

debug vpdn

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

[no] {error | event [disconnect] | l2f-errors | l2f-events | l2f-packets | packet}

Syntax Description

error	Displays VPDN errors.
event	Displays VPDN events.
disconnect	(Optional) Displays VPDN disconnect events.
l2f-errors	Displays L2F protocol errors.
l2f-events	Displays L2F protocol events.
l2f-packets	Displays detailed information about L2F control packets.
packet	Displays information about VPDN packets.

Usage Guidelines

Note that the **debug vpdn packet** command generates several debug operations per packet. Depending on the L2F traffic pattern, this command may cause the CPU load to increase to a high level that impacts performance.

Sample Display

This section contains the following examples:

- Debugging VPDN Events on a NAS—Normal Operations
- Debugging VPDN Events on the Home Gateway—Normal Operations
- Debugging Protocol-Specific Events on the NAS—Normal Operations
- Debugging Protocol-Specific Events on the Home Gateway—Normal Operations
- Debugging Errors on the NAS—Error Conditions
- Debugging L2F Control Packets for Complete Information

Example 1 Debugging VPDN Events on a NAS—Normal Operations

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

```
vpdn outgoing cisco.com nas1 ip 172.21.9.26
username stella 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 183 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 183 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 --	Domain name is identified.
Async6 VPN Forwarding...	
Async6 VPN Bind interface direction=1	Tunnel is bound to the interface. These are the direction values: <ul style="list-style-type: none"> • 1—From the NAS to the home gateway • 2—From the home gateway 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.

Table 183 debug vpdn event Field Descriptions for the NAS (continued)

Field	Description
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	Normal cleanup operations performed when the line or virtual access interface goes down.
Async6 VPN reset	
Async6 VPN Unbind interface	

Example 2 Debugging VPDN Events on the Home Gateway—Normal Operations

The home gateway 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 incoming nas1 nas1 virtual-template 1
```

The following is sample output from the **debug vpdn event** command on the home gateway 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 home gateway 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 184 describes the fields shown in two previous outputs. The output describes normal operations when an L2F tunnel is brought up or down on a home gateway.

Table 184 debug vpdn event Field Descriptions for the Home Gateway

Field	Description
Tunnel coming up	
L2F: Chap authentication succeeded for nas1.	PPP CHAP authentication status for the tunnel named nas1.

Table 184 debug vpdn event Field Descriptions for the Home Gateway (continued)

Field	Description
Virtual-Access3 VPN Virtual interface created for user6@cisco.com	Virtual access interface was set up on the home gateway 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: <ul style="list-style-type: none"> • 1—From the NAS to the home gateway • 2—From the home gateway 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 home gateway 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.

Example 3 Debugging Protocol-Specific Events on the NAS—Normal Operations

The following is sample output from the **debug vpdn l2f-events** command on the NAS when an L2F tunnel is brought up successfully:

```
Router# debug vpdn l2f-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 l2f-events** command on a NAS when an L2F tunnel is brought down normally:

```
Router# debug vpdn l2f-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 185 describes the fields shown in the displays.

Table 185 debug vpdn l2f-events Field Descriptions—NAS

Field	Description
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 home gateway IP address.
L2F_CONF received	L2F_CONF signal was received. When sent from the home gateway to the NAS, an L2F_CONF indicates the home gateway's recognition of the tunnel creation request.

Table 185 debug vpdn l2f-events Field Descriptions—NAS (continued)

Field	Description
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 home gateway accepted the NAS configuration of an L2F tunnel.
L2F building nas2gw_mid0	L2F is building a tunnel between the NAS and the home gateway, 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 home gateway 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 home gateway.
L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6	L2F synchronized the Client IDs on the NAS and the home gateway, 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.

Table 185 debug vpdn l2f-events Field Descriptions—NAS (continued)

Field	Description
%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.

Example 4 Debugging Protocol-Specific Events on the Home Gateway—Normal Operations

The following is sample output from the **debug vpdn l2f-events** command on a home gateway when an L2F tunnel is created:

```
Router# debug vpdn l2f-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 l2f-events** command on a home gateway when the L2F tunnel is brought down normally:

```
Router# debug vpdn l2f-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 186 describes the significant fields shown in the displays.

Table 186 debug vpdn l2f-events Field Descriptions—Home Gateway

Field	Description
Tunnel coming up	
L2F_CONF received	L2F configuration is received from the NAS. When sent from a NAS to a home gateway, the L2F_CONF is the initial packet in the conversation.

Table 186 debug vpdn l2f-events Field Descriptions—Home Gateway (continued)

Field	Description
L2F Creating new tunnel for nas1	Tunnel named <i>nas1</i> is being created.
L2F Got a tunnel named nas1, responding	Home gateway 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.
L2F Got a MID management packet	L2F MID management packets are used to communicate between the NAS and the home gateway.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up	Home gateway 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 home gateway 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.

Table 186 debug vpdn l2f-events Field Descriptions—Home Gateway (continued)

Field	Description
L2F_CLOSE received	Router is performing normal cleanup operations when the tunnel is being brought down.
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	Line protocol is down; virtual access interface 1 cannot be used.

Example 5 Debugging Errors on the NAS—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 187 describes the significant fields shown in the display.

Table 187 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 from asynchronous interface 1 link went down and then came back up.
%LINK-3-UPDOWN: Interface Async1, changed state to up	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up	Line protocol on the asynchronous interface came back up.

Table 187 **debug vpdn error Field Descriptions for the NAS (continued)**

Field	Description
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 home gateway 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 l2f-errors** command:

```
Router# debug vpdn l2f-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 188 describes the significant fields shown in the display.

Table 188 **debug vpdn l2f-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 home gateway, 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.
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.

Example 6 Debugging L2F Control Packets for Complete Information

The following is sample output from the **debug vpdn l2f-packets** command on a NAS. This example displays a trace for a **ping** command:

```
Router# debug vpdn l2f-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
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 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
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
```

Table 189 describes the significant fields shown in the display.

Table 189 debug vpdn l2f-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 home gateway is always 1, indicating L2F management.
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.

Table 189 debug vpdn l2f-packets Field Descriptions (continued)

Field	Description
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

debug aaa authentication

debug vpm all

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

[no] debug vpm all

Sample Displays

The **debug vpm all** EXEC 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

debug vpm dsp
debug vpm port
debug vpm signal
debug vpm spi

debug vpm dsp

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

[no] debug vpm dsp

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.

Sample Display

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:

```
vscm_dsp_message: MSG_TX_DTMF_DIGIT: digit=4
vscm_dsp_message: MSG_TX_DTMF_DIGIT: digit=1
vscm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vscm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vscm_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
vscm_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

Use the **debug vpm port** EXEC command to limit the debug output to a particular port. The **no** form of this command disables debugging output.

```
[no] debug vpm port slot-number/subunit-number/port
```

Syntax Description

<i>slot-number</i>	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 slot where it has been installed.
<i>subunit-number</i>	Specifies the subunit on the voice interface card where the voice port is located. Valid entries are 0 or 1.
<i>port</i>	Specifies the voice port. Valid entries are 0 or 1.

Usage Guidelines

Use the **debug vpm port** 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 with any or all of the other debug modes.

For example, the following shows **debug vpm dsp** messages only for port 1/0/0:

```
debug vpm dsp
debug vpm port 1/0/0
```

The following shows the **debug vpm signal** messages only for ports 1/0/0 and 1/0/1:

```
debug vpm signal
debug vpm port 1/0/0
debug vpm port 1/0/1
```

The following shows how to turn off debugging on a port:

```
no debug vpm port 1/0/0
```

The following produces no output because port level debugs work in conjunction with other levels:

```
debug vpm port 1/0/0
```

Execution of **no debug all** will turn off all port 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.

debug vpm signal

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

[no] debug vpm signal

Usage Guidelines

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

Sample Display

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** 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.

[no] debug vpm spi

Usage Guidelines

The **debug vpm spi** EXEC 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.

Sample Display

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
vcsn_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.

```
vcsn_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
vcsn_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.

```
vcsn_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
vcsn_timer: 47055
vcsn_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
vcsn_timer: 47079
vcsn_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
vcsn_timer: 47173
vcsn_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
vcsn_timer: 47197
vcsn_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
vcsn_timer: 47217
vcsn_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:

```
vscm_process_event: [1/0/1, 0.8, 15] act_bridge
vscm_process_event: [1/0/1, 0.8, 20] act_caps_ind
vscm_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:

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

The following output shows the voice quality statistics collected periodically:

```
vscm_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
vscm_process_event: [1/0/1, 0.13, 28]
vscm_process_event: [1/0/1, 0.13, 29]
vscm_process_event: [1/0/1, 0.13, 32]
vscm_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
vscm_process_event: [1/0/1, 0.13, 28]
vscm_process_event: [1/0/1, 0.13, 29]
vscm_process_event: [1/0/1, 0.13, 32]
vscm_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
vscm_process_event: [1/0/1, 0.13, 28]
vscm_process_event: [1/0/1, 0.13, 29]
vscm_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:

```
vscm_process_event: [1/0/1, 0.13, 4] act_generate_disc
vscm_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
vscm_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
vscm_timer: 48762
vscm_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
vscm_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
vscm_timer: 48762
dsp_set_sig_state: [1/0/1] packet_len=14 channel_id=129 packet_id=39 state=0x4
timestamp=0x0
vscm_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 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.

[no] debug vtemplate

Sample Displays

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 vtemplatel 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 encaps 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 190 describes the lines in the display.

Table 190 **Debug Vtemplate 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 encaps ppp ip unnumbered Ethernet0 no ip mroute-cache fair-queue 64 256 0 no cdp enable ppp authentication chap end	The specific configuration commands in virtual interface template 1 that are being applied to the virtual access interface 8.

Table 190 **Debug Vtemplate Field Descriptions (continued)**

Field	Description
%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.
%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	The access queue cleanup is proceeding and the template is being uncloned.
VTEMPLATE ***** UNCLONE VACCESS8 *****	
VTEMPLATE Unclone to-be-freed vaccess8 command#7	The specific configuration commands to be removed from the virtual access interface 8.
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 encaps ppp default ip address end	
VTEMPLATE set default vaccess8 with no ip address	The 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	The virtual access interface is added to the recycle queue.

debug x25

To display information about X.25 traffic, use one of the following **debug x25** 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. The generic forms of this command should be restricted to low-speed, low-usage links running below 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 packets per second.

To display information about all X.25 traffic, including traffic for X.25, CMNS, and XOT services, use the **debug x25 EXEC** command. The **no** form of this command disables debugging output.

```
[no] debug x25 [events | all]
```

To display information about a specific X.25 service class, use the following form of the **debug x25 EXEC** command:

```
[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 EXEC** command:

```
[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 EXEC** command:

```
[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 EXEC** 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:

```
[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 EXEC** command:

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

Use the **debug x25 EXEC** 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.

```
[no] debug x25 aodi
```

Syntax Description

events	(Optional) Displays all traffic except data and RR packets.
all	(Optional) Displays all traffic. This is the default.
only cmns xot	Displays information about the specified services: X.25 only, CMNS, or XOT.
aodi	Causes the debug x25 command to display AO/DI events and processing information.
<i>serial-interface</i>	X.25 serial interface.
<i>cmns-interface</i> mac mac-address	CMNS interface and remote host's MAC address. 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.

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 of the router's virtual circuits.

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 Receive Ready (RR) flow control packets; the normal flow of Data and RR packets is commonly large as well as 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, respectively, 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.

Sample Display

The following is sample output from an X.25 Restart event, a Call setup, data exchange, and Clear. The first two lines describe a Restart service exchange.

```
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 191 describes the fields in the display.

Table 191 Debug X25 Field Descriptions

Field	Description
Serial0	Interface on which the X.25 event occurred.
X25	Type of event this message describes.
I	Letter indicating whether the X.25 packet was input (I) or output (O) through the interface.

Table 191 Debug X25 Field Descriptions (continued)

Field	Description
R3	<p>State of the service or virtual circuit. Possible values follow:</p> <ul style="list-style-type: none"> • R/Inactive—Packet layer awaiting link layer service • R1—Packet layer ready • R2—DTE restart request • R3—DCE restart indication <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • D/Inactive—VC awaiting setup • D1—Flow control ready • D2—DTE reset request • D3—DCE reset indication <p>See Annex B of the 1984 ITU-T X.25 Recommendation for more information on these states.</p>
Restart	<p>The type of X.25 packet. Possible values follow:</p> <p>R Events</p> <ul style="list-style-type: none"> • Restart • Restart Confirm • Diagnostic <p>P Events</p> <ul style="list-style-type: none"> • Call • Call Confirm • Clear • Clear Confirm <p>D Events</p> <ul style="list-style-type: none"> • Reset • Reset Confirm <p>D1 Events</p> <ul style="list-style-type: none"> • Data • RNR (Receiver Not Ready) • RR (Receiver Ready) • Interrupt • Interrupt Confirm <p>XOT Overhead</p> <ul style="list-style-type: none"> • PVC Setup

Table 191 **Debug X25 Field Descriptions (continued)**

Field	Description
(5)	Number of bytes in the packet.
8	Modulo of the virtual circuit. Possible values are 8 or 128.
lci 0	Virtual circuit number. See Annex A of the X.25 Recommendation for information on virtual circuit 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 (<i>cause/diag</i>).

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
```

Related Commands

```
debug ppp bap
debug ppp bap negotiation
debug ppp multilink
debug ppp negotiation
```

debug x28

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

[no] debug x28

Sample Display

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 xns packet

Use the **debug xns packet** 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.

[no] debug xns packet

Usage Guidelines

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

Sample Display

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 192 describes significant fields in the display.

Table 192 Debug XNS Packet 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** EXEC command to display information on XNS routing transactions. The **no** form of this command disables debugging output.

[no] debug xns routing

Usage Guidelines

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

Sample Display

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 193 describes significant fields in the display.

Table 193 Debug XNS Routing Field Descriptions

Field	Description
XNSRIP:	This is an XNS routing packet.
sending standard periodic update to 5.ffff.ffff.ffff	Router indicates that this is a periodic XNS routing information update. 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 <ul style="list-style-type: none"> • 1—routing information • 2—echo • 3—router error