

debug v120 event

Use the **debug v120 event** EXEC command to display information on V.120 activity. The **no** form of this command disables debugging output.

[no] debug v120 event

Usage Guidelines

V.120 is an ITU specification that allows for reliable transport of synchronous, asynchronous, or bit transparent data over ISDN bearer channels.

For complete information on the V.120 process, use the **debug v120 packet** command along with the **debug v120 event** command. V.120 events are activity events rather than error conditions.

Sample Display

Figure 2-266 shows sample **debug v120 event** output of V.120 starting up and stopping. Also included is the interface that V.120 is running on (BR 0) and where the V.120 configuration parameters are obtained from (default).

Figure 2-266 Sample Debug V.120 Event Output

```
Router# debug v120 event

0:01:47: BR0:1-v120 started - Setting default V.120 parameters
0:02:00: BR0:1:removing v120
```

Related Command

debug v120 packet

debug v120 packet

Use the **debug v120 packet** EXEC command to display general information on all incoming and outgoing V.120 packets. The **no** form of this command disables debugging output.

[no] debug v120 packet

Usage Guidelines

The **debug v120 packet** command shows every packet on the V.120 session. You can use this information to determine whether incompatibilities exist between Cisco's V.120 implementation and other vendors' V.120 implementations.

V.120 is an ITU specification that allows for reliable transport of synchronous, asynchronous, or bit transparent data over ISDN bearer channels.

For complete information on the V.120 process, use the **debug v120 events** command along with the **debug v120 packet** command.

Sample Display

Figure 2-267 shows sample **debug v120 packet** output for a typical session startup.

Figure 2-267 Sample Debug V.120 Packet Output

```
Router# debug v120 packet

0:03:27: BR0:1: I SABME:11i 256 C/R 0 P/F=1
0:03:27: BR0:1: O UA:11i 256 C/R 1 P/F=1
0:03:27: BR0:1: O IFRAME:11i 256 C/R 0 N(R)=0 N(S)=0 P/F=0 len 43
0x83 0xD 0xA 0xD 0xA 0x55 0x73 0x65
0x72 0x20 0x41 0x63 0x63 0x65 0x73 0x73
0:03:27: BR0:1: I RR:11i 256 C/R 1 N(R)=1 P/F=0
0:03:28: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=0 P/F=0 len 2
0x83 0x63
0:03:28: BR0:1: O RR:11i 256 C/R 1 N(R)=1 P/F=0
0:03:29: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=1 P/F=0 len 2
0x83 0x31
0:03:29: BR0:1: O RR:11i 256 C/R 1 N(R)=2 P/F=0
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to up
0:03:31: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=2 P/F=0 len 2
0x83 0x55
0:03:32: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=3 P/F=0 len 3
0x83 0x31 0x6F
0:03:32: BR0:1: O RR:11i 256 C/R 1 N(R)=3 P/F=0
0:03:32: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=4 P/F=0 len 2
0x83 0x73
0:03:32: BR0:1: O RR:11i 256 C/R 1 N(R)=5 P/F=0
0:03:32: BR0:1: I IFRAME:11i 256 C/R 0 N(R)=1 N(S)=5 P/F=0 len 2
0x83 0xA
0:03:32: BR0:1: O IFRAME:11i 256 C/R 0 N(R)=6 N(S)=1 P/F=0 len 9
0x83 0xD 0xA 0x68 0x65 0x66 0x65 0x72 0x3E
```

Table 2-138 describes the fields shown in the display.

Table 2-138 Debug V.120 Packet Field Descriptions

Field	Descriptions
BR0:1	Interface number associated with this debugging information.
I/O	Packet going into or out of the interface.
SABME, UA, IFRAME, RR	V.120 packet type. In this case: <ul style="list-style-type: none"> • SABME—set asynchronous balanced mode, extended • US—unnumbered acknowledgment • IFRAME—information frame • RR—receive ready
lli 256	Logical link identifier number.
C/R 0	Command or response.
P/F=1	Poll final.
N(R)=0	Number received.
N(S)=0	Number sent.
len 43	Number of data bytes in the packet.
0x83	Up to 16 bytes of data.

Related Command

debug v120 event

debug vg-anylan

To monitor error information and 100VG connection activity, use the **debug vg-anylan EXEC** command. The **no** form of this command disables debugging output.

[no] debug vg-anylan

Usage Guidelines

This command could create large amounts of command output.

Sample Display

Figure 2-268 shows sample output from the **debug vg-anylan** command.

Figure 2-268 Sample Debug VG-AnyLAN Output

```
Router# debug vg-anylan

%HP100VG-5-LOSTCARR: HP100VG(2/0), lost carrier
```

Table 2-139 lists the possible messages that could be generated by this command.

Table 2-139 Debug VG-AnyLAN Message Descriptions

Message	Description	Action
%HP100VG-5-LOSTCARR: HP100VG(2/0), lost carrier	Lost carrier debug message. The VG controller detects that the link to the hub is down due to cable, hub, or VG controller problem.	Check, repair, or replace the cable or hub. If you determine that the cable and hub are functioning normally, repair or replace the 100VG-AnyLAN port adapter.
%HP100VG-5-CABLEERR: HP100VG(2/0), cable error, training failed	Bad cable error messages. Cable did not pass training. ¹	Check, repair, or replace the cable or hub. If you determine that the cable and hub are functioning normally, repair or replace the 100VG-AnyLAN port adapter.
%HP100VG-5-NOCABLE: HP100VG(2/0), no tone detected, check cable, hub	No cable attached error message. The VG MAC cannot hear tones from the hub. ¹	Check, repair, or replace the cable or hub. If you determine that the cable and hub are functioning normally, repair or replace the 100VG-AnyLAN port adapter.
HP100VG-1-FAIL: HP100VG(2/0), Training Fail - unable to login to the hub	Training to the VG network failed. Login to the hub rejected by the hub. ¹	Take action based on the following error messages: <ul style="list-style-type: none"> • %HP100VG-1-DUPMAC: HP100VG(2/0), A duplicate MAC address has been detected • HP100VG-1-LANCNF: HP100VG(2/0), Configuration is not compatible with the network • %HP100VG-1-ACCESS: HP100VG(2/0), Access to network is not allowed

Table 2-139 Debug VG-AnyLAN Message Descriptions (Continued)

Message	Description	Action
%HP100VG-1-DUPMAC: HP100VG(2/0), A duplicate MAC address has been detected	Duplicate MAC address on the same VG network. Two VG devices on the same LAN segment have the same MAC address.	Check the router configuration to make sure that no duplicate MAC address is configured.
%HP100VG-1-LANCF: HP100VG(2/0), Configuration is not compatible with the network	Configuration of the router is not compatible to the network.	Check that the configuration of the hub for Frame Format, Promiscuous, and Repeater bit indicates the proper configuration.
%HP100VG-1-ACCESS: HP100VG(2/0), Access to network is not allowed	Access to the VG network is denied by the hub.	Check the configuration of the hub.
%HP100VG-3-NOTHP100VG: Device reported 0x5101A	Could not find the 100VG PCI device on a 100VG-AnyLAN port adapter.	Make sure the 100VG-AnyLAN port adapter is properly seated in the slot. Otherwise repair or replace the 100VG-AnyLAN port adapter.
%HP100VG-1-DISCOVER: Only found 0 interfaces on bay 2, shutting down bay	No 100VG interface detected on a 100VG-AnyLAN port adapter in a slot.	Make sure the 100VG-AnyLAN port adapter is properly seated in the slot. Otherwise repair or replace the 100VG-AnyLAN port adapter.

1. This message might display when the total load on the cascaded hub is high. Wait at least 20 seconds before checking to see if the training really failed. Check if the protocol is up after 20 seconds before starting troubleshooting.

debug vines arp

Use the **debug vines arp** EXEC command to display debugging information on all Virtual Integrated Network Service (VINES) Address Resolution Protocol (ARP) packets that the router sends or receives. The **no** form of this command disables debugging output.

[no] debug vines arp

Sample Display

Figure 2-269 shows sample **debug vines arp** output.

Figure 2-269 Sample Debug VINES ARP Output

```
Router# debug vines arp

VNSARP: received ARP type 0 from 0260.8c43.a7e4
VNSARP: sending ARP type 1 to 0260.8c43.a7e4
VNSARP: received ARP type 2 from 0260.8c43.a7e4
VNSARP: sending ARP type 3 to 0260.8c43.a7e4 assigning address 3001153C:8004
VSARP: received ARP type 0 from 0260.8342.1501
VSARP: sending ARP type 1 to 0260.8342.1501
VSARP: received ARP type 2 from 0260.8342.1501
VSARP: sending ARP type 3 to 0260.8342.1501 assigning address 3001153C:8005,
sequence 143C, metric 2
```

In Figure 2-269, the first four lines show a non-sequenced ARP transaction and the second four lines show a sequenced ARP transaction. Within the first group of four lines, the first line shows that the router received an ARP request (type 0) from indicated station address 0260.8c43.a7e4. The second line shows that the router is sending back the ARP service response (type 1), indicating that it is willing to assign VINES Internet addresses. The third line shows that the router received a VINES Internet address assignment request (type 2) from address 0260.8c43.a7e4. The fourth line shows that the router is responding (type 3) to the address assignment request from the client and assigning it the address 3001153C:8004.

Within the second group of four lines, the sequenced ARP packet also includes the router' current sequence number and the metric value between the router and the client.

Table 2-140 describes significant fields shown in Figure 2-269.

Table 2-140 Debug VINES ARP Field Descriptions

Field	Description
VNSARP:	Banyan VINES nonsequenced ARP message.
VSARP:	Banyan VINES sequenced ARP message.
received ARP type 0	<p>ARP request of type 0 was received. Possible type values follow:</p> <ul style="list-style-type: none"> • 0—Query request. The ARP client broadcasts a type 0 message to request an ARP service to respond. • 1—Service response. The ARP service responds with a type 1 message to an ARP client's query request. • 2—Assignment request. The ARP client responds to a service response with a type 2 message to request a VINES Internet address. • 3—Assignment response. The ARP service responds to an assignment request with a type 3 message that includes the assigned VINES Internet address.

Table 2-140 Debug VINES ARP Field Descriptions (Continued)

Field	Description
from 0260.8c43.a7e4	Indicates the source address of the packet.

debug vines echo

Use the **debug vines echo** EXEC command to display information on all MAC-level echo packets that the router sends or receives. Banyan VINES interface testing programs make use of these echo packets. The **no** form of this command disables debugging output.

[no] debug vines echo

Note These echo packets do not include network layer addresses.

Sample Display

Figure 2-270 shows sample **debug vines echo** output.

Figure 2-270 Sample Debug VINES Echo Output

```
Router# debug vines echo

VINESECHO: 100 byte packet from 0260.8c43.a7e4
```

Table 2-141 describes the fields shown in Figure 2-270.

Table 2-141 Debug VINES Echo Field Descriptions

Field	Description
VINESECHO	Indication that this is a debug vines echo message.
100 byte packet	Packet size in bytes.
from 0260.8c43.a7e4	Source address of the echo packet.

debug vines ipc

Use the **debug vines ipc** EXEC command to display information on all transactions that occur at the VINES IPC layer, which is one of the two VINES transport layers. The **no** form of this command disables debugging output.

[no] debug vines ipc

Usage Guidelines

You can use the **debug vines ipc** command to discover why an IPC layer process on the router is not communicating with another IPC layer process on another router or Banyan VINES server.

Sample Display

Figure 2-271 shows sample **debug vines ipc** output for three pairs of transactions. For more information about these fields or their values, refer to Banyan VINES documentation.

Figure 2-271 Sample Debug VINES IPC Output

```
Router# debug vines ipc

VIPC: sending IPC Data to Townsaver port 7 from port 7
      r_cid 0, l_cid 1, seq 1, ack 0, length 12
VIPC: received IPC Data from Townsaver port 7 to port 7
      r_cid 51, l_cid 1, seq 1, ack 1, length 32
VIPC: sending IPC Ack to Townsaver port 0 from port 0
      r_cid 51, l_cid 1, seq 1, ack 1, length 0
```

Table 2-142 describes the fields shown in Figure 2-271.

Table 2-142 Debug VINES IPC Field Descriptions

Field	Description
VIPC:	Indicates that this is output from the debug vines ipc command.
sending	Indicates that the router is either sending an IPC packet to another router or has received an IPC packet from another router.
IPC Data to	Indicates the type of IPC frame: <ul style="list-style-type: none"> • Acknowledgment • Data • Datagram • Disconnect • Error • Probe
Townsaver port 7	Indicates the machine name as assigned using the VINES host command, or IP address of the other router. Also indicates the port on that machine through which the packet has been transmitted.
from port 7	Indicates the port on the router through which the packet has been transmitted.
r_cid 0, l_cid 1, seq 1, ack 0, length 12	Indicates the values for various fields in the IPC layer header of this packet. Refer to Banyan VINES documentation for more information.

debug vines netrpc

Use the **debug vines netrpc** EXEC command to display information on all transactions that occur at the VINES NetRPC layer, which is the VINES Session/Presentation layer. The **no** form of this command disables debugging output.

[no] debug vines netrpc

Usage Guidelines

You can use the **debug vines netrpc** command to discover why a NetRPC layer process on the router is not communicating with another NetRPC layer process on another router or Banyan server.

Sample Display

Figure 2-272 shows sample **debug vines netrpc** output. For more information about these fields or their values, refer to Banyan VINES documentation.

Figure 2-272 Sample Debug VINES NetRPC Output

```
Router# debug vines netrpc

VRPC: sending RPC call to Townsaver
VRPC: received RPC return from Townsaver
```

Table 2-143 describes the fields shown in the first line of output in Figure 2-272.

Table 2-143 Debug VINES NetRPC Field Descriptions

Field	Description
VRPC:	Indicates that this is output from the debug vines netrpc command.
sending RPC call	Indicates that the router is either sending a NetRPC packet to another router or has received a NetRPC packet from another router.
Townsaver	Indicates the transaction type: <ul style="list-style-type: none"> • abort • call • reject • return • return address • search • search all
	Indicates the machine name as assigned using the VINES host command or IP address of the other router.

debug vines packet

Use the **debug vines packet** EXEC command to display general VINES debugging information. This information includes packets received, generated, and forwarded, as well as failed access checks and other operations. The **no** form of this command disables debugging output.

[no] debug vines packet

Sample Display

Figure 2-273 shows sample **debug vines packet** output.

Figure 2-273 Sample Debug VINES Packet Output

```
Router# debug vines packet

VINES: s=30028CF9:1 (Ether2), d=FFFFFFFF:FFFF, rcvd w/ hops 0
VINES: s=3000CBD4:1 (Ether1), d=3002ABEA:1 (Ether2), g=3002ABEA:1, sent
VINES: s=3000CBD4:1 (Ether1), d=3000B959:1, rcvd by gw
VINES: s=3000B959:1 (local), d=3000CBD4:1 (Ether1), g=3000CBD4:1, sent
```

Table 2-144 describes the fields shown in the first line of output.

Table 2-144 Debug VINES Packet Field Descriptions

Field	Description
VINES:	Indicates that this is a Banyan VINES packet.
s = 30028CF9:1 (Ether2)	Indicates source address of the packet. Indicates the interface through which the packet was received.
d = FFFFFFFF:FFFF	Indicates that the destination is a broadcast address.
rcvd w/ hops 0	Indicates that the packet was received because it was a local broadcast packet. The remaining hop count in the packet was zero (0).

Explanations for other lines in Figure 2-273 follow.

In the following line, the destination is the address 3002ABEA:1 associated with interface Ether2. Source address 3000CBD4:1 sent a packet to this destination through the gateway at address 3000ABEA:1.

```
VINES: s=3000CBD4:1 (Ether1), d=3002ABEA:1 (Ethernet2), g=3002ABEA:1, sent
```

In the following line, the router being debugged is the destination address (3000B959:1):

```
VINES: s=3000CBD4:1 (Ether1), d=3000B959:1, rcvd by gw
```

In the following line, (local) indicates that the router being debugged generated the packet:

```
VINES: s=3000B959:1 (local), d=3000CBD4:1 (Ether1), g=3000CBD4:1, sent
```

debug vines routing

Use the **debug vines routing** EXEC command to display information on all VINES RTP update messages sent or received and all routing table activities that occur in the router. The **no** form of this command disables debugging output.

[no] debug vines routing [verbose]

Syntax Description

verbose (Optional) Provides detailed information about the contents of each update.

Sample Displays

Figure 2-274 shows sample **debug vines routing** output.

Figure 2-274 Sample Debug VINES Routing Output

```

router# debug vines routing
VSRTTP: generating change update, sequence number 0002C791
Update sent VSRTTP: sent update to Broadcast on Hssi0
Update received VSRTTP: received update from LabRouter on Hssi0
VSRTTP: LabRouter-Hs0-HDLC up -> up, change update, onemore
VRTTP: sending update to Broadcast on Ethernet0
VSRTTP: generating null update
VSRTTP: Sending update to Aloe on Hssi0

```

S2854

Figure 2-275 shows sample **debug vines routing verbose** output.

Figure 2-275 Sample Debug VINES Routing Verbose Output

```

Router# debug vines routing verbose

VRTTP: sending update to Broadcast on Ethernet0
network 30011E7E, metric 0020 (0.4000 seconds)
network 30015800, metric 0010 (0.2000 seconds)
network 3003148A, metric 0020 (0.4000 seconds)
VSRTTP: generating change update, sequence number 0002C795
network Router9 metric 0010, seq 00000000, flags 09
network RouterZZ metric 0230, seq 00052194, flags 02
VSRTTP: sent update to Broadcast on Hssi0
VSRTTP: received update from LabRouter on Hssi0
update: type 00, flags 07, id 000E, ofst 0000, seq 15DFC, met 0010
network LabRouter from the server
network Router9 metric 0020, seq 00000000, flags 09
VSRTTP: LabRouter-Hs0-HDLC up -> up, change update, onemore

```

Figure 2-275 describes two VINES routing updates; the first includes two entries and the second includes three entries. Explanations for selected lines of Figure 2-275 follow.

The following line shows that the router sent a periodic routing update to the broadcast address FFFFFFFF:FFFF through the Ethernet0 interface:

```
VRTTP: sending update to Broadcast on Ethernet0
```

The following line indicates that the router knows how to reach network 30011E7E, which is a metric of 0020 away from the router. The value that follows the metric (0.4000 seconds) interprets the metric in seconds.

```
network 30011E7E, metric 0020 (0.4000 seconds)
```

The following lines show that the router sent a change routing update to the Broadcast addresses on the Hssi0 interface using the Sequenced Routing Update Protocol (SRTP) routing protocol:

```
VS RTP: generating change update, sequence number 0002C795  
VS RTP: Sending update to Broadcast on Hssi0
```

The lines in between the previous two indicate that the router knows how to reach network Router9, which is a metric of 0010 (0.2000 seconds) away from the router. The sequence number for Router9 is zero, and according to the 0x08 bit in the flags field, is invalid. The 0x01 bit of the flags field indicates that Router9 is attached via a LAN interface.

```
network Router9          metric 0010, seq 00000000, flags 09
```

The next lines indicate that the router can reach network RouterZZ, which is a metric of 0230 (7.0000 seconds) away from the router. The sequence number for RouterZZ is 0052194. The 0x02 bit of the flags field indicates that RouterZZ is attached via a WAN interface.

```
network RouterZZ        metric 0230, seq 00052194, flags 02
```

The following line indicates that the router received a routing update from the router LabRouter through the Hssi0 interface:

```
VS RTP: received update from LabRouter on Hssi0
```

The following line displays all SRTP values contained in the header of the SRTP packet. This is a type 00 packet, which is a routing update, and the flags field is set to 07, indicating that this is a change update (0x04) and contains both the beginning (0x01) and end (0x02) of the update. This overall update is update number 000E from the router, and this fragment of the update contains the routes beginning at offset 0000 of the update. The sending router's sequence number is currently 00015DFC, and its configured metric for this interface is 0010.

```
update: type 00, flags 07, id 000E, ofst 0000, seq 00015DFC, met 0010
```

The following line implies that the server sending this update is directly accessible to the router (even though VINES servers do not explicitly list themselves in routing updates). Because this is an implicit entry in the table, the other information for this entry is taken from the previous line.

```
network LabRouter from the server
```

As the first actual entry in the routing update from LabRouter, the following line indicates that Router9 can be reached by sending to this server. This network is a metric of 0020 away from the sending server.

```
network Router9          metric 0020, seq 00000000, flags 09
```

debug vines service

Use the **debug vines service EXEC** command to display information on all transactions that occur at the VINES Service (or applications) layer. The **no** form of this command disables debugging output.

[no] debug vines service

Usage Guidelines

You can use the **debug vines service** command to discover why a VINES Service layer process on the router is not communicating with another Service layer process on another router or Banyan server.

Note Because the **debug vines service** command provides the highest level overview of VINES traffic through the router, it is best to begin debugging using this command, and then proceed to use lower-level VINES **debug** commands as necessary.

Sample Display

Figure 2-276 shows sample **debug vines service** output.

Figure 2-276 Sample Debug VINES Service Output

```
router# debug vines service
```

Sent/ Response pair	<pre>VSRV: Get Time Info sent to Townsaver VSRV: Get Time Info response from Townsaver, time: 01:47:54 PDT Apr 29 1993 VSRV: epoch SS@Aloe@Servers-10, age: 0:15:15</pre>	S2565
---------------------------	---	-------

As Figure 2-276 suggests, **debug vines service** lines of output appear as activity pairs—either a sent/response pair as shown, or as a received/sent pair.

Table 2-145 describes the fields shown in the second line of output in Figure 2-276. For more information about these fields or their values, refer to Banyan VINES documentation.

Table 2-145 Debug VINES Service Field Descriptions—Part 1

Field	Description
VSRV:	Indicates that this is output from the debug vines service command.
Get Time Info	Indicates one of three packet types: <ul style="list-style-type: none"> Get Time Info Time Set Time Sync
response from	Indicates whether the packet was sent to another router, a response from another router, or received from another router.
Townsaver	Indicates the machine name as assigned using the VINES host command, or IP address of the other router.
time: 01:47:54 PDT Apr 29 1993	Indicates the current time in hours:minutes:seconds and current date.

Table 2-146 describes the fields shown in the third line of output in Figure 2-276. This line is an extension of the first two lines of output. For more information about these fields or their values, refer to Banyan VINES documentation.

Table 2-146 Debug VINES Service Field Descriptions—Part 2

Field	Description
VSRV:	Output from the debug vines service command.
epoch	Line of output that describes a VINES epoch.
SS@Aloe@Servers-10	Epoch name.
age: 0:15:15	Epoch—elapsed time since the time was last set in the network.

debug vines state

Use the **debug vines state** EXEC command to display information on the VINES SRTP state machine transactions. The **no** form of this command disables debugging output.

[no] debug vines state

Usage Guidelines

This command provides a subset of the information provided by the **debug vines routing** command, showing only the transactions made by the SRTP state machine. Refer to the **debug vines routing** command for descriptions of output from the **debug vines state** command.

debug vines table

Use the **debug vines table** EXEC command to display information on all modifications to the VINES routing table. The **no** form of this command disables debugging output.

[no] debug vines table

Usage Guidelines

This command provides a subset of the information produced by the **debug vines routing** command, as well as some more detailed information on table additions and deletions.

Sample Display

Figure 2-277 shows sample **debug vines table** output.

Figure 2-277 Sample Debug VINES Table Output

```
Router# debug vines table  
  
VINESRTP: create neighbor 3001153C:8004, interface Ethernet0
```

Table 2-147 describes significant fields shown in Figure 2-277.

Table 2-147 Debug VINES Table Field Descriptions

Field	Description
VINESRTP:	Indicates that this is a debug vines routing or debug vines table message.
create neighbor 3001153C:8004	Indicates that the client at address 3001153C:8004 has been added to the Banyan VINES neighbor table.
interface Ethernet 0	Indicates that this neighbor can be reached through the router interface named Ethernet0.

debug vlan packet

Use the **debug vlan packet** EXEC command to display general information on virtual LAN (VLAN) packets that the router received but is not configured to support. The **no** form of this command disables debugging output.

[no] debug vlan packet

Usage Guidelines

The **debug vlan packet** command displays only packets with a VLAN identifier that the router is not configured to support. This command allows you to identify other VLAN traffic on the network. Virtual LAN packets that the router is configured to route or switch are counted and indicated when you use the **show vlans** command.

Sample Display

Figure 2-278 shows sample **debug vlan packet** output. In this example, a VLAN packet with a VLAN ID of 1000 was received on FDDI 0 interface and this interface was not configured to route or switch this VLAN packet.

Figure 2-278 Sample Debug VLAN Packet Output

```
Router# debug vlan packet

vLAN: IEEE 802.10 packet bearing vLAN ID 1000 received on interface
      Fddi0 which is not configured to route/switch ID 1000.
```

debug vpdn

To display debug traces for the Virtual Private Dialup Network (VPDN) feature, which provides PPP tunnels using the Layer 2 Forwarding (L2F) protocol, use the **debug vpdn** EXEC command. The **no** form of this command disables debugging output.

```
[no] debug vpdn {errors | events | packets | l2f-errors | l2f-events | l2f-packets}
```

Syntax Description

errors	Displays errors that prevent a tunnel from being established or errors that cause an established tunnel to be closed.
events	Displays messages about events that are part of normal tunnel establishment or shutdown.
packets	Displays each protocol packet exchanged. This option may result in a large number of debug messages and should generally only be used on a debug chassis with a single active session.
l2f-errors	Displays L2F protocol errors that prevent L2F establishment or prevent its normal operation.
l2f-events	Displays messages about events that are part of normal tunnels establishment or shutdown for L2F.
l2f-packets	Displays messages about L2F protocol headers and status.

Debug VPDN Events on a Network Access Server—Normal Operations

The network access server has the following VPDN configuration:

```
vpdn outgoing cisco.com stella ip 172.21.9.26
username stella password stella
```

Figure 2-279 shows sample output of the **debug vpdn events** command on a network access server when the L2F tunnel is brought up and CHAP authentication of the tunnel succeeds.

Figure 2-279 Debug VPDN Events on the Network Access Server—Tunnel Coming Up

```
Router# debug vpdn events

%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 bum6@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 stella.
```

Figure 2-280 shows sample output of the **debug vpdn events** command on a network access server when the L2F tunnel is brought down normally.

Figure 2-280 Debug VPDN Events on the Network Access Server—Tunnel Coming Down Normally

```
Router# debug vpdn events

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:27:18.865: 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 2-148 describes the fields in Figure 2-279 and Figure 2-280. The output describes normal operations when a tunnel is brought up or down on a network access server.

Table 2-148 Debug VPDN Events Field Descriptions for the Network Access Server

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.
Async6 VPN Bind interface direction=1	Tunnel is bound to the interface. These are the direction values: 1—From the network access server to the home gateway 2—From the home gateway to the network access server
Async6 VPN vpn_forward_user bum6@cisco.com is forwarded	Tunnel for the specified user and domain name (bum6@cisco.com) is forwarded.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol is up.
L2F: Chap authentication succeeded for stella.	Tunnel was authenticated with the tunnel password <i>stella</i> .
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.

Debug VPDN Events on the Home Gateway—Normal Operations

The home gateway has the following VPDN configuration, which uses *stella* as the tunnel name and the tunnel authentication name. The tunnel authentication name might be entered in a users file on an AAA server and used to define authentication requirements for the tunnel.

```
vpdn incoming stella stella virtual-template 1
```

Figure 2-281 shows sample output of the **debug vpdn events** command on the home gateway when the tunnel is brought up successfully.

Figure 2-281 Debug VPDN Events on the Home Gateway—Tunnel Coming Up

```
Router# debug vpdn events

L2F: Chap authentication succeeded for stella.
Virtual-Access3 VPN Virtual interface created for bum6@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
```

Figure 2-282 shows sample output of the **debug vpdn events** command on a home gateway when the tunnel is brought down normally.

Figure 2-282 Debug VPDN Events on the Home Gateway—Tunnel Coming Down Normally

```
Router# debug vpdn events

%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 2-149 describes the fields in Figure 2-281 and Figure 2-282. The output describes normal operations when a tunnel is brought up or down on a network access server.

Table 2-149 Debug VPDN Events Field Descriptions for Home Gateway

Field	Description
Tunnel Coming Up	
L2F: Chap authentication succeeded for stella.	PPP CHAP authentication status for the tunnel named <i>stella</i> .
Virtual-Access3 VPN Virtual interface created for bum6@cisco.com	Virtual access interface was set up on the home gateway for the user bum6@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.

Table 2-149 Debug VPDN Events Field Descriptions for Home Gateway (Continued)

Field	Description
Virtual-Access3 VPN Bind interface direction=2	Tunnel is bound to the interface. These are the direction values: 1—From the network access server to the home gateway 2—From the home gateway to the network access server
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK	PPP LCP configuration settings (negotiated between the remote client and the network access server) 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.

Debug VPDN L2F-Events on the Network Access Server—Normal Operations

Figure 2-283 shows sample output of the **debug vpdn l2f-events** command on the network access server when the L2F tunnel is brought up successfully.

Figure 2-283 Debug VPDN L2F-Events on the Network Access Server—Tunnel Coming Up

```
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
```

Figure 2-284 shows sample output of the **debug vpdn l2f-events** command on a network access server when the tunnel is brought down normally.

Figure 2-284 Debug VPDN L2F-Events on the Network Access Server—Tunnel Coming Down

```
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 2-150 describes the fields in Figure 2-283 and Figure 2-284.

Table 2-150 Debug VPDN L2F-Events Fields—Network Access Server

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 UDP socket to the home gateway IP address.
L2F_CONF received	The L2F_CONF signal was received. When sent from the home gateway to the network access server, an L2F_CONF indicates the home gateway's recognition of the tunnel creation request.
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	The L2F_OPEN management message was received, indicating that home gateway accepted the network access server configuration of an L2F tunnel.
L2F building nas2gw_mid0	L2F is building a tunnel between the network access server and the home gateway, using MID 0.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	The line protocol came up. Indicates whether the software processes that handle the line protocol regard the interface as usable.
L2F_OPEN received	The L2F_OPEN management message was received, indicating that home gateway accepted the network access server configuration of an L2F tunnel.

Table 2-150 Debug VPDN L2F-Events Fields—Network Access Server (Continued)

Field	Descriptions
L2F Got a MID management packet	Multiplex ID (MID) management packets are used to communicate between the network access server and the home gateway.
L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6	L2F synchronized the Client IDs on the network access server 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	The 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	The network access server received a request to close the tunnel.
L2F Destroying mid	The connection identified by the MID is begin taken down.
L2F Tunnel is going down!	Advisory message about impending tunnel shutdown.
L2F Initiating tunnel shutdown.	Tunnel shutdown has started.
L2F_CLOSE received	The network access server received a request to close the tunnel.
L2F Got closing for tunnel	The network access server began tunnel closing operations.
%LINK-3-UPDOWN: Interface Async6, changed state to down	The asynchronous interface was taken down.
L2F Closed tunnel structure	The network access server closed the tunnel.
L2F Deleted inactive tunnel	The now-inactivated tunnel was deleted.

Debug VPDN L2F-Events on the Home Gateway—Normal Operations

Figure 2-285 shows sample output of the **debug vpdn l2f-events** command on a home gateway when the L2F tunnel is created.

Figure 2-285 Debug VPDN L2F-Events on the Home Gateway—Tunnel is Created

```

Router# debug vpdn l2f-events

L2F_CONF received
L2F Creating new tunnel for stella
L2F Got a tunnel named stella, 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

```

Figure 2-286 shows sample output of the **debug vpdn l2f-events** command on a home gateway when the L2F tunnel is brought down normally.

Figure 2-286 Debug VPDN L2F-Events on a Home Gateway—Tunnel Coming 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 2-151 describes the fields in Figure 2-285 and Figure 2-286.

Table 2-151 Debug VPDN L2F-Events Field Descriptions—Home Gateway

Field	Description
Tunnel Coming Up	
L2F_CONF received	L2F configuration is received from the network access server. When sent from a network access server to a home gateway, the L2F_CONF is the initial packet in the conversation.
L2F Creating new tunnel for stella	The tunnel named <i>stella</i> is being created.
L2F Got a tunnel named stella, responding	Home gateway is responding.
L2F Open UDP socket to 172.21.9.25	Opening a socket to the network access server IP address.
L2F_OPEN received	The L2F_OPEN management message was received, indicating the network access server is opening an L2F tunnel.
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 Got a MID management packet	L2F MID management packets are used to communicate between the network access server and the home gateway.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up	The 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	The line protocol is up. The line can be used.

Table 2-151 Debug VPDN L2F-Events Field Descriptions—Home Gateway (Continued)

Field	Description
Tunnel Coming Down	
L2F_CLOSE received	The network access server or home gateway received a request to close the tunnel.
L2F Destroying mid	The connection identified by the MID is begin 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.

Debug VPDN on the Network Access Server—Error Conditions

Figure 2-287 shows sample output of the **debug vpdn errors** command on a network access server when the tunnel is not set up.

Figure 2-287 Debug VPDN Errors on the Network Access Server—Error Conditions

```
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 2-152 describes the fields in Figure 2-287.

Table 2-152 Debug VPDN Errors Fields Descriptions for the Network Access Server

Field	Description
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down	The 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	The 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	The line protocol on the asynchronous interface came back up.
VPDN tunnel management packet failed to authenticate	Tunnel authentication failed. This the most common VPDN error. Note: Check the password for the network access server and the home gateway name. If you store the password on an AAA server, you can use the debug aaa authentication command.

Figure 2-288 shows sample output of the **debug vpdn l2f-errors** command.

Figure 2-288 Debug VPDN L2F-Errors on the Network Access Server—Error Conditions

```
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 home.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 2-153 describes the fields in Figure 2-288.

Table 2-153 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 home.com	Tunnel was established from the network access server to the home gateway, home.com.
L2F Received a close request for a non-existent mid	Multiplex ID was not used previously; cannot close the tunnel.

Table 2-153 Debug VPDN L2F-Errors Field Descriptions (Continued)

Field	Description
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.

Debugging VPDN Packets for Complete Information

Figure 2-289 shows sample output of the **debug vpdn l2f-packets** command on a network access server. This example displays a trace for a **ping** command.

Figure 2-289 Debug VPDN L2F-Packets on a Network Access Server

```

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 Otput 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 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
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 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 2-154 describes the fields in Figure 2-289.

Table 2-154 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	The protocol for negotiation of the point-to-point link between the network access server 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 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 present, and payload. Length does not reflect the addition of the checksum, if present.
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 ppp authentication
debug ppp negotiations
debug ppp error

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

Figure 2-290 shows sample output from the **debug vtemplate** command when a virtual access interface come up. The virtual access interface is cloned from virtual template 1.

Figure 2-290 Sample Debug Vtemplate Output—Virtual Interface Coming Up

```
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
```

Figure 2-291 shows 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.

Figure 2-291 Sample Debug Vtemplate Output—Virtual Interface Going Down

```
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 2-155 explains the lines in Figure 2-290 and Figure 2-291.

Table 2-155 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.
%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 VTEMPLATE ***** UNCLONE VACCESS8 *****	The access queue cleanup is proceeding and the template is being uncloned.

Table 2-155 Debug Vtemplate Field Descriptions (Continued)

Field	Description
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	The specific configuration commands to be removed from the virtual access interface 8.
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. To disable debugging output, use the **no** form of this command.

[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]

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.
<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.)

Sample Display

Figure 2-292 shows the debug output for an X.25 Restart event, a Call setup, data exchange, and Clear. The first two lines describe a Restart service exchange.

Figure 2-292 Sample Debug X25 Output

```
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 2-156 describes the fields in Figure 2-292.

Table 2-156 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 2-156 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 • 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 <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 2-156 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>causeddiag</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

```

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

Figure 2-293 shows sample output while the PAD initiates an X.28 outgoing call.

Figure 2-293 Sample Debug X28 Output

```
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

Figure 2-294 shows sample **debug xns packet** output.

Figure 2-294 Sample Debug XNS Packet Output.

```
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 2-157 describes significant fields shown in Figure 2-294.

Table 2-157 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 in Figure 2-294, as 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

Figure 2-295 shows sample **debug xns routing** output.

Figure 2-295 Sample Debug XNS Routing Output

```
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 2-158 describes significant fields shown in Figure 2-295.

Table 2-158 Debug XNS Routing Field Descriptions

Field	Description
XNSRIP:	This is an XNS routing packet.
sending standard periodic update to 5.ffff.ffff.ffff via Ethernet2	Router indicates that this is a periodic XNS routing information update. Destination address is ffff.ffff.ffff on network 5. 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 socket 1	Router indicates that it has received an XNS routing information update from address 0000.0c00.440f on network 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