]

Syntax Description

interface *interface*	(Optional) Specifies the interface number.
slave *number*	(Optional) Specifies the slave number (beginning with zero).

Command History

Release	Modification
12.0(5)T	This command was introduced.

Usage Guidelines

This command is most commonly used with the **debug vsi packets** command to monitor incoming and outgoing VSI messages.

If the:

- Interface parameter is specified—Output is restricted to messages sent and received on the indicated VSI control interface
- Slave number is specified—Output is further restricted to messages sent and received on the session with the indicated slave.

Note Slave numbers are the same as the session numbers discussed under the **show controllers vsi session** command.

Note **param-groups** stands for parameter groups. A parameter group is a component of a VSI message.

Multiple uses of the form of the command, which specifies slave number, allows multiple slaves to be debugged at once. For example, the following commands restrict output to that for messages received on atm2/0 for sessions 0 and 1, (but for no other sessions).

```
Router# debug vsi param-groups interface atm2/0 slave 0
Router# debug vsi param-groups interface atm2/0 slave 1
```

Examples

The following is an example of the display you see when you enter **debug vsi param-groups**:

```
Router# debug vsi param-groups

Outgoing VSI msg of 12 bytes (not including encap):
 01.02.00.80 00.00.95.c2 00.00.00.00
Incoming VSI msg of 72 bytes (not including encap):
 01.02.00.81 00.00.95.c2 00.0f.00.3c 00.10.00.08
```

```
 00.01.00.00 00.00.00.00 01.00.00.08 00.00.00.09
 00.00.00.09 01.10.00.20 01.01.01.00 0c.08.80.00
 00.01.0f.a0 00.13.00.15 00.0c.01.00 00.00.00.00
 42.50.58.2d 56.53.49.31
Outgoing VSI msg of 12 bytes (not including encap):
 01.02.00.80 00.00.95.c3 00.00.00.00
Incoming VSI msg of 72 bytes (not including encap):
 01.02.00.81 00.00.95.c3 00.0f.00.3c 00.10.00.08
 00.01.00.00 00.00.00.00 01.00.00.08 00.00.00.09
 00.00.00.09 01.10.00.20 01.01.01.00 0c.08.80.00
 00.01.0f.a0 00.13.00.15 00.0c.01.00 00.00.00.00
 42.50.58.2d 56.53.49.31
```

Table 214 defines the significant fields shown in this display.

Table 214 debug vsi param-groups Command Field Descriptions

Field	Description
Outgoing	Message was sent by the VSI master.
Incoming	Message was received by the VSI master.
bytes	Number of bytes in the message, starting at the VSI header, and excluding the link layer encapsulation.
01.02...	Up to the first 128 bytes of the message, in hexadecimal form.

debug vtemplate

To display cloning information for a virtual access interface from the time it is cloned from a virtual template to the time the virtual access interface comes down when the call ends, use the **debug vtemplate** privileged EXEC command. The **no** form of this command disables debugging output.

debug vtemplate

no debug vtemplate

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug vtemplate** command when a virtual access interface comes up. The virtual access interface is cloned from virtual template 1.

```
Router# debug vtemplate

VTEMPLATE Reuse vaccess8, New Recycle queue size:50

VTEMPLATE set default vaccess8 with no ip address

Virtual-Access8 VTEMPLATE hardware address 0000.0c09.ddfd
VTEMPLATE vaccess8 has a new cloneblk vtemplate, now it has vtemplate
VTEMPLATE undo default settings vaccess8

VTEMPLATE ************* CLONE VACCESS8 *****************

VTEMPLATE Clone from vtemplate1 to vaccess8
interface Virtual-Access8
no ip address
encap ppp
ip unnumbered Ethernet0
no ip mroute-cache
fair-queue 64 256 0
no cdp enable
ppp authentication chap
end

%LINK-3-UPDOWN: Interface Virtual-Access8, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to up
```

The following is sample output from the **debug vtemplate** command when a virtual access interface goes down. The virtual interface is uncloned and returns to the recycle queue.

```
Router# debug vtemplate

%LINK-3-UPDOWN: Interface Virtual-Access8, changed state to down
VTEMPLATE Free vaccess8

%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to down
VTEMPLATE clean up dirty vaccess queue, size:1

VTEMPLATE Found a dirty vaccess8 clone with vtemplate
VTEMPLATE ************ UNCLONE VACCESS8 **************
VTEMPLATE Unclone to-be-freed vaccess8 command#7
interface Virtual-Access8
```

```
default ppp authentication chap
default cdp enable
default fair-queue 64 256 0
default ip mroute-cache
default ip unnumbered Ethernet0
default encap ppp
default ip address
end

VTEMPLATE set default vaccess8 with no ip address

VTEMPLATE remove cloneblk vtemplate from vaccess8 with vtemplate

VTEMPLATE Add vaccess8 to recycle queue, size=51
```

Table 215 describes the lines in the display.

Table 215 debug vtemplate Command Field Descriptions

Field	Description
VTEMPLATE Reuse vaccess8, New Recycle queue size:50 VTEMPLATE set default vaccess8 with no ip address	Virtual access interface 8 is reused; the current queue size is 50.
Virtual-Access8 VTEMPLATE hardware address 0000.0c09.ddfd	MAC address of virtual interface 8.
VTEMPLATE vaccess8 has a new cloneblk vtemplate, now it has vtemplate	Recording that virtual access interface 8 is cloned from the virtual interface template.
VTEMPLATE undo default settings vaccess8	Removing the default settings.
VTEMPLATE ********** CLONE VACCESS8 ********** *******	Banner: Cloning is in progress on virtual access interface 8.
VTEMPLATE Clone from vtemplate1 to vaccess8 interface Virtual-Access8 no ip address encap ppp ip unnumbered Ethernet0 no ip mroute-cache fair-queue 64 256 0 no cdp enable ppp authentication chap end	Specific configuration commands in virtual interface template 1 that are being applied to the virtual access interface 8.
%LINK-3-UPDOWN: Interface Virtual-Access8, changed state to up	Link status: The link is up.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to up	Line protocol status: The line protocol is up.
%LINK-3-UPDOWN: Interface Virtual-Access8, changed state to down	Link status: The link is down.
VTEMPLATE Free vaccess8	Freeing virtual access interface 8.

Table 215 debug vtemplate Command Field Descriptions (continued)

Field	Description
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access8, changed state to down	Line protocol status: The line protocol is down.
VTEMPLATE clean up dirty vaccess queue, size:1 VTEMPLATE Found a dirty vaccess8 clone with vtemplate VTEMPLATE ********** UNCLONE VACCESS8 **********	Access queue cleanup is proceeding and the template is being uncloned.
VTEMPLATE Unclone to-be-freed vaccess8 command#7 interface Virtual-Access8 default ppp authentication chap default cdp enable default fair-queue 64 256 0 default ip mroute-cache default ip unnumbered Ethernet0 default encap ppp default ip address end	Specific configuration commands to be removed from the virtual access interface 8.
VTEMPLATE set default vaccess8 with no ip address	Default is set again.
VTEMPLATE remove cloneblk vtemplate from vaccess8 with vtemplate	Removing the record of cloning from a virtual interface template.
VTEMPLATE Add vaccess8 to recycle queue, size=51	Virtual access interface is added to the recycle queue.

debug vtsp all

Use the **debug vtsp all** privileged EXEC command to show debugging information for all of the **debug vtsp** commands. Use the **no** form of this command to disable debugging output.

debug vtsp all

no debug vtsp all

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vtsp all** command enables the following debug vtsp commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**. For more information or sample output, refer to the individual commands in this chapter.

debug vtsp dsp

Use the **debug vtsp dsp** privileged EXEC command to show messages from the DSP on the VFC to the access server. Use the **no** form of this command to disable debugging output.

debug vtsp dsp

no debug vtsp dsp

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vtsp dsp** command shows messages from the DSP on the VFC to the router; this command can be useful if you suspect that the VFC is not functional. It is a simple way to check of the VFC is responding to off-hook indications.

Examples

Figure 43 shows the collection of DTMF digits from the DSP.

Figure 43 Sample Debug Vtsp Dsp Output

```
*Nov 30 00:44:34.491: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=3
*Nov 30 00:44:36.267: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=1
*Nov 30 00:44:36.571: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
*Nov 30 00:44:36.711: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
*Nov 30 00:44:37.147: vtsp_process_dsp_message: MSG_TX_DTMF_DIGIT: digit=2
```

debug vtsp port

To observe the behavior of the VTSP state machine, use the **debug vtsp port** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug vtsp port *slot-number/subunit-number/port*

no debug vtsp port *slot-number/subunit-number/port*

Syntax Description

slot-number	Specifies the slot number in the Cisco router where the voice interface card is installed. Valid entries are from 0 to 3, depending on the router being used and the slot where the voice interface card has been installed.
subunit-number	Specifies the subunit on the voice interface card where the voice port is located. Valid entries are 0 or 1.
port	Specifies the voice port. Valid entries are 0 or 1.

Command History

Release	Modification
12.0(3)XG	This command was introduced.

Usage Guidelines

This command is not supported on Cisco 7200 series routers or on the Cisco MC3810.

Use this command to limit the debug output to a particular port. The debug output can be quite voluminous for a single channel. A 12-port box might create problems. Use this debug command with any or all of the other debug modes.

Execution of the **no debug vtsp all** command will turn off all VTSP-level debugging. It is usually a good idea to turn off all debugging and then enter the debug commands you are interested in one by one. This will help to avoid confusion about which ports you are actually debugging.

Examples

The following example shows sample output from the **debug vtsp port 1/1/0** command:

```
router# debug vtsp port 1/1/0
*Mar  1 03:17:33.691: vtsp_tsp_call_setup_ind (sdb=0x613FD514, tdm_info=0x0,
   tsp_info=0x613FD438, calling_number= called_number= redirect_number=): peer_tag=1110
*Mar  1 03:17:33.691: vtsp_do_call_setup_ind
*Mar  1 03:17:33.691: dsp_close_voice_channel: [] packet_len=8 channel_id=1
   packet_id=75
*Mar  1 03:17:33.691: dsp_open_voice_channel: [] packet_len=12
   channel_id=1 packet_id=74 alaw_ulaw_select=0 transport_protocol=2
*Mar  1 03:17:33.695: dsp_set_playout_delay: [] packet_len=18
   channel_id=1 packet_id=76 mode=1 initial=60 min=4 max=200 fax_nom=300
*Mar  1 03:17:33.695: dsp_echo_canceller_control: [] packet_len=10 channel_id=1
   packet_id=66 flags=0x0
*Mar  1 03:17:33.695: dsp_set_gains: [] packet_len=12 channel_id=1 packet_id=91
   in_gain=0 out_gain=65506
*Mar  1 03:17:33.695: dsp_vad_enable: [] packet_len=10 channel_id=1 packet_id=78
   thresh=-38
*Mar  1 03:17:33.695: vtsp_process_event(): [, 0.S_SETUP_INDICATED, E_CC_PROCEEDING]
*Mar  1 03:17:33.699: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
   E_CC_BRIDGE]act_bridge
```

```
*Mar  1 03:17:33.699: vtsp_ring_noan_timer_start: 1185370
*Mar  1 03:17:33.699: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
   E_CC_CAPS_IND]act_caps_ind
*Mar  1 03:17:33.699: act_caps_ind: Encap 2, Vad 2, Codec 0x1000, CodecBytes 60,
              FaxRate 2, FaxBytes 30,
              Sub-channel 10, Bitmask 0x0 SignalType 2
*Mar  1 03:17:33.703: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
   E_CC_CAPS_ACK]act_caps_ack
*Mar  1 03:17:33.703: dsp_idle_mode: [] packet_len=8 channel_id=1 packet_id=68
*Mar  1 03:17:33.703: vtsp_process_event(): [, 0.S_SETUP_INDICATED,
   E_CC_CONNECT]act_connect
*Mar  1 03:17:33.703: vtsp_ring_noan_timer_stop: 1185370
*Mar  1 03:17:33.911: vtsp_process_event(): [, 0.S_CONNECT, E_DSPRM_PEND_SUCCESS]
   act_pend_codec_success
*Mar  1 03:17:33.911: dsp_close_voice_channel: [] packet_len=8 channel_id=1
   packet_id=75
*Mar  1 03:17:33.911: dsp_open_voice_channel: [] packet_len=12 channel_id=1
   packet_id=74 alaw_ulaw_select=0 transport_protocol=2
*Mar  1 03:17:33.911: dsp_set_playout_delay: [] packet_len=18 channel_id=1 packet_id=76
   mode=1 initial=60 min=4 max=200 fax_nom=300
*Mar  1 03:17:33.911: dsp_echo_canceller_control: [] packet_len=10 channel_id=1
   packet_id=66 flags=0x0
*Mar  1 03:17:33.911: dsp_set_gains: [] packet_len=12 channel_id=1 packet_id=91
   in_gain=0 out_gain=65506
*Mar  1 03:17:33.911: dsp_vad_enable: [] packet_len=10 channel_id=1 packet_id=78
   thresh=-38
*Mar  1 03:17:33.911: dsp_encap_config: [] packet_len=24 channel_id=1 packet_id=
 92 TransportProtocol 3 SID_support=0 sequence_number=0 rotate_flag=0 header_bytes 0xA0
*Mar  1 03:17:33.915: dsp_voice_mode: [] packet_len=22 channel_id=1 packet_id=73
 coding_type=14 voice_field_size=60 VAD_flag=1 echo_length=128
 comfort_noise=1 fax_detect=1 digit_relay=0
```

Related Commands

Command	Description
debug vpm all	Enables debugging of all VPM areas.
debug vtsp vofr subframe	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug vtsp session

Use the **debug vtsp session** privileged EXEC command to trace how the router interacts with the DSP based on the signaling indications from the signaling stack and requests from the application. Use the **no** form of this command to disable debugging output.

debug vtsp session

no debug vtsp session

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug vtsp session** command traces how the router interacts with the DSP based on the signalling indications from the signalling stack and requests from the application. This debug command displays information about how each network indication and application request is handled, signalling indications, and DSP control messages.

This debug level shows the internal workings of the voice telephony call state machine.

Examples

Figure 44 shows that the call has been accepted and that the system is now checking for incoming dial-peer matches.

Figure 44 Sample debug vtsp Session Output

Accept call and check for incoming dial-peer match:

```
*Nov 30 00:46:19.535: vtsp_tsp_call_accept_check (sdb=0x60CD4C58,
calling_number=408 called_number=1): peer_tag=0
*Nov 30 00:46:19.535: vtsp_tsp_call_setup_ind (sdb=0x60CD4C58,
tdm_info=0x60B80044, tsp_info=0x60B09EB0, calling_number=408 called_number=1):
peer_tag=1
```

Figure 45 shows that a DSP has been allocated to handle the call and indicated the call to the higher layer code.

Figure 45 Sample debug vtsp Session Output

```
*Nov 30 00:46:19.535: vtsp_do_call_setup_ind:
*Nov 30 00:46:19.535: dsp_open_voice_channel: [0:D:12] packet_len=12
channel_id=8737 packet_id=74 alaw_ulaw_select=0 transport_protocol=2
*Nov 30 00:46:19.535: dsp_set_playout_delay: [0:D:12] packet_len=18
channel_id=8737 packet_id=76 mode=1 initial=60 min=4 max=200 fax_nom=300
*Nov 30 00:46:19.535: dsp_echo_canceller_control: [0:D:12] packet_len=10
channel_id=8737 packet_id=66 flags=0x0
*Nov 30 00:46:19.539: dsp_set_gains: [0:D:12] packet_len=12 channel_id=8737
packet_id=91 in_gain=0 out_gain=0
*Nov 30 00:46:19.539: dsp_vad_enable: [0:D:12] packet_len=10 channel_id=8737
packet_id=78 thresh=-38
*Nov 30 00:46:19.559: vtsp_process_event: [0:D:12, 0.3, 13] act_setup_ind_ack
```

Figure 46 shows that the higher-layer code has accepted the call, placed the DSP in DTMF mode, and collected digits.

Figure 46 Sample debug vtsp Session Output

```
*Nov 30 00:46:19.559: dsp_voice_mode: [0:D:12] packet_len=20 channel_id=8737
packet_id=73 coding_type=1 voice_field_size=160 VAD_flag=0 echo_length=64
comfort_noise=1 fax_detect=1
*Nov 30 00:46:19.559: dsp_dtmf_mode: [0:D:12] packet_len=10 channel_id=8737
packet_id=65 dtmf_or_mf=0
*Nov 30 00:46:19.559: dsp_cp_tone_on: [0:D:12] packet_len=30 channel_id=8737
packet_id=72 tone_id=3 n_freq=2 freq_of_first=350 freq_of_second=440
amp_of_first=4000 amp_of_second=4000 direction=1 on_time_first=65535
off_time_first=0 on_time_second=65535 off_time_second=0
*Nov 30 00:46:19.559: vtsp_timer: 278792
*Nov 30 00:46:22.059: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.059: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.059: vtsp_timer: 279042
*Nov 30 00:46:22.363: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.363: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.363: vtsp_timer: 279072
*Nov 30 00:46:22.639: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.639: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.639: vtsp_timer: 279100
*Nov 30 00:46:22.843: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:22.843: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:22.843: vtsp_timer: 279120
*Nov 30 00:46:23.663: vtsp_process_event: [0:D:12, 0.4, 25] act_dcollect_digit
*Nov 30 00:46:23.663: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:23.663: vtsp_timer: 279202
```

Figure 47 shows that the call proceeded and that DTMF was disabled.

Figure 47 Sample debug vtsp Session Output

```
*Nov 30 00:46:23.663: vtsp_process_event: [0:D:12, 0.4, 15] act_dcollect_proc
*Nov 30 00:46:23.663: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:23.663: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
```

Figure 48 shows that the telephony call leg was conferenced with the packet network call leg and performed capabilities exchange with the network-side call leg.

Figure 48 Sample debug vtsp Session Output

```
*Nov 30 00:46:23.699: vtsp_process_event: [0:D:12, 0.5, 17] act_bridge
*Nov 30 00:46:23.699: vtsp_process_event: [0:D:12, 0.5, 22] act_caps_ind
*Nov 30 00:46:23.699: vtsp_process_event: [0:D:12, 0.5, 23] act_caps_ack
Go into voice mode with codec indicated in caps exchange.
*Nov 30 00:46:23.699: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:23.699: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
```

```
*Nov 30 00:46:23.699: dsp_voice_mode: [0:D:12] packet_len=20 channel_id=8737
packet_id=73 coding_type=6 voice_field_size=20 VAD_flag=1 echo_length=64
comfort_noise=1 fax_detect=1
```

Figure 49 shows the call connected at remote side.

Figure 49 Sample debug vtsp Session Output

```
*Nov 30 00:46:23.779: vtsp_process_event: [0:D:12, 0.5, 10] act_connect
```

Figure 50 shows that disconnect was indicated, and passed to upper layers.

Figure 50 Sample debug vtsp Session Output

```
*Nov 30 00:46:30.267: vtsp_process_event: [0:D:12, 0.11, 5] act_generate_disc
```

Figure 51 shows that the conference was torn down and disconnect handshake completed.

Figure 51 Sample debug vtsp Session Output

```
*Nov 30 00:46:30.267: vtsp_process_event: [0:D:12, 0.11, 18] act_bdrop
*Nov 30 00:46:30.267: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:30.267: vtsp_process_event: [0:D:12, 0.11, 20] act_disconnect
*Nov 30 00:46:30.267: dsp_get_error_stat: [0:D:12] packet_len=10 channel_id=0
packet_id=6 reset_flag=1
*Nov 30 00:46:30.267: vtsp_timer: 279862
```

Figure 52 shows that the final DSP statistics were retrieved.

Figure 52 Sample debug vtsp Session Output

```
*Nov 30 00:46:30.275: vtsp_process_event: [0:D:12, 0.17, 30] act_get_error
*Nov 30 00:46:30.275: 0:D:12: rx_dropped=0 tx_dropped=0 rx_control=353
tx_control=338 tx_control_dropped=0 dsp_mode_channel_1=2 dsp_mode_channel_2=0
c[0]=71 c[1]=71 c[2]=71 c[3]=71 c[4]=68 c[5]=71 c[6]=68 c[7]=73 c[8]=83 c[9]=84
c[10]=87 c[11]=83 c[12]=84 c[13]=87 c[14]=71 c[15]=6
*Nov 30 00:46:30.275: dsp_get_levels: [0:D:12] packet_len=8 channel_id=8737
packet_id=89
*Nov 30 00:46:30.279: vtsp_process_event: [0:D:12, 0.17, 34] act_get_levels
*Nov 30 00:46:30.279: dsp_get_tx_stats: [0:D:12] packet_len=10 channel_id=8737
packet_id=86 reset_flag=1
*Nov 30 00:46:30.287: vtsp_process_event: [0:D:12, 0.17, 31] act_stats_complete
*Nov 30 00:46:30.287: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:30.287: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
*Nov 30 00:46:30.287: vtsp_timer: 279864
```

Figure 53 shows that the DSP channel was closed and released.

Figure 53 Sample debug vtsp Session Output

```
*Nov 30 00:46:30.287: vtsp_process_event: [0:D:12, 0.18, 6] act_wrelease_release
*Nov 30 00:46:30.287: dsp_cp_tone_off: [0:D:12] packet_len=8 channel_id=8737
packet_id=71
*Nov 30 00:46:30.287: dsp_idle_mode: [0:D:12] packet_len=8 channel_id=8737
packet_id=68
*Nov 30 00:46:30.287: dsp_close_voice_channel: [0:D:12] packet_len=8
channel_id=8737 packet_id=75
*Nov 30 00:46:30.287: vtsp_process_event: [0:D:12, 0.16, 42] act_terminate
```

debug vtsp vofr subframe

To display the first 10 bytes (including header) of selected VoFR subframes for the interface, use the **debug vtsp vofr subframe** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug vtsp vofr subframe *payload* [**from-dsp**] [**to-dsp**]

no debug vtsp vofr subframe

Syntax Description

payload	Number used to selectively display subframes of a specific payload. The payload types are: • 0—Primary Payload **Caution** This option may cause network instability • 1—Annex-A • 2—Annex-B • 3—Annex-D • 4—All other payloads • 5—All payloads **Caution** This option may cause network instability
from-dsp	(Optional) Displays only the subframes received from the DSP.
to-dsp	(Optional) Displays only the subframes going to the DSP.

Command History

Release	Modification
12.0(3)XG	This command was introduced.

Usage Guidelines

This command is not supported on Cisco 7200 series routers or on the Cisco MC3810.

Each debug output displays the first 10 bytes of the FRF.11 subframe, including header bytes. The **from-dsp** and **to-dsp** options can be used to limit the debugs to a single direction. If not specified, debugs are displayed for subframes when they are received from the DSP and before they are sent to the DSP.

Use extreme caution in selecting payload options 0 and 5. These options may cause network instability.

Examples

The following example shows sample output from the **debug vtsp vofr subframe** command:

```
router# debug vtsp vofr subframe 2
vtsp VoFR subframe debugging is enabled for payload 2 to and from DSP 3620_vofr#
*Mar  6 18:21:17.413:VoFR frame received from Network (24 bytes):9E 02 19 AA AA AA AA
```

```
      AA AA AA
*Mar  6 18:21:17.449:VoFR frame received from DSP (18 bytes):9E 02 19 AA AA AA AA AA AA
      AA
*Mar  6 18:21:23.969:VoFR frame received from Network (24 bytes):9E 02 19 AA AA AA AA
      AA AA AA
*Mar  6 18:21:24.005:VoFR frame received from DSP (18 bytes):9E 02 19 AA AA AA AA AA AA
      AA
```

Related Commands

Command	Description
debug vtsp all	Enables debugging of all VTSP areas.
debug vtsp port	Shows the behavior of the VTSP state machine.

debug x25

To display information about X.25 traffic, use one of the following **debug x25** privileged EXEC commands. The commands allow you to display all information or an increasingly restrictive part of the information.

Caution This command is processor intensive and can render the router useless. Use this command only when the aggregate of all reportable X.25 traffic is fewer than five packets per second (pps). The generic forms of this command should be restricted to low-speed, low-usage links running at less than 19.2 kbps. Because the **debug x25 vc** command and the **debug x25 vc events** command display traffic for only a small subset of virtual circuits, they are safer to use under heavy traffic conditions, as long as events for that virtual circuit are fewer than 25 pps.

To display information about all X.25 traffic, including traffic for X.25, Connection Mode Network Service (CMNS), and X.25 over TCP (XOT) services, use the **debug x25** command (default **all**). Use the **no** form of this command to disable debugging output.

debug x25

no debug x25

To display information about all X.25 traffic except data and resource record packets, use the **debug x25 events** command. Use the **no** form of this command to disable debugging output.

debug x25 events

no debug x25 events

To display information about a specific X.25 service class, use the following form of the **debug x25** command. Use the **no** form of this command to disable debugging output.

debug x25 [**only** | **cmns** | **xot**] [**events** | **all**]

no debug x25 [**only** | **cmns** | **xot**] [**events** | **all**]

To display information about a specific X.25 or CMNS context, use the following form of the **debug x25** command. Use the **no** form of this command to disable debugging output.

debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} [**events** | **all**]

no debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} [**events** | **all**]

To display information about a specific X.25 or CMNS virtual circuit, use the following form of the **debug x25** command. Use the **no** form of this command to disable debugging output.

debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} **vc** *number* [**events** | **all**]

no debug x25 interface {*serial-interface* | *cmns-interface* **mac** *mac-address*} **vc** *number* [**events** | **all**]

To display information about traffic for all virtual circuits using a given number, use the following form of the **debug x25** command. The **no** form of this command removes the filter for a particular virtual circuit from the **debug x25 all** or **debug x25 events** output. Use the **no** form of this command to disable debugging output.

debug x25 vc *number* [**events** | **all**]

no debug x25 vc *number* [**events** | **all**]

To display information about traffic to or from a specific XOT host, use the following form of the **debug x25 xot** command. Use the **no** form of this command to disable debugging output.

debug x25 xot [**remote** *ip-address* [**port** *number*]] [**local** *ip-address* [**port** *number*]] [**events** | **all**]

no debug x25 xot [**remote** *ip-address* [**port** *number*]] [**local** *ip-address* [**port** *number*]] [**events** | **all**]

Use the **debug x25** command with the **aodi** keyword to display information about an interface running PPP over an X.25 session. The **no** form of this command disables debugging output. Use the **no** form of this command to disable debugging output.

debug x25 aodi

no debug x25 aodi

Syntax Description

events	(Optional) Displays all traffic except Data and Receiver Ready (RR) packets.
only | **cmns** | **xot**	(Optional) Displays information about the specified services: X.25 only, CMNS, or XOT.
all	(Optional) Displays all traffic.
serial-interface	X.25 serial interface.
cmns-interface **mac** *mac-address*	MAC address of the CMNS interface and remote host. The interface type can be Ethernet, Token Ring, or FDDI.
vc *number*	Virtual circuit number, in the range 1 to 4095.
remote *ip-address* [**port** *number*]	(Optional) Remote IP address and, optionally, a port number in the range 1 to 65535.
local *ip-address* [**port** *number*]	(Optional) Local host IP address and, optionally, a port number in the range 1 to 65535.
aodi	Causes the **debug x25** command to display Always On/Dynamic ISDN (AO/DI) events and processing information.

Defaults

The default is that all traffic is displayed.

Command History

Release	Modification
10.0	This command was introduced.
12.0(5)T	For DNS-based X.25 routing, additional functionality was added to the **debug x25 events** command to describe the events occurring while resolving the X.25 address to an IP address using a DNS server. The **debug domain** command can be used along with **debug x25 events** to observe the whole DNS-based X.25 routing data flow. (For more details, see "debug x25 events for DNS-Based X.25 Routing" in the "Examples" section.)
12.0(7)T	For the X.25 CUGs feature, functionality was added to the **debug x25 events** command to describe events occurring during CUG activity. (For more details, see "debug x25 events for X.25 CUGs" in the "Examples" section.)

Usage Guidelines

This command is particularly useful for diagnosing problems encountered when placing calls. The **debug x25 all** output includes data, control messages, and flow control packets for all virtual circuits of the router.

All **debug x25** command forms can take either the **events** or **all** keyword. The keyword **all** is the default and causes all packets meeting the other debug criteria to be reported. The keyword **events** omits reports of any Data or Receiver Ready (RR) flow control packets; the normal flow of data and RR packets is commonly large and less interesting to the user, so event reporting can significantly decrease the processor load induced by debug reporting.

The **debug x25 interface** command is useful for diagnosing problems encountered with a single X.25 or CMNS host or virtual circuit.

Because no interface is specified by the **debug x25 vc** command, traffic on any virtual circuit that has the specified number is reported.

Virtual circuit zero (**vc 0**) cannot be specified. It is used for X.25 service messages, such as RESTART packets, not virtual circuit traffic. Service messages can be monitored only when no virtual circuit filter is used.

The **debug x25 xot** output allows you to restrict the debug output reporting to XOT traffic for one or both hosts or host/port combinations. Because each XOT virtual circuit uses a unique TCP connection, an XOT debug request that specifies both host addresses and ports will report traffic only for that virtual circuit. Also, you can restrict reporting to sessions initiated by the local or remote router by specifying 1998 for the remote or local port. (XOT connections are received on port 1998.)

Use the **debug x25 aodi** command to display interface PPP events running over an X.25 session and to debug X.25 connections between a client and server configured for AO/DI.

Examples

The following is sample output from the **debug x25** command, displaying output concerning the functions X.25 restart, call setup, data exchange, and clear:

```
Router# debug x25

Serial0: X.25 I R/Inactive Restart (5) 8 lci 0
  Cause 7, Diag 0 (Network operational/No additional information)
Serial0: X.25 O R3 Restart Confirm (3) 8 lci 0
Serial0: X.25 I P1 Call (15) 8 lci 1
From(6): 170091 To(6): 170090
   Facilities: (0)
   Call User Data (4): 0xCC000000 (ip)
```

```
Serial0: X.25 O P3 Call Confirm (3) 8 lci 1
Serial0: X.25 I D1 Data (103) 8 lci 1 PS 0 PR 0
Serial0: X.25 O D1 Data (103) 8 lci 1 PS 0 PR 1
Serial0: X.25 I P4 Clear (5) 8 lci 1
  Cause 9, Diag 122 (Out of order/Maintenance action)
Serial0: X.25 O P7 Clear Confirm (3) 8 lci 1
```

Table 216 describes the fields shown in the display.

Table 216 debug x25 Field Descriptions

Field	Description
Serial0	Interface on which the X.25 event occurred.
X.25	Type of event this message describes.
I	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.
R3	State of the service or virtual circuit (VC). Possible values follow: • R/Inactive—Packet layer awaiting link layer service • R1—Packet layer ready • R2—Data terminal equipment (DTE) restart request • R3—Data circuit-terminating equipment (DCE) restart indication • P/Inactive—VC awaiting packet layer service • P1—Idle • P2—DTE waiting for DCE to connect CALL • P3—DCE waiting for DTE to accept CALL • P4—Data transfer • P5—CALL collision • P6—DTE clear request • P7—DCE clear indication • D/Inactive—VC awaiting setup • D1—Flow control ready • D2—DTE reset request • D3—DCE reset indication See Annex B of the *ITU-T Recommendation X.25* for more information on these states.

Table 216 debug x25 Field Descriptions (continued)

Field	Description
Restart	The type of X.25 packet. Possible values follow: • R Events —Restart —Restart Confirm —Diagnostic • P Events —Call —Call Confirm —Clear —Clear Confirm • D Events —Reset —Reset Confirm • D1 Events —Data —RNR (Receiver Not Ready) —RR (Receiver Ready) —Interrupt —Interrupt Confirm • XOT Overhead —PVC Setup
(5)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 or 128.
lci 0	VC number. See Annex A of the *ITU-T Recommendation X.25* for information on VC assignment.
Cause 7	Code indicating the event that triggered the packet. The Cause field can only appear in entries for Clear, Reset, and Restart packets. Possible values for the Cause field can vary, depending on the type of packet. Refer to the "X.25 Cause and Diagnostic Codes" appendix for an explanation of these codes.
Diag 0	Code providing an additional hint as to what, if anything, went wrong. The Diag field can only appear in entries for Clear, Diagnostic (as "error 0"), Reset, and Restart packets. Refer to the "X.25 Cause and Diagnostic Codes" appendix for an explanation of these codes.
(Network operational/ No additional information)	The standard explanations of the Cause and Diagnostic codes (*cause/diag*).

The following example shows a sequence of increasingly restrictive **debug x25** commands:

```
Router# debug x25
X.25 packet debugging is on

Router# debug x25 events
X.25 special event debugging is on

Router# debug x25 interface serial 0
X.25 packet debugging is on
X.25 debug output restricted to interface Serial0

Router# debug x25 vc 1024
X.25 packet debugging is on
X.25 debug output restricted to VC number 1024

Router# debug x25 interface serial 0 vc 1024
X.25 packet debugging is on
X.25 debug output restricted to interface Serial0
X.25 debug output restricted to VC number 1024

Router# debug x25 interface serial 0 vc 1024 events
X.25 special event debugging is on
X.25 debug output restricted to interface serial 0
X.25 debug output restricted to VC number 1024
```

The following examples show the normal sequence of events for both the AO/DI client and server sides:

Client Side

```
Router# debug x25 aodi
PPP-X25: Virtual-Access1: Initiating AODI call request
PPP-X25: Bringing UP X.25 AODI VC
PPP-X25: AODI Client Call Confirm Event Received
PPP-X25: Cloning interface for AODI is Di1
PPP-X25: Queuing AODI Client Map Event
PPP-X25: Event:AODI Client Map
PPP-X25: Created interface Vi2 for AODI service
PPP-X25: Attaching primary link Vi2 to Di1
PPP-X25: Cloning Vi2 for AODI service using Di1
PPP-X25: Vi2: Setting the PPP call direction as OUT
PPP-X25: Vi2: Setting vectors for RFC1598 operation on BRI3/0:0 VC 0
PPP-X25: Vi2: Setting the interface default bandwidth to 10 Kbps
PPP-X25: Virtual-Access2: Initiating AODI call request
PPP-X25: Bringing UP X.25 AODI VC
PPP-X25: AODI Client Call Confirm Event Received
```

Server Side

```
Router# debug x25 aodi
PPP-X25: AODI Call Request Event Received
PPP-X25: Event:AODI Incoming Call Request
PPP-X25: Created interface Vi1 for AODI service
PPP-X25: Attaching primary link Vi1 to Di1
PPP-X25: Cloning Vi1 for AODI service using Di1
PPP-X25: Vi1: Setting vectors for RFC1598 operation on BRI3/0:0 VC 1
PPP-X25: Vi1: Setting the interface default bandwidth to 10 Kbps
PPP-X25: Binding X.25 VC 1 on BRI3/0:0 to Vi1
```

debug x25 events for X.25 CUGs

The following example of the **debug x25 events** command shows output related to the X.25 CUGs feature. It shows messages concerning a DCE rejecting a call because the selected network CUG had not been subscribed to by the caller.

```
Router# debug x25 events
00:48:33:Serial1:X.25 I R1 Call (14) 8 lci 1024
00:48:33:  From (3):111 To (3):444
00:48:33:  Facilities:(2)
00:48:33:    Closed User Group (basic):40
00:48:33:  Call User Data (4):0x01000000 (pad)
00:48:33:X.25 Incoming Call packet, Closed User Group (CUG) protection, selected network
CUG not subscribed
00:48:33:Serial1:X.25 O R1 Clear (5) 8 lci 1024
00:48:33:  Cause 11, Diag 65 (Access barred/Facility code not allowed)
```

debug x25 events for DNS-Based X.25 Routing

The following example of the **debug x25 events** command shows output related to the DNS-Based X.25 Routing feature. It shows messages concerning access of the DNS server. In the following example, nine alternate addresses for one XOT path are entered in the DNS server database. All nine addresses are returned to the host cache of the router by the DNS server. However, only six addresses will be used during the XOT switch attempt, because this is the limit that XOT allows.

```
Router# debug x25 events
00:18:25:Serial1:X.25 I R1 Call (11) 8 lci 1024
00:18:25:  From (0): To (4):444
00:18:25:  Facilities:(0)
00:18:25:  Call User Data (4):0x01000000 (pad)
00:18:25:X.25 host name sent for DNS lookup is "444"
00:18:26:%3-TRUNCATE_ALT_XOT_DNS_DEST:Truncating excess XOT addresses (3)
returned by DNS
00:18:26:DNS got X.25 host mapping for "444" via network
00:18:32:[10.1.1.8 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:38:[10.1.1.7 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:44:[10.1.1.6 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:50:[10.1.1.5 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:18:56:[10.1.1.4 (pending)]:XOT open failed (Connection timed out; remote host not
responding)
00:20:04:[10.1.1.3,1998/10.1.1.3,11007]:XOT O P2 Call (17) 8 lci 1
00:20:04:  From (0): To (4):444
00:20:04:  Facilities:(6)
00:20:04:    Packet sizes:128 128
00:20:04:    Window sizes:2 2
00:20:04:  Call User Data (4):0x01000000 (pad)
00:20:04:[10.1.1.3,1998/10.1.1.3,11007]:XOT I P2 Call Confirm (11) 8 lci 1
00:20:04:  From (0): To (0):
00:20:04:  Facilities:(6)
00:20:04:    Packet sizes:128 128
00:20:04:    Window sizes:2 2
00:20:04:Serial1:X.25 O R1 Call Confirm (5) 8 lci 1024
00:20:04:  From (0): To (0):
00:20:04:  Facilities:(0)
```

Related Commands

Command	Description
debug ppp bap	Displays general BACP transactions.
debug ppp bap negotiation	Displays general BACP transactions, as well as successive steps in negotiations between peers.
debug ppp multilink	Displays information about important multilink events.
debug ppp multilink negotiation	Displays information about important multilink events and events affecting multilink groups controlled by BACP.

debug x25 annexg

To display information about Annex G (X.25 over Frame Relay) events, use the **debug x25 annexg** command. To disable debugging output, use the **no** form of this command.

debug x25 annexg

no debug x25 annexg

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0 T	This command was introduced.

Usage Guidelines

It is generally recommended that the **debug x25 annexg** command be used only when specifically requested by Cisco TAC to obtain information about a problem with an Annex G configuration. The messages displayed by the **debug x25 annexg** command are meant to aid in the diagnosing of internal errors.

Caution The X.25 debug commands can generate large amounts of debugging output. If logging of debug output to the router console is enabled (the default condition), this output may fill the console buffer, preventing the router from processing packets until the contents of the console buffer have been printed.

Examples

The following example shows sample output for the **debug x25 annexg** command for a Frame Relay data-link connection identifier (DLCI) configured for Annex G operation:

```
Router# debug x25 annexg

Jul 31 05:23:20.316:annexg_process_events:DLCI 18 attached to interface Serial2/0:0 is
ACTIVE
Jul 31 05:23:20.316:annexg_ctxt_create:Creating X.25 context over Serial2/0:0 (DLCI:18
using X.25 profile:OMC), type 10, len 2, addr 00 12
Jul 31 05:23:20.316:annexg_create_lower_layer:Se2/0:0 DLCI 18, payload 1606, overhead 2
Jul 31 05:23:20.320:annexg_restart_tx:sending pak to Serial2/0:0
Jul 31 05:23:23.320:annexg_restart_tx:sending pak to Serial2/0:0
```

Table 217 describes significant fields shown in the display.

Table 217 debug x25 annexg Field Descriptions

Field	Description
payload	Amount of buffer space available per message before adding Frame Relay and device-specific headers.
overhead	The length of the Frame Relay header and any device-specific header that may be needed.

Related Commands

Command	Description
debug x25	Displays information about X.25 traffic.

debug x28

Use the **debug x28** privileged privileged EXEC command to monitor error information and X.28 connection activity. The **no** form of this command disables debugging output.

debug x28

no debug x28

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output while the PAD initiates an X.28 outgoing call:

```
Router# debug x28
X28 MODE debugging is on
Router# x28

*
03:30:43: X.28 mode session started
03:30:43: X28 escape is exit
03:30:43: Speed for console & vty lines :9600
*call 123456
COM
03:39:04: address ="123456", cud="[none]" 03:39:04: Setting X.3 Parameters for this
call...1:1 2:1 3:126 4:0 5:1 6:2 7:2 8:0 9:0 10:0 11:14 12:1 13:0 14:0 15:0 16:127 17:24
18:18 19:2 20:0 21:0 22:0

Router> exit
CLR CONF

*
*03:40:50: Session ended
* exit

Router#
*03:40:51: Exiting X.28 mode
```

debug xcctsp all

To debug External Call Control TSP information, use the **debug xcctsp all** privileged EXEC command. To turn off debugging, use the **no** form of this command.

debug xcctsp all

no debug xcctsp all

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off external call control debugging:

```
AS5300-TGW# debug xcctsp all
External call control all debugging is on

AS5300-TGW# no debug xcct all
External call control all debugging is off

AS5300-TGW#
```

Related Commands

Command	Description
debug xcctsp error	Enables debugging on external call control errors.
debug xcctsp session	Enables debugging on external call control sessions.

debug xcctsp error

To debug External Call Control TSP error information, use the **debug xcctsp error** privileged EXEC command. To turn off error debugging, use the **no** form of this command.

debug xcctsp error

no debug xcctsp error

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off error-level debugging:

```
AS5300-TGW# debug xcctsp error
External call control error debugging is on

AS5300-TGW# no debug xcctsp error
External call control error debugging is off
```

Related Commands

Command	Description
debug xcctsp all	Enables debugging on all external call control levels.
debug xcctsp session	Enables debugging on external call control sessions.

debug xcctsp session

To debug External Call Control TSP session information, use the **debug xcctsp session** privileged EXEC command. To turn off debugging, use the **no** form of this command.

debug xcctsp session

no debug xcctsp session

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(5)T	This command was introduced.
12.0(7)T	Support for this command was extended to the Cisco uBR924 cable modem.

Examples

See the following examples to turn on and off session-level debugging:

```
AS5300-TGW# debug xcct session
External call control session debugging is on

AS5300-TGW# no debug xcct session
External call control session debugging is off

AS5300-TGW#
```

Related Commands

Command	Description
debug xcctsp all	Enables debugging on external call control levels.
debug xcctsp error	Enables debugging on external call control errors.

debug xns packet

Use the **debug xns packet** privileged EXEC command to display information on XNS packet traffic, including the addresses for source, destination, and next hop router of each packet. The **no** form of this command disables debugging output.

debug xns packet

no debug xns packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

To gain the fullest understanding of XNS routing activity, you should enable **debug xns routing** and **debug xns packet** together.

Examples

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

```
Router# debug xns packet

XNS: src=5.0000.0c02.6d04, dst=5.ffff.ffff.ffff, packet sent
XNS: src=1.0000.0c00.440f, dst=1.ffff.ffff.ffff, rcvd. on Ethernet0
XNS: src=1.0000.0c00.440f, dst=1.ffff.ffff.ffff, local processing
```

Table 218 describes significant fields in the display.

Table 218 debug xns packet Command Field Descriptions

Field	Description
XNS:	Indicates that this is an XNS packet.
src = 5.0000.0c02.6d04	Indicates that the source address for this message is 0000.0c02.6d04 on network 5.
dst = 5.ffff.ffff.ffff	Indicates that the destination address for this message is the broadcast address ffff.ffff.ffff on network 5.
packet sent	Indicates that the packet to destination address 5.ffff.ffff.ffff has displayed using the **debug xns packet** command, was queued on the output interface.
rcvd. on Ethernet0	Indicates that the router just received this packet through the Ethernet0 interface.
local processing	Indicates that the router has examined the packet and determined that it must process it, rather than forwarding it.

debug xns routing

Use the **debug xns routing** privileged EXEC command to display information on XNS routing transactions. The **no** form of this command disables debugging output.

debug xns routing

no debug xns routing

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

To gain the fullest understanding of XNS routing activity, enable **debug xns routing** and **debug xns packet** together.

Examples

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

```
Router# debug xns routing

XNSRIP: sending standard periodic update to 5.ffff.ffff.ffff via Ethernet2
 network 1, hop count 1
 network 2, hop count 2

XNSRIP: got standard update from 1.0000.0c00.440f socket 1 via Ethernet0
 net 2: 1 hops
```

Table 219 describes significant fields in the display.

Table 219 debug xns routing Command Field Descriptions

Field	Description
XNSRIP:	This is an XNS routing packet.
sending standard periodic update	Router indicates that this is a periodic XNS routing information update.
to 5.ffff.ffff.ffff	Destination address is ffff.ffff.ffff on network 5.
via Ethernet2	Name of the output interface.
network 1, hop count 1	Network 1 is one hop away from this router.
got standard update from 1.0000.0c00.440f	Router indicates that it has received an XNS routing information update from address 0000.0c00.440f on network 1.
socket 1	The socket number is a well-known port for XNS. Possible values include • 1—routing information • 2—echo • 3—router error