

## debug ipx ipxwan

Use the **debug ipx ipxwan EXEC** command to display debug information for interfaces configured to use IPXWAN. The **no** form of this command disables debugging output.

**[no] debug ipx ipxwan**

### Usage Guidelines

The **debug ipx ipxwan** command is useful for verifying the startup negotiations between two routers running the IPX protocol through a WAN. This command produces output only during state changes or startup. During normal operations, no output is produced.

### Sample Display

Figure 2-133 shows sample **debug ipx ipxwan** output during link startup.

**Figure 2-133 Sample Debug IPX IPXWAN Output**

```
Router# debug ipx ipxwan

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial11, changed state to up
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial11/6666:200 (IPX line
state brought up)]
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial11/6666:200 (IPX line
state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial11/6666:200 (IPX line
state brought up)]

IPXWAN: Send TIMER_REQ [seq 0] out Serial11/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial11/6666:200
IPXWAN: Send TIMER_REQ [seq 2] out Serial11/6666:200
IPXWAN: Send TIMER_REQ [seq 0] out Serial11/6666:200

IPXWAN: Rcv TIMER_REQ on Serial11/6666:200, NodeID 1234, Seq 1
IPXWAN: Send TIMER_REQ [seq 1] out Serial11/6666:200
IPXWAN: Rcv TIMER_RSP on Serial11/6666:200, NodeID 1234, Seq 1, Del 6
IPXWAN: state (Sending Timer Requests -> Master: Sent RIP/SAP) [Serial11/6666:200
(Received Timer Response as master)]
IPXWAN: Send RIPSAP_INFO_REQ [seq 0] out Serial11/6666:200
IPXWAN: Rcv RIPSAP_INFO_RSP from Serial11/6666:200, NodeID 1234, Seq 0
IPXWAN: state (Master: Sent RIP/SAP -> Master: Connect) [Serial11/6666:200 (Received
Router Info Rsp as Master)]
```

Explanations for representative lines of output in Figure 2-133 follow.

The following line indicates that the interface has initialized:

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial11, changed state to up
```

The following lines indicate that the startup process failed to receive a timer response, brought the link down, then brought the link up and tried again with a new timer set:

```
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial11/6666:200 (IPX line
state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial11/6666:200 (IPX line
state brought up)]
```

The following lines indicate that the interface is sending timer requests and waiting on timer response:

```
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
```

The following lines indicate that the interface has received a timer request from the other end of the link and has sent a timer response. The fourth line shows that the interface has come up as the master on the link.

```
IPXWAN: Rcv TIMER_REQ on Serial1/6666:200, NodeID 1234, Seq 1
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Rcv TIMER_RSP on Serial1/6666:200, NodeID 1234, Seq 1, Del 6
IPXWAN: state (Sending Timer Requests -> Master: Sent RIP/SAP) [Serial1/6666:200
(Received Timer Response as master)]
```

The following lines indicate that the interface is sending RIP/SAP requests:

```
IPXWAN: Send RIPSAP_INFO_REQ [seq 0] out Serial1/6666:200
IPXWAN: Rcv RIPSAP_INFO_RSP from Serial1/6666:200, NodeID 1234, Seq 0
IPXWAN: state (Master: Sent RIP/SAP -> Master: Connect) [Serial1/6666:200 (Received
Router Info Rsp as Master)]
```

## debug ipx nasi

Use the **debug ipx nasi** EXEC command to display information about the NetWare Asynchronous Services Interface (NASI) connections. The **no** form of this command disables debugging output.

**[no] debug ipx nasi {packets | error | activity}**

### Syntax Description

<b>packets</b>	Displays normal operating messages relating to incoming and outgoing NASI packets. This is the default.
<b>error</b>	Displays messages indicating an error or failure in the protocol processing.
<b>activity</b>	Displays messages relating to internal NASI processing of NASI connections. The <b>activity</b> option includes all NASI activity such as traffic indication, timer events, and state changes.

### Usage Guidelines

Use the **debug ipx nasi** command to display handshaking or negotiating details between the protocol (SPX or NASI) and the other protocols or applications. Use the **packets** option to determine the NASI traffic flow, and use the **error** option as a quick check of failure reasons in NASI connections.

### Sample Display

Figure 2-134 shows sample **debug ipx nasi** output of the **packet** and **error** options.

**Figure 2-134 Sample Debug IXP NASI Output**

```
Router# debug ipx nasi packet
Router# debug ipx nasi error

NASI0: 6E6E Check server info
NASI0: 6E6E sending server-info 4F00 Good response: 43 bytes
NASI0: 7A6E Query Port. Find first
NASI0: FFirst: line 0 DE, port: TTY1-_____ASYNC___^, group: ASYNC___^
NASI0: 7A6E sending Qport find-first response: 300 bytes
NASI0: 7B6E port request. setting up port
NASI: Check-login User: c h r i s
NASI: Check-login PW hash: C7 A6 C5 C7 C4 C0 C5 C3 C4 CC C5 CF C4 C8 C5 CB C4 D4 C5 D7
C4 D0 C5 D3 C4
NASI: Check-login PW: 1 a b
NASI1: 7B6E sending NCS Good server Data Ack in 0 bytes pkt in 13 size pkt
NASI1: 7B6E sending Preq response: 303 bytes Good
NASI1: 7B6E port request. setting up port
NASI1: 7B6E sending NCS Good server Data Ack in 0 bytes pkt in 13 size pkt
NASI1: 7B6E sending Preq response: 303 bytes Good
NASI1: 7B6E Unknown NASI code 4500 Pkt Size: 13
45 0 0 FC 0 2 0 20 0 0 FF 1 0
NASI1: 7B6E Flush Rx Buffers
NASI1: 7B6E sending NASI server TTY data: 1 byte in 14 size pkt
NASI1: 7B6E sending NCS Good server Data Ack in 1 bytes pkt in 13 size pkt
```

Explanations of representative lines of the output in Figure 2-134 follow.

In the following line, the 0 in *NASI0* is the number of the terminal (TTY) to which this NASI connection is attached. TTY 0 is used by all NASI control connections. 6E6E is the associated SPX connection pointer for this NASI connection. *Check server info* is a type of NASI packet that indicates an incoming NASI packet of this type.

```
NASI0: 6E6E Check server info
```

The following message indicates the router is sending back a *server-info* packet with a positive acknowledgment, and the packet size is 43 bytes:

```
NASI0: 6E6E sending server-info 4F00 Good response: 43 bytes
```

The following line is a NASI packet type. *Find first* and *find next* are NASI packet types.

```
NASI0: 7A6E Query Port. Find first
```

The following line indicates that the outgoing find first packet for the NASI connection 7A6E has line 0 DE, port name TTY1, and general name ASYNC:

```
NASI0: Ffirst: line 0 DE, port: TTY1-_____ASYNC___^, group: ASYNC___^
```

The following two lines indicate a received NASI packet for NASI connection on line 1. 7B6E is the NASI connection pointer. The packet code is 4500 and is not recognizable by Cisco. The second line is a hexadecimal dump of the packet.

```
NASI1: 7B6E Unknown NASI code 4500 Pkt Size: 13  
45 0 0 FC 0 2 0 20 0 0 FF 1 0
```

## Related Command

**debug ipx spx**

## debug ipx packet

Use the **debug ipx packet** EXEC command to display information about packets received, transmitted, and forwarded. The **no** form of this command disables debugging output.

**[no] debug ipx packet**

### Usage Guidelines

This command is useful for learning whether IPX packets are traveling over a router.

---

**Note** In order to generate **debug ipx packet** information on all IPX traffic traveling over the router, you must first configure the router so that fast switching is disabled. Use the **no ipx route-cache** command on all interfaces on which you want to observe traffic. If the router is configured for IPX fast switching, only non-fast switched packets will produce output. When the IPX cache is invalidated or cleared, one packet for each destination is displayed as the cache is repopulated.

---

### Sample Display

Figure 2-135 shows sample **debug ipx packet** output.

**Figure 2-135 Sample Debug IPX Packet Output**

```
Router# debug ipx packet
IPX: src=160.0260.8c4c.4f22, dst=1.0000.0000.0001, packet received
IPX: src=160.0260.8c4c.4f22, dst=1.0000.0000.0001,gw=183.0000.0c01.5d85,
sending packet
```

In Figure 2-135, the first line indicates that the router receives a packet from a Novell station (address 160.0260.8c4c.4f22); this trace does not indicate the address of the immediate router sending the packet to this router. In the second line, the router forwards the packet toward the Novell server (address 1.0000.0000.0001) through an immediate router (183.0000.0c01.5d85).

Table 2-72 describes significant fields shown in Figure 2-135.

**Table 2-72 Debug IPX Packet Field Descriptions**

Field	Description
IPX	Indication that this is an IPX packet.
src = 160.0260.8c4c.4f22	Source address of the IPX packet. The Novell network number is 160. Its MAC address is 0260.8c4c.4f22.
dst = 1.0000.0000.0001	Destination address for the IPX packet. The address 0000.0000.0001 is an internal MAC address, and the network number 1 is the internal network number of a Novell 3.11 server.
packet received	The router received this packet from a Novell station, possibly through an intermediate router.
gw = 183.0000.0c01.5d85	The router is sending the packet over to the next hop router; its address of 183.0000.0c01.5d85 was learned from the IPX routing table.
sending packet	The router is attempting to send this packet.

## debug ipx routing

Use the **debug ipx routing** EXEC command to display information on IPX routing packets that the router sends and receives. The **no** form of this command disables debugging output.

```
[no] debug ipx routing {activity | events}
```

### Syntax Description

**activity** Displays messages relating to IPX routing activity.

**events** Displays messages relating to IPX routing events.

### Usage Guidelines

Normally, a router or server sends out one routing update per minute. Each routing update packet can include up to 50 entries. If many networks exist on the internetwork, the router sends out multiple packets per update. For example, if a router has 120 entries in the routing table, it would send three routing update packets per update. The first routing update packet would include the first 50 entries, the second packet would include the next 50 entries, and the last routing update packet would include the last 20 entries.

### Sample Display

Figure 2-136 shows sample **debug ipx routing** output.

**Figure 2-136 Sample Debug IPX Routing Output**

```
Router# debug ipx routing

IPXRIP: update from 9999.0260.8c6a.1733
        110801 in 1 hops, delay 2
IPXRIP: sending update to 12FF02:ffff.ffff.ffff via Ethernet 1
        network 555, metric 2, delay 3
        network 1234, metric 3, delay 4
```

Table 2-73 describes significant fields shown in Figure 2-136.

**Table 2-73 Debug IPX Routing Field Descriptions**

Field	Description
IPXRIP	This is an IPX RIP packet.
update from 9999.0260.8c6a.1733	This packet is a routing update from an IPX server at address 9999.0260.8c6a.1733.
110801 in 1 hops	Network 110801 is one hop away from the router at address 9999.0260.8c6a.1733.
delay 2	Delay is a time measurement (1/18th second) that the NetWare shell uses to estimate how long to wait for a response from a file server. Also known as ticks.
sending update to 12FF02:ffff.ffff.ffff via Ethernet 1	The router is sending this IPX routing update packet to address 12FF02:ffff.ffff.ffff through its Ethernet 1 interface.

**Table 2-73      Debug IPX Routing Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
network 555	The packet includes routing update information for network 555.
metric 2	Network 555 is two metrics (or hops) away from the router.
delay 3	Network 555 is a delay of 3 away from the router. Delay is a measurement that the NetWare shell uses to estimate how long to wait for a response from a file server. Also known as ticks.

**Related Command**

**debug ipx sap**

## debug ipx sap

Use the **debug ipx sap** EXEC command to display information about IPX Service Advertisement Protocol (SAP) packets. The **no** form of this command disables debugging output.

[no] **debug ipx sap** [activity | events]

### Syntax Description

<b>activity</b>	(Optional) Provides more detailed output of SAP packets, including displays of services in SAP packets.
<b>events</b>	(Optional) Limits amount of detailed output for SAP packets to those that contain interesting events.

### Usage Guidelines

Normally, a router or server sends out one SAP update per minute. Each SAP packet can include up to seven entries. If many servers are advertising on the network, the router sends out multiple packets per update. For example, if a router has 20 entries in the SAP table, it would send three SAP packets per update. The first SAP would include the first seven entries, the second SAP would include the next seven entries, and the last update would include the last six entries.

Obtain the most meaningful detail by using the **debug ipx sap activity** and the **debug ipx sap events** commands together.



**Caution** Because the **debug ipx sap** command can generate a lot of output, use it with caution on networks that have many interfaces and large service tables.

### Sample Display

Figure 2-137 shows sample **debug ipx sap** output.

**Figure 2-137 Sample Debug IPX SAP Output**

```
Router# debug ipx sap

IPXSAP: at 0023F778:
I SAP Response type 0x2 len 160 src:160.0000.0c00.070d dest:160.ffff.ffff.ffff(452)
  type 0x4, "Hello2", 199.0002.0004.0006 (451), 2 hops
  type 0x4, "Hello1", 199.0002.0004.0008 (451), 2 hops
IPXSAP: sending update to 160
IPXSAP: at 00169080:
  O SAP Update type 0x2 len 96 ssoc:0x452 dest:160.ffff.ffff.ffff(452)
  IPX: type 0x4, "Magnolia", 42.0000.0000.0001 (451), 2hops
```

As Figure 2-137 shows, the **debug ipx sap** command generates multiple lines of output for each SAP packet—a packet summary message and a service detail message.

The first line displays the internal router memory address of the packet. The technical support staff may use this information in problem debugging.

```
IPXSAP: at 0023F778:
```

Table 2-74 describes the fields shown in the second line of output in Figure 2-137.

**Table 2-74      Debug IPX SAP Field Descriptions—Part 1**

Field	Description
I	Indication as to whether the router received the SAP packet as input (I) or is sending an update as output (O).
SAP Response type 0x2	Packet type. Format is 0xn; possible values for n include: <ul style="list-style-type: none"> <li>• 1—General query</li> <li>• 2—General response</li> <li>• 3—Get Nearest Server request</li> <li>• 4—Get Nearest Server response</li> </ul>
len 160	Length of this packet (in bytes).
src: 160.000.0c00.070d	Source address of the packet.
dest:160.ffff.ffff.ffff	The IPX network number and broadcast address of the destination IPX network for which the message is intended.
(452)	IPX socket number of the process sending the packet at the source address. This number is always 452, which is the socket number for the SAP process.

Table 2-75 describes the fields shown in the third and fourth lines of output in Figure 2-137.

**Table 2-75 Debug IPX SAP Field Descriptions—Part 2**

Field	Description
type 0x4	<p>Indicates the type of service the server sending the packet provides. Format is 0xn. Some of the values for n are proprietary to Novell. Those values for n that have been published include</p> <ul style="list-style-type: none"> <li>• 0—Unknown</li> <li>• 1—User</li> <li>• 2—User group</li> <li>• 3—Print queue</li> <li>• 4—File server</li> <li>• 5—Job server</li> <li>• 6—Gateway</li> <li>• 7—Print server</li> <li>• 8—Archive queue</li> <li>• 9—Archive server</li> <li>• A—Job queue</li> <li>• B—Administration</li> <li>• 21—NAS SNA gateway</li> <li>• 24—Remote bridge server</li> <li>• 2D—Time Synchronization VAP</li> <li>• 2E—Dynamic SAP</li> <li>• 47—Advertising print server</li> <li>• 4B—Btrieve VAP 5.0</li> <li>• 4C—SQL VAP</li> <li>• 7A—TES—NetWare for VMS</li> <li>• 98—NetWare access server</li> <li>• 9A—Named Pipes server</li> <li>• 9E—Portable NetWare—UNIX</li> <li>• 111—Test server</li> <li>• 166—NetWare management</li> <li>• 233—NetWare management agent</li> <li>• 237—NetExplorer NLM</li> <li>• 239—HMI hub</li> <li>• 23A—NetWare LANalyzer agent</li> <li>• 26A—NMS management</li> <li>• FFFF—Wildcard (any SAP service)</li> </ul> <p>Contact Novell for more information.</p>
“HELLO2”	Name of the server being advertised.
199.0002.0004.0006 (451)	Indicates the network number and address (and socket) of the server generating the SAP packet.
2 hops	Number of hops to the server from the router.

The fifth line of output indicates that the router sent a SAP update to network 160:

```
IPXSAP: sending update to 160
```

As Figure 2-137 shows, the format for **debug ipx sap** output describing a SAP update the router sends is similar to that describing a SAP update the router receives, except that the `ssoc:` field replaces the `src:` field, as the following line of output indicates:

```
O SAP Update type 0x2 len 96 ssoc:0x452 dest:160.ffff.ffff.ffff(452)
```

Table 2-76 describes possible values for the `ssoc:` field.

**Table 2-76 Debug IPX SAP Field Descriptions—Part 3**

Field	Description
ssoc:0x452	Indicates the IPX socket number of the process sending the packet at the source address. Possible values include <ul style="list-style-type: none"><li>• 451—Network Core Protocol</li><li>• 452—Service Advertising Protocol</li><li>• 453—Routing Information Protocol</li><li>• 455—NetBIOS</li><li>• 456—Diagnostics</li><li>• 4000 to 6000—Ephemeral sockets used for interaction with file servers and other network communications</li></ul>

Related Command  
**debug ipx routing**

## debug ipx spooof

Use the **debug ipx spooof** EXEC command to display information about SPX keepalive and IPX watchdog packets when **ipx watchdog** and **ipx spx-spooof** are configured on the router. The **no** form of this command disables debugging output.

**[no] debug ipx spooof**

### Usage Guidelines

Use this command to troubleshoot connections that use sequential packet exchange (SPX) spoofing when SPX keepalive spoofing is enabled.

### Sample Display

Figure 2-138 shows sample **debug ipx spooof** output.

**Figure 2-138 Sample Debug IPX Spooof Output**

```
Router# debug ipx spooof

IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (new) (changed:yes) Last Changed 0
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (new) (changed:yes) Last Changed 0

IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: 80 0 2B8 7104 29 7
7 (early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 42 tc=02, SPX: 80 0 4B8 7004 1D 8
8 (early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 32 tc=02, watchdog
IPX: local:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 32 tc=00, watchdog snet
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7
7 (early)
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7
7 (Last Changed 272 sec)
IPX: local:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, spx keepalive sent 80
0 7104 2B8 7 29 2E
```

Explanations for lines of output shown in Figure 2-138 follow.

The following lines show that SPX packets were seen, but they are not seen for a connection that exists in the SPX table:

```
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (new) (changed:yes) Last Changed 0
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (new) (changed:yes) Last Changed 0
```

The following lines show SPX packets are for connections that exist in the SPX table but SPX idle time has not yet elapsed and spoofing has not started:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: 80 0 2B8 7104 29 7
7 (early)
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 42 tc=02, SPX: 80 0 4B8 7004 1D 8
8 (early)
```

The following lines show an IPX watchdog packet and the spoofed reply:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.da75 ln= 32 tc=02, watchdog
IPX: local:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 32 tc=00, watchdog sent
```

The following lines show SPX packets that arrived more than two minutes after spoofing started. This situation occurs when the other sides of the SPX table are cleared. When the table is cleared, the routing processes stop spoofing the connection, which allows SPX keepalives from the local side to travel to the remote side and repopulate the SPX table.

```
IPX: Tu1:200.0260.8c8d.da75->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7004 4B8 8 1D
23 (changed:clear) Last Changed 0
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7
7 (early)
IPX: Tu1:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, SPX: 80 0 7104 2B8 7 29
2E (changed:clear) Last Changed 0
```

The following lines show that an SPX keepalive packet came in and was spoofed:

```
IPX: Et1:CC0001.0000.0000.0001->200.0260.8c8d.c558 ln= 42 tc=02, SPX: C0 0 2B8 7104 29 7
7 (Last Changed 272 sec)
IPX: local:200.0260.8c8d.c558->CC0001.0000.0000.0001 ln= 42 tc=02, spx keepalive sent 80
0 7104 2B8 7 29 2E
```

## debug ipx spx

Use the **debug ipx spx** EXEC command to display debugging messages related to the Sequenced Packet Exchange (SPX) protocol. The **no** form of this command disables debugging output.

**[no] debug ipx spx**

### Usage Guidelines

Use the **debug ipx spx** command to display handshaking or negotiating details between the SPX protocol and the other protocols or applications. SPX debugging messages indicate various states of SPX connections such as incoming and outgoing traffic information, timer events, and related processing of SPX connections.

### Sample Display

Figure 2-139 shows sample **debug ipx spx** output.

**Figure 2-139 Sample Debug IPX SPX Output**

```
Router# debug ipx spx

SPX: Sent an SPX packet
SPX: I Con Src/Dst 776E/20A0 d-strm 0 con-ctl 80
SPX: I Con Src/Dst 776E/20A0 d-strm FE con-ctl 40
SPX: C847C Connection close requested by peer
SPX: Sent an SPX packet
SPX: purge timer fired. Cleaning up C847C
SPX: purging spxcon C847C from conQ
SPX: returning inQ buffers
SPX: returning outQ buffers
SPX: returning unackedQ buffers
SPX: returning spxcon
SPX: I Con Src/Dst 786E/FFFF d-strm 0 con-ctl C0
SPX: new connection request for listening socket
SPX: Sent an SPX packet
SPX: I Con Src/Dst 786E/20B0 d-strm 0 con-ctl 40
SPX: 300 bytes data recvd
SPX: Sent an SPX packet
```

Explanations for representative lines of output in Figure 2-139 follow.

The following line indicates an incoming SPX packet that has a source connection ID of 776E and a destination connection ID of 20A0 (both in hexadecimal). The data stream value in the SPX packet is indicated by *d-strm*, and the connection control value in the SPX packet is indicated by *con-ctl* (both in hexadecimal). All data packets received are followed by an SPX debug message indicating the size of the packet. All control packets received are consumed internally.

```
SPX: I Con Src/Dst 776E/20A0 d-strm 0 con-ctl 80
```

The following lines indicate that SPX is attempting to remove an SPX connection that has the address C8476 from its list of connections:

```
SPX: purge timer fired. Cleaning up C847C
SPX: purging spxcon C847C from conQ
```

### Related Command

**debug ipx nasi**

## debug isdn event

Use the **debug isdn event** EXEC command to display Integrated Services Digital Network (ISDN) events occurring on the user side (on the router) of the ISDN interface. The ISDN events that can be displayed are Q.931 events (call setup and teardown of ISDN network connections). The **no** form of this command disables debugging output.

**[no] debug isdn event**

### Usage Guidelines

Although the **debug isdn event** and the **debug isdn q931** commands provide similar debug information, the information is displayed in a different format. If you want to see the information in both formats, enable both commands at the same time. The displays will be intermingled.

Use the **show dialer** command to retrieve information about the status and configuration of the ISDN interface on the router.

Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, refer to Appendix B, “ISDN Switch Types, Codes, and Values.”

### Sample Displays

Figure 2-140 shows sample **debug isdn event** output of call setup events for an outgoing call.

**Figure 2-140 Sample Debug ISDN Event Output—Call Setup Outgoing Call**

```
Router# debug isdn event

ISDN Event: Call to 415555121202
received HOST_PROCEEDING
Channel ID i = 0x0101
-----
Channel ID i = 0x89
received HOST_CONNECT
Channel ID i = 0x0101
ISDN Event: Connected to 415555121202 on B1 at 64 Kb/s
```

Figure 2-141 shows sample **debug isdn event** output of call setup events for an incoming call. The values used for internal purposes are unpacked information elements. The values that follow the ISDN specification are an interpretation of the unpacked information elements. Refer to Appendix B, “ISDN Switch Types, Codes, and Values,” for information about these values.

**Figure 2-141 Sample Debug ISDN Event Output—Call Setup Incoming Call**

```
Router# debug isdn event

received HOST_INCOMING_CALL
Bearer Capability i = 0x080010
-----
Channel ID i = 0x0101
Calling Party Number i = 0x0000, '415555121202'
IE out of order or end of 'private' IEs --
Bearer Capability i = 0x8890
Channel ID i = 0x89
```

```

Calling Party Number i = 0x0083, '415555121202'
ISDN Event: Received a call from 415555121202 on B1 at 64 Kb/s
ISDN Event: Accepting the call
received HOST_CONNECT
Channel ID i = 0x0101
ISDN Event: Connected to 415555121202 on B1 at 64 Kb/s

```

Figure 2-142 shows sample **debug isdn event** output of call teardown events for a call that has been disconnected by the host side of the connection.

#### Figure 2-142 Sample Debug ISDN Event Output—Call Teardown by Far End

```

Router# debug isdn event

received HOST_DISCONNECT
ISDN Event: Call to 415555121202 was hung up

```

Figure 2-143 shows sample **debug isdn event** output of a call teardown event for an outgoing or incoming call that has been disconnected by the ISDN interface on the router side.

#### Figure 2-143 Sample Debug ISDN Event Output—Call Teardown Local Side

```

Router# debug isdn event

ISDN Event: Hangup call to call id 0x8008

```

Table 2-77 describes significant fields shown in Figure 2-140 through Figure 2-143.

**Table 2-77 Debug ISDN Event Field Descriptions**

Field	Description
Bearer Capability	Indicates the requested bearer service to be provided by the network. Refer to Table B-4 in the “ISDN Switch Types, Codes, and Values” appendix for detailed information about bearer capability values.
i=	Indicates the Information Element Identifier. The value depends on the field it is associated with. Refer to the ITU-T <sup>1</sup> Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.
Channel ID	Channel Identifier. The values and corresponding channels might be identified in several ways: Channel ID i = 0x0101—Channel B1 Channel ID i = 0x0102—Channel B2  ITU-T Q.931 defines the values and channels as exclusive or preferred: Channel ID i = 0x83—Any B-channel Channel ID i = 0x89—Channel B1 (exclusive) Channel ID i = 0x8A—Channel B2 (exclusive)  Channel ID i = 0x81—B1 (preferred) Channel ID i = 0x82—B2 (preferred)
Calling Party Number	Identifies the called party. This field is only present in outgoing calls. The Calling Party Number field uses the IA5 character set. Note that it may be replaced by the Keypad facility field.

**Table 2-77 Debug ISDN Event Field Descriptions (Continued)**

Field	Description
IE out of order or end of 'private' IEs	Indicates that an information element identifier is out of order or there are no more private network information element identifiers to interpret.
Received a call from 415555121202 on B1 at 64Kb/s	Identifies the origin of the call. This field is present only in incoming calls. Note that the information about the incoming call includes the channel and speed. Whether the channel and speed are displayed depends on the network delivering the calling party number.

1. The ITU-T carries out the functions of the former Consultative Committee for International Telegraph and Telephone.

Figure 2-144 shows sample **debug isdn event** output of a call teardown event for a call that has passed call screening then has been hung up by the ISDN interface on the far end side.

**Figure 2-144 Sample Debug ISDN Event Output—Call Screening Normal Disconnect**

```
Router# debug isdn event

Jan  3 11:29:52.559: ISDN BR0: RX <- DISCONNECT pd = 8  callref = 0x81
Jan  3 11:29:52.563:          Cause i = 0x8090 - Normal call clearing
```

Figure 2-145 shows sample **debug isdn event** output of a call teardown event for a call that has not passed call screening and has been rejected by the ISDN interface on the router side.

**Figure 2-145 Sample Debug ISDN Event Output—Call Screening Call Rejection**

```
Router# debug isdn event

Jan  3 11:32:03.263: ISDN BR0: RX <- DISCONNECT pd = 8  callref = 0x85
Jan  3 11:32:03.267:          Cause i = 0x8095 - Call rejected
```

Figure 2-146 shows sample **debug isdn event** output of a call teardown event for an outgoing call that uses a dialer subaddress.

**Figure 2-146 Sample Debug ISDN Event Output—Called Party Subaddress**

```
Router# debug isdn event

Jan  3 11:41:48.483: ISDN BR0: Event: Call to 61885:1212 at 64 Kb/s
Jan  3 11:41:48.495: ISDN BR0: TX -> SETUP pd = 8  callref = 0x04
Jan  3 11:41:48.495:          Bearer Capability i = 0x8890
Jan  3 11:41:48.499:          Channel ID i = 0x83
Jan  3 11:41:48.503:          Called Party Number i = 0x80, '61885'
Jan  3 11:41:48.507:          Called Party SubAddr i = 0x80, 'P1212'
Jan  3 11:41:48.571: ISDN BR0: RX <- CALL_PROC pd = 8  callref = 0x84
Jan  3 11:41:48.575:          Channel ID i = 0x89
Jan  3 11:41:48.587: ISDN BR0: Event: incoming ces value = 1
Jan  3 11:41:48.587: ISDN BR0: received HOST_PROCEEDING
Jan  3 11:41:48.591:          Channel ID i = 0x0101
Jan  3 11:41:48.591:          -----
Jan  3 11:41:48.591:          Channel ID i = 0x89
Jan  3 11:41:48.731: ISDN BR0: RX <- CONNECT pd = 8  callref = 0x84
Jan  3 11:41:48.743: ISDN BR0: Event: incoming ces value = 1
Jan  3 11:41:48.743: ISDN BR0: received HOST_CONNECT
```

```

Channel ID i = 0x0101
Jan  3 11:41:48.747: -----
%LINK-3-UPDOWN: Interface BRI0:1 changed state to up
Jan  3 11:41:48.771: ISDN BR0: Event: Connected to 61885:1212 on B1 at 64 Kb/s
Jan  3 11:41:48.775: ISDN BR0: TX -> CONNECT_ACK pd = 8  callref = 0x04
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI0:1 is now connected to 61885:1212 goodie

```

The output shown in Figure 2-144 through Figure 2-146 is similar to the output of **debug isdn q931**. Refer to the **debug isdn q931** command for detailed field descriptions.

Figure 2-147 shows sample **debug isdn event** command output of call setup events for a successful callback for legacy DDR.

**Figure 2-147 Successful Call for Caller ID Callback and Legacy DDR**

```

Router# debug isdn event

BRI0:Caller id Callback server starting to spanky 81012345678902
: Callback timer expired
BRI0:beginning callback to spanky 81012345678902
BRI0: Attempting to dial 81012345678902

```

Figure 2-148 shows sample **debug isdn event** command output for a callback that was unsuccessful because the router had no dialer map for the calling number.

**Figure 2-148 Unsuccessful Call for Caller ID Callback and Legacy DDR**

```

Router# debug isdn event

BRI0:Caller id 81012345678902 callback - no matching map

```

Table 2-78 describes significant fields shown in Figure 2-147 and Figure 2-148.

**Table 2-78 Debug ISDN Event Field Descriptions for Caller ID Callback and Legacy DDR**

Field	Description
BRI0:Caller id Callback server starting to ...	Caller ID callback has started, plus host name and number called. The callback enable timer starts now.
: Callback timer expired	Callback timer has expired; callback can proceed.
BRI0:beginning callback to ... BRI0: Attempting to dial ...	Actions proceeding after the callback timer expired, plus host name and number called.

Figure 2-149 shows sample **debug isdn event** command output for a callback that was successful when the dialer profiles DDR feature is configured.

**Figure 2-149 Successful Call for Caller ID Callback and Dialer Profiles**

```

*Mar  1 00:46:51.827: BR0:1:Caller id 81012345678901 matched to profile delorean
*Mar  1 00:46:51.827: Dialer1:Caller id Callback server starting to delorean 81012345678901
*Mar  1 00:46:54.151: : Callback timer expired
*Mar  1 00:46:54.151: Dialer1:beginning callback to delorean 81012345678901

```

## debug isdn event

---

```
*Mar 1 00:46:54.155: Freeing callback to delorean 81012345678901
*Mar 1 00:46:54.155: BRI0: Dialing cause Callback return call
*Mar 1 00:46:54.155: BRI0: Attempting to dial 81012345678901
*Mar 1 00:46:54.503: %LINK-3-UPDOWN: Interface BRI0:2, changed state to up
*Mar 1 00:46:54.523: %DIALER-6-BIND: Interface BRI0:2 bound to profile Dialer1
*Mar 1 00:46:55.139: %LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:2, changed state to up
*Mar 1 00:46:58.187: %ISDN-6-CONNECT: Interface BRI0:2 is now connected to 81012345678901 delorean
```

Table 2-79 describes significant fields shown in Figure 2-149 of call setup events for a successful call back for the sample output from the **debug isdn events** command when the dialer profiles DDR feature is configured.

**Table 2-79** Debug ISDN Event Field Descriptions for Caller ID Callback and Dialer Profiles

Field	Description
BR0:1:Caller id ... matched to profile ...	Interface, channel number, caller ID that is matched, and the profile to bind to the interface.
: Callback timer expired	Callback timer has expired; callback can proceed.
Dialer1:beginning callback to...	Callback process is beginning to the specified number.
Freeing callback to...	Callback has been started to the specified number, and the number has been removed from the callback list.
BRI0: Dialing cause Callback return call BRI0: Attempting to dial	The reason for the call and the number being dialed.
%LINK-3-UPDOWN: Interface BRI0:2, changed state to up	Interface status: up.
%DIALER-6-BIND: Interface BRI0:2 bound to profile Dialer1	Profile bound to the interface.
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:2, changed state to up	Line protocol status: up.
%ISDN-6-CONNECT: Interface BRI0:2 is now connected to ...	Interface is now connected to the specified host and number.

## debug isdn q921

Use the **debug isdn q921** EXEC command to display data link layer (Layer 2) access procedures that are taking place at the router on the D channel (LAPD) of its Integrated Services Digital Network (ISDN) interface. The **no** form of this command disables debugging output.

**[no] debug isdn q921**

### Usage Guidelines

The ISDN data link layer interface provided by the router conforms to the user interface specification defined by ITU-T recommendation Q.921. The **debug isdn q921** command output is limited to commands and responses exchanged during peer-to-peer communication carried over the D channel. This debug information does not include data transmitted over the B channels that are also part of the router's ISDN interface. The peers (data link layer entities and layer management entities on the routers) communicate with each other via an ISDN switch over the D channel.

---

**Note** The ISDN switch provides the network interface defined by Q.921. This debug command does not display data link layer access procedures taking place within the ISDN network (that is, procedures taking place on the network side of the ISDN connection). See Appendix B, "ISDN Switch Types, Codes, and Values," for a list of the supported ISDN switch types.

---

A router can be the calling or called party of the ISDN Q.921 data link layer access procedures. If the router is the calling party, the command displays information about an outgoing call. If the router is the called party, the command displays information about an incoming call and the keepalives.

The **debug isdn q921** command can be used with the **debug isdn event** and the **debug isdn q931** commands at the same time. The displays will be intermingled.

Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, refer to Appendix B, "ISDN Switch Types, Codes, and Values."

### Sample Display

Figure 2-150 shows sample **debug isdn q921** output for an outgoing call.

**Figure 2-150 Sample Debug ISDN Q921 Output—Outgoing Call**

```
Router# debug isdn q921

Jan  3 14:52:24.475: ISDN BR0: TX -> INFOc sapi = 0  tei = 64  ns = 5  nr = 2
                        i = 0x08010705040288901801837006803631383835
Jan  3 14:52:24.503: ISDN BR0: RX <- RRr sapi = 0  tei = 64  nr = 6
Jan  3 14:52:24.527: ISDN BR0: RX <- INFOc sapi = 0  tei = 64  ns = 2  nr = 6
                        i = 0x08018702180189
Jan  3 14:52:24.535: ISDN BR0: TX -> RRr sapi = 0  tei = 64  nr = 3
Jan  3 14:52:24.643: ISDN BR0: RX <- INFOc sapi = 0  tei = 64  ns = 3  nr = 6
                        i = 0x08018707
Jan  3 14:52:24.655: ISDN BR0: TX -> RRr sapi = 0  tei = 64  nr = 4
%LINK-3-UPDOWN: Interface BRI0:1, changed state to up
Jan  3 14:52:24.683: ISDN BR0: TX -> INFOc sapi = 0  tei = 64  ns = 6  nr = 4
```

```

                                i = 0x0801070F
Jan  3 14:52:24.699: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 7
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
%ISDN-6-CONNECT: Interface BRI0:1 is now connected to 61885 goodie
Jan  3 14:52:34.415: ISDN BR0: RX <-  RRp sapi = 0  tei = 64  nr = 7
Jan  3 14:52:34.419: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 4

```

In the following lines from Figure 2-150, the seventh and eighth most significant hexadecimal numbers indicate the type of message. 0x05 indicates a Call Setup message, 0x02 indicates a Call Proceeding message, 0x07 indicates a Call Connect message, and 0x0F indicates a Connect Ack message.

```

Jan  3 14:52:24.475: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 5  nr = 2
                                i = 0x08010705040288901801837006803631383835
Jan  3 14:52:24.527: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 2  nr = 6
                                i = 0x08018702180189
Jan  3 14:52:24.643: ISDN BR0: RX <-  INFOc sapi = 0  tei = 64  ns = 3  nr = 6
                                i = 0x08018707
Jan  3 14:52:24.683: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 6  nr = 4
                                i = 0x0801070F

```

Figure 2-151 shows sample **debug isdn q921** output for a startup message on a DMS-100 switch.

#### Figure 2-151 Sample Debug ISDN Q921 Output—Startup Message on a DMS-100 Switch

```

Router# debug isdn q921

Jan  3 14:47:28.455: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127  0
Jan  3 14:47:30.171: ISDN BR0: TX ->  IDREQ  ri = 31815  ai = 127
Jan  3 14:47:30.219: ISDN BR0: RX <-  IDASSN  ri = 31815  ai = 64
Jan  3 14:47:30.223: ISDN BR0: TX ->  SABMEp  sapi = 0  tei = 64
Jan  3 14:47:30.227: ISDN BR0: RX <-  IDCKRQ  ri = 0  ai = 127
Jan  3 14:47:30.235: ISDN BR0: TX ->  IDCKRP  ri = 16568  ai = 64
Jan  3 14:47:30.239: ISDN BR0: RX <-  UAf sapi = 0  tei = 64
Jan  3 14:47:30.247: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 0  nr = 0
                                i = 0x08007B3A03313233
Jan  3 14:47:30.267: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 1
Jan  3 14:47:34.243: ISDN BR0: TX ->  INFOc sapi = 0  tei = 64  ns = 1  nr = 0
                                i = 0x08007B3A03313233
Jan  3 14:47:34.267: ISDN BR0: RX <-  RRr sapi = 0  tei = 64  nr = 2
Jan  3 14:47:43.815: ISDN BR0: RX <-  RRp sapi = 0  tei = 64  nr = 2
Jan  3 14:47:43.819: ISDN BR0: TX ->  RRf sapi = 0  tei = 64  nr = 0
Jan  3 14:47:53.819: ISDN BR0: TX ->  RRp sapi = 0  tei = 64  nr = 0

```

The first seven lines of this example indicate an L2 link establishment. Explanations for individual lines of output from Figure 2-151 follow.

The following lines indicate the message exchanges between the data link layer entity on the local router (user side) and the assignment source point (ASP) on the network side during the TEI assignment procedure. This assumes that the link is down and no TEI currently exists.

```

Jan  3 14:47:30.171: ISDN BR0: TX ->  IDREQ  ri = 31815  ai = 127
Jan  3 14:47:30.219: ISDN BR0: RX <-  IDASSN  ri = 31815  ai = 64

```

At 14:47:30.171, the local router data link layer entity sent an Identity Request message to the network data link layer entity to request a TEI value that can be used in subsequent communication between the peer data link layer entities. The request includes a randomly generated reference number (31815) to differentiate among user devices that request automatic TEI assignment and an action indicator of 127 to indicate that the ASP can assign any TEI value available. The ISDN user interface on the router uses automatic TEI assignment.

At 14:47:30.219, the network data link entity responds to the Identity Request message with an Identity Assigned message. The response includes the reference number (31815) previously sent in the request and TEI value (64) assigned by the ASP.

The following lines indicate the message exchanges between the layer management entity on the network and the layer management entity on the local router (user side) during the TEI check procedure:

```
Jan 3 14:47:30.227: ISDN BR0: RX <- IDCKRQ ri = 0 ai = 127
Jan 3 14:47:30.235: ISDN BR0: TX -> IDCKRP ri = 16568 ai = 64
```

At 14:47:30.227, the layer management entity on the network sends the Identity Check Request message to the layer management entity on the local router to check whether a TEI is in use. The message includes a reference number that is always 0 and the TEI value to check. In this case, an ai value of 127 indicates that all TEI values should be checked. At 14:47:30.227, the layer management entity on the local router responds with an Identity Check Response message indicating that TEI value 64 is currently in use.

The following lines indicate the messages exchanged between the data link layer entity on the local router (user side) and the data link layer on the network side to place the network side into modulo 128 multiple frame acknowledged operation. Note that the data link layer entity on the network side also can initiate the exchange.

```
Jan 3 14:47:30.223: ISDN BR0: TX -> SABMEp sapi = 0 tei = 64
Jan 3 14:47:30.239: ISDN BR0: RX <- UAf sapi = 0 tei = 64
```

At 14:47:30.223, the data link layer entity on the local router sends the SABME command with a SAPI of 0 (call control procedure) for TEI 64. At 14:47:30.239, the first opportunity, the data link layer entity on the network responds with a UA response. This response indicates acceptance of the command. The data link layer entity sending the SABME command may have to send it more than once before receiving a UA response.

The following lines indicate the status of the data link layer entities. Both are ready to receive I frames.

```
Jan 3 14:47:43.815: ISDN BR0: RX <- RRp sapi = 0 tei = 64 nr = 2
Jan 3 14:47:43.819: ISDN BR0: TX -> RRf sapi = 0 tei = 64 nr = 0
```

These I frames are typically exchanged every 10 seconds (T203 timer).

Figure 2-152 shows sample **debug isdn q921** output for an incoming call. It is an incoming SETUP message that assumes the L2 link is already established to the other side.

**Figure 2-152 Sample Debug ISDN Q921 Output—Incoming Call**

```
Router# debug isdn q921

Jan 3 14:49:22.507: ISDN BR0: TX -> RRp sapi = 0 tei = 64 nr = 0
Jan 3 14:49:22.523: ISDN BR0: RX <- RRf sapi = 0 tei = 64 nr = 2
Jan 3 14:49:32.527: ISDN BR0: TX -> RRp sapi = 0 tei = 64 nr = 0
Jan 3 14:49:32.543: ISDN BR0: RX <- RRf sapi = 0 tei = 64 nr = 2
Jan 3 14:49:42.067: ISDN BR0: RX <- RRp sapi = 0 tei = 64 nr = 2
Jan 3 14:49:42.071: ISDN BR0: TX -> RRf sapi = 0 tei = 64 nr = 0
Jan 3 14:49:47.307: ISDN BR0: RX <- UI sapi = 0 tei = 127
                        i = 0x08011F05040288901801897006C13631383836
%LINK-3-UPDOWN: Interface BRI0:1, changed state to up
Jan 3 14:49:47.347: ISDN BR0: TX -> INFOc sapi = 0 tei = 64 ns = 2 nr = 0
                        i = 0x08019F07180189
Jan 3 14:49:47.367: ISDN BR0: RX <- RRr sapi = 0 tei = 64 nr = 3
Jan 3 14:49:47.383: ISDN BR0: RX <- INFOc sapi = 0 tei = 64 ns = 0 nr = 3
                        i = 0x08011F0F180189
```

```
Jan 3 14:49:47.391: ISDN BR0: TX -> RRr sapi = 0 tei = 64 nr = 1
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0:1, changed state to up
```

Table 2-80 describes significant fields in Figure 2-150, Figure 2-151, and Figure 2-152.

**Table 2-80 Debug ISDN Q921 Field Descriptions**

Field	Description
Jan 3 14:49:47.391	Indicates the date and time at which the frame was transmitted from or received by the data link layer entity on the router. The time is maintained by an internal clock.
TX	Indicates that this frame is being transmitted from the ISDN interface on the local router (user side).
RX	Indicates that this frame is being received by the ISDN interface on the local router from the peer (network side).
IDREQ	Indicates the Identity Request message type sent from the local router to the network (assignment source point [ASP]) during the automatic terminal endpoint identifier (TEI) assignment procedure. This message is sent in a UI command frame. The service access point identifier (SAPI) value for this message type is always 63 (indicating that it is a Layer 2 management procedure) but it is not displayed. The TEI value for this message type is 127 (indicating that it is a broadcast operation).
ri = 31815	Indicates the Reference number used to differentiate between user devices requesting TEI assignment. This value is a randomly generated number between 0 and 65535. The same ri value sent in the IDREQ message should be returned in the corresponding IDASSN message. Note that a Reference number of 0 indicates that the message is sent from the network side management layer entity and a reference number has not been generated.
ai = 127	Indicates the Action indicator used to request that the ASP assign any TEI value. It is always 127 for the broadcast TEI. Note that in some message types, such as IDREM, a specific TEI value is indicated.
IDREM	Indicates the Identity Remove message type sent from the ASP to the user side layer management entity during the TEI removal procedure. This message is sent in a UI command frame. The message includes a reference number that is always 0, because it is not responding to a request from the local router. The ASP sends the Identity Remove message twice to avoid message loss.
IDASSN	Indicates the Identity Assigned message type sent from the ISDN service provider on the network to the local router during the automatic TEI assignment procedure. This message is sent in a UI command frame. The SAPI value for this message type is always 63 (indicating that it is a Layer 2 management procedure). The TEI value for this message type is 127 (indicating it is a broadcast operation).
ai = 64	Indicates the TEI value automatically assigned by the ASP. This TEI value is used by data link layer entities on the local router in subsequent communication with the network. The valid values are in the range 64 to 126.

Table 2-80 Debug ISDN Q921 Field Descriptions (Continued)

Field	Description
SABME	Indicates the set asynchronous balanced mode extended command. This command places the recipient into modulo 128 multiple frame acknowledged operation. This command also indicates that all exception conditions have been cleared. The SABME command is sent once a second for N200 times (typically three times) until its acceptance is confirmed with a UA response. For a list and brief description of other commands and responses that can be exchanged between the data link layer entities on the local router and the network, see ITU-T Recommendation Q.921.
sapi = 0	Identifies the service access point at which the data link layer entity provides services to Layer 3 or to the management layer. A SAPI with the value 0 indicates it is a call control procedure. Note that the Layer 2 management procedures such as TEI assignment, TEI removal, and TEI checking, which are tracked with the <b>debug isdn q921</b> command, do not display the corresponding SAPI value; it is implicit. If the SAPI value were displayed it would be 63.
tei = 64	Indicates the TEI value automatically assigned by the ASP. This TEI value will be used by data link layer entities on the local router in subsequent communication with the network. The valid values are in the range 64 to 126.
IDCKRQ	Indicates the Identity Check Request message type sent from the ISDN service provider on the network to the local router during the TEI check procedure. This message is sent in a UI command frame. The ri field is always 0. The ai field for this message contains either a specific TEI value for the local router to check or 127, which indicates that the local router should check all TEI values. For a list and brief description of other message types that can be exchanged between the local router and the ISDN service provider on the network, see Appendix B, "ISDN Switch Types, Codes, and Values."
IDCKRP	Indicates the Identity Check Response message type sent from the local router to the ISDN service provider on the network during the TEI check procedure. This message is sent in a UI command frame in response to the IDCKRQ message. The ri field is a randomly generated number between 0 and 65535. The ai field for this message contains the specific TEI value that has been checked.
Uaf	Confirms that the network side has accepted the SABME command previously sent by the local router. The final bit is set to 1.
INFOc	Indicates that this is an Information command. It is used to transfer sequentially numbered frames containing information fields that are provided by Layer 3. The information is transferred across a datalink connection.
INFORMATION pd = 8 callref = (null)	Indicates the information fields provided by Layer 3. The information is sent one frame at a time. If multiple frames need to be sent, several Information commands are sent. The pd value is the protocol discriminator. The value 8 indicates it is call control information. The call reference number is always null for SPID information.

**Table 2-80      Debug ISDN Q921 Field Descriptions (Continued)**

Field	Description
SPID information i = 0x3431353930333833363031	Indicates the service profile identifier (SPID). The local router sends this information to the ISDN switch to indicate the services to which it subscribes. SPIDs are assigned by the service provider and are usually 10-digit telephone numbers followed by optional numbers. Currently, only the DMS-100 switch supports SPIDs, one for each B channel. If SPID information is sent to a switch type other than DMS-100, an error may be displayed in the debug information.
ns = 0	Indicates the send sequence number of transmitted I frames.
nr = 0	Indicates the expected send sequence number of the next received I frame. At time of transmission, this value should be equal to the value of ns. The value of nr is used to determine whether frames need to be retransmitted for recovery.
RRr	Indicates the Receive Ready response for unacknowledged information transfer. The RRr is a response to an INFOc.
RRp	Indicates the Receive Ready command with the poll bit set. The data link layer entity on the user side uses the poll bit in the frame to solicit a response from the peer on the network side.
RRf	Indicates the Receive Ready response with the final bit set. The data link layer entity on the network side uses the final bit in the frame to indicate a response to the poll.
sapi	Indicates the service access point identifier. The SAPI is the point at which data link services are provided to a network layer or management entity. Currently, this field can have the value 0 (for call control procedure) or 63 (for Layer 2 management procedures).
tei	Indicates the terminal endpoint identifier (TEI) that has been assigned automatically by the assignment source point (ASP) (also called the layer management entity on the network side). The valid range is 64 to 126. The value 127 indicates a broadcast.

## debug isdn q931

Use the **debug isdn q931** EXEC command to display information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network. The **no** form of this command disables debugging output.

**[no] debug isdn q931**

### Usage Guidelines

The ISDN network layer interface provided by the router conforms to the user interface specification defined by ITU-T recommendation Q.931, supplemented by other specifications such as for switch type VN4. The router tracks only activities that occur on the user side, not the network side, of the network connection. The display information **debug isdn q931** command output is limited to commands and responses exchanged during peer-to-peer communication carried over the D channel. This debug information does not include data transmitted over the B channels, which are also part of the router's ISDN interface. The peers (network layers) communicate with each other via an ISDN switch over the D channel.

A router can be the calling or called party of the ISDN Q.931 network connection call setup and tear-down procedures. If the router is the calling party, the command displays information about an outgoing call. If the router is the called party, the command displays information about an incoming call.

You can use the **debug isdn q931** command with the **debug isdn event** and the **debug isdn q921** commands at the same time. The displays will be intermingled. Use the **service timestamps debug datetime msec** global configuration command to include the time with each message.

For more information on ISDN switch types, codes, and values, refer to Appendix B, "ISDN Switch Types, Codes, and Values."

### Sample Displays

Figure 2-153 shows sample **debug isdn q931** output of a call setup procedure for an outgoing call.

**Figure 2-153 Sample Debug ISDN Q931 Output—Call Setup Procedure for an Outgoing Call**

```
Router# debug isdn q931

TX -> SETUP pd = 8 callref = 0x04
  Bearer Capability i = 0x8890
  Channel ID i = 0x83
  Called Party Number i = 0x80, '415555121202'
RX <- CALL_PROC pd = 8 callref = 0x84
  Channel ID i = 0x89
RX <- CONNECT pd = 8 callref = 0x84
TX -> CONNECT_ACK pd = 8 callref = 0x04....
Success rate is 0 percent (0/5)
```

Figure 2-154 shows sample **debug isdn q931** output of a call setup procedure for an incoming call.

**Figure 2-154 Sample Debug ISDN Q931 Output—Call Setup Procedure for an Incoming Call**

```
Router# debug isdn q931

RX <- SETUP pd = 8 callref = 0x06
  Bearer Capability i = 0x8890
```

```

Channel ID i = 0x89
Calling Party Number i = 0x0083, '81012345678902'
TX -> CONNECT pd = 8 callref = 0x86
RX <- CONNECT_ACK pd = 8 callref = 0x06
    
```

Figure 2-155 shows sample **debug isdn q931** output of a call teardown procedure from the network.

**Figure 2-155 Sample Debug ISDN Q931 Output—Call Teardown Procedure from the Network**

```

Router# debug isdn q931

RX <- DISCONNECT pd = 8 callref = 0x84
Cause i = 0x8790
Looking Shift to Codeset 6
Codeset 6 IE 0x1 1 0x82 '10'
TX -> RELEASE pd = 8 callref = 0x04
Cause i = 0x8090
RX <- RELEASE_COMP pd = 8 callref = 0x84
    
```

Figure 2-156 shows sample **debug isdn q931** output of a call teardown procedure from the router.

**Figure 2-156 Sample Debug ISDN Q931 Output—Call Teardown Procedure from the Router**

```

Router# debug isdn q931

TX -> DISCONNECT pd = 8 callref = 0x05
Cause i = 0x879081
RX <- RELEASE pd = 8 callref = 0x85
Looking Shift to Codeset 6
Codeset 6 IE 0x1 1 0x82 '10'
TX <- RELEASE_COMP pd = 8 callref = 0x05
    
```

Table 2-81 describes significant fields in Figure 2-153 through Figure 2-156.

**Table 2-81 Debug ISDN Q931 Call Setup Procedure Field Descriptions**

Field	Description
TX ->	Indicates that this message is being transmitted from the local router (user side) to the network side of the ISDN interface.
RX <-	Indicates that this message is being received by the user side of the ISDN interface from the network side.
SETUP	Indicates that the SETUP message type has been sent to initiate call establishment between peer network layers. This message can be sent from either the local router or the network.
pd	Indicates the protocol discriminator. The protocol discriminator distinguishes messages for call control over the user-network ISDN interface from other ITU-T-defined messages, including other Q.931 messages. The protocol discriminator is 8 for call control messages such as SETUP. For basic-1tr6, the protocol discriminator is 65.

**Table 2-81 Debug ISDN Q931 Call Setup Procedure Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
callref	Indicates the call reference number in hexadecimal. The value of this field indicates the number of calls made from either the router (outgoing calls) or the network (incoming calls). Note that the originator of the SETUP message sets the high-order bit of the call reference number to 0. The destination of the connection sets the high-order bit to 1 in subsequent call control messages, such as the CONNECT message. For example, callref = 0x04 in the request becomes callref = 0x84 in the response.
Bearer Capability	Indicates the requested bearer service to be provided by the network. Refer to Table B-4 in Appendix B, "ISDN Switch Types, Codes, and Values," for detailed information about bearer capability values.
i =	Indicates the Information Element Identifier. The value depends on the field it is associated with. Refer to the ITU-T Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.
Channel ID	Indicates the Channel Identifier. The value 83 indicates any channel, 89 indicates the B1 channel, and 8A indicates the B2 channel. For more information about the Channel Identifier, refer to ITU-T Recommendation Q.931.
Called Party Number	Identifies the called party. This field is only present in outgoing SETUP messages. Note that it can be replaced by the Keypad facility field. This field uses the IA5 character set.
Calling Party Number	Identifies the origin of the call. This field is present only in incoming SETUP messages. This field uses the IA5 character set.
CALL_PROC	Indicates the CALL PROCEEDING message; the requested call setup has begun and no more call setup information will be accepted.
CONNECT	Indicates that the called user has accepted the call.
CONNECT_ACK	Indicates that the calling user acknowledges the called user's acceptance of the call.
DISCONNECT	Indicates either that the user side has requested the network to clear an end-to-end connection or that the network has cleared the end-to-end connection.
Cause	Indicates the cause of the disconnect. Refer to Table B-2 and Table B-3 in Appendix B, "ISDN Switch Types, Codes, and Values," for detailed information about DISCONNECT cause codes and RELEASE cause codes.
Looking Shift to Codeset 6	Indicates that the next information elements will be interpreted according to information element identifiers assigned in codeset 6. Codeset 6 means that the information elements are specific to the local network.
Codeset 6 IE 0x1 i = 0x82, '10'	Indicates charging information. This information is specific to the NTT switch type and may not be sent by other switch types.
RELEASE	Indicates that the sending equipment will release the channel and call reference. The recipient of this message should prepare to release the call reference and channel.
RELEASE_COMP	Indicates that the sending equipment has received a RELEASE message and has now released the call reference and channel.

## debug isis adj packets

Use the **debug isis adj packets** EXEC command to display information on all adjacency-related activity such as hello packets sent and received and IS-IS adjacencies going up and down. The **no** form of this command disables debugging output.

**[no] debug isis adj packets**

### Sample Display

Figure 2-157 shows sample **debug isis adj packets** output.

**Figure 2-157 Sample Debug ISIS Adj Packets Output**

```
Router# debug isis adj packets

ISIS-Adj: Rec L1 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
ISIS-Adj: Rec L2 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
ISIS-Adj: Rec L1 IIH from 0000.0c00.0c36 (Ethernet1), cir type 3, cir id CCCC.CCCC.CCCC.03
ISIS-Adj: Area mismatch, level 1 IIH on Ethernet1
ISIS-Adj: Sending L1 IIH on Ethernet1
ISIS-Adj: Sending L2 IIH on Ethernet1
ISIS-Adj: Rec L2 IIH from 0000.0c00.0c36 (Ethernet1), cir type 3, cir id BBBB.BBBB.BBBB.03
```

Explanations for individual lines of output from Figure 2-157 follow.

The following line indicates that the router received an IS-IS hello packet (IIH) on Ethernet0 from the Level 1 router (L1) at MAC address 0000.0c00.40af. The circuit type is the interface type: 1—Level 1 only; 2—Level 2 only; 3—Level 1/2.

The circuit ID is what the neighbor interprets as the designated router for the interface.

```
ISIS-Adj: Rec L1 IIH from 0000.0c00.40af (Ethernet0), cir type 3, cir id BBBB.BBBB.BBBB.01
```

The following line indicates that the router (configured as a Level 1 router) received on Ethernet1 an IS-IS hello packet from a Level 1 router in another area, thereby declaring an area mismatch:

```
ISIS-Adj: Area mismatch, level 1 IIH on Ethernet1
```

The following lines indicates that the router (configured as a Level 1/Level 2 router) sent on Ethernet1 a Level 1 IS-IS hello packet, and then a Level 2 IS-IS packet:

```
ISIS-Adj: Sending L1 IIH on Ethernet1
ISIS-Adj: Sending L2 IIH on Ethernet1
```

## debug isis spf statistics

Use the **debug isis spf statistics** EXEC command to display statistical information about building routes between intermediate systems (ISs). The **no** form of this command disables debugging output.

**[no] debug isis spf statistics**

### Usage Guidelines

The Intermediate System-to-Intermediate System (IS-IS) Intra-Domain Routing Exchange Protocol (IDRP) provides routing between ISs by flooding the network with link-state information. IS-IS provides routing at two levels, intra-area (Level 1) and intra-domain (Level 2). Level 1 routing allows Level 1 ISs to communicate with other Level 1 ISs in the same area. Level 2 routing allows Level 2 ISs to build an interdomain backbone between Level 1 areas by traversing only Level 2 ISs. Level 1 ISs only need to know the path to the nearest Level 2 IS in order to take advantage of the interdomain backbone created by the Level 2 ISs.

The IS-IS protocol uses the Shortest Path First (SPF) routing algorithm to build Level 1 and Level 2 routes. The **debug isis spf statistics** command provides information for determining how long it takes to place a Level 1 IS or Level 2 IS on the shortest path tree (SPT) using the IS-IS protocol.

---

**Note** The SPF algorithm is also called the Dijkstra algorithm, after the creator of the algorithm.

---

### Sample Display

Figure 2-158 shows sample **debug isis spf statistics** output.

**Figure 2-158 Sample Debug ISIS SPF Statistics Output**

```
Router# debug isis spf statistics

ISIS-Stats: Compute L1 SPT, Timestamp 2780.328 seconds
ISIS-Stats: Complete L1 SPT, Compute time 0.004, 1 nodes on SPT
ISIS-Stats: Compute L2 SPT, Timestamp 2780.3336 seconds
ISIS-Stats: Complete L2 SPT, Compute time 0.056, 12 nodes on SPT
```

Table 2-82 describes significant fields shown in Figure 2-158.

**Table 2-82 Debug ISIS SPF Statistics Field Descriptions**

Field	Description
Compute L1 SPT	Indicates that Level 1 ISs are to be added to a Level 1 area.
Timestamp	Indicates the time at which the SPF algorithm was applied. The time is expressed as the number of seconds elapsed since the system was up and configured.
Complete L1 SPT	Indicates that the algorithm has completed for Level 1 routing.
Compute time	Indicates the time it took to place the ISs on the shortest path tree (SPT).
nodes on SPT	Indicates the number of ISs that have been added.
Compute L2 SPT	Indicates that Level 2 ISs are to be added to domain.
Complete L2 SPT	Indicates that the algorithm has completed for Level 2 routing.

Explanations for individual lines of output from Figure 2-158 follow.

The following lines show the statistical information available for Level 1 ISs:

```
ISIS-Stats: Compute L1 SPT, Timestamp 2780.328 seconds  
ISIS-Stats: Complete L1 SPT, Compute time 0.004, 1 nodes on SPT
```

The output indicates that the SPF algorithm was applied 2780.328 seconds after the system was up and configured. Given the existing intra-area topology, it took 4 milliseconds to place one Level 1 IS on the SPT.

The following lines show the statistical information available for Level 2 ISs:

```
ISIS-Stats: Compute L2 SPT, Timestamp 2780.3336 seconds  
ISIS-Stats: Complete L2 SPT, Compute time 0.056, 12 nodes on SPT
```

This output indicates that the SPF algorithm was applied 2780.3336 seconds after the system was up and configured. Given the existing intra-domain topology, it took 56 milliseconds to place 12 Level 2 ISs on the SPT.

## debug isis update-packets

Use the **debug isis update-packets** EXEC command to display various sequence number protocol data units (PDUs) and link state packets that are detected by a router. This router has been configured for IS-IS routing. The **no** form of this command disables debugging output.

**[no] debug isis update-packets**

### Sample Display

Figure 2-159 shows sample **debug isis update-packets** output.

**Figure 2-159 Sample Debug ISIS Update-Packets Output**

```
Router# debug isis update-packets

ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Updating L2 LSP
ISIS-Update: Delete link 888.8800.0181.00 from L2 LSP 1600.8906.4022.00-00, seq E
ISIS-Update: Updating L1 LSP
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Add link 8888.8800.0181.00 to L2 LSP 1600.8906.4022.00-00, new seq 10,
len 91
ISIS-Update: Sending L2 LSP 1600.8906.4022.00-00, seq 10, ht 1198 on Tunnel0
ISIS-Update: Sending L2 CSNP on Tunnel0
ISIS-Update: Updating L2 LSP
ISIS-Update: Rate limiting L2 LSP 1600.8906.4022.00-00, seq 11 (Tunnel0)
ISIS-Update: Updating L1 LSP
ISIS-Update: Rec L2 LSP 888.8800.0181.00.00-00 (Tunnel0)
ISIS-Update: PSNP entry 1600.8906.4022.00-00, seq 10, ht 1196
```

Explanations for individual lines of output from Figure 2-159 follow.

The following lines indicate that the router has sent a periodic Level 1 and Level 2 complete sequence number PDU on Ethernet 0:

```
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
```

The following lines indicate that the network service access point (NSAP) identified as 8888.8800.0181.00 was deleted from the Level 2 LSP 1600.8906.4022.00-00. The sequence number associated with this LSP is 0xE.

```
ISIS-Update: Updating L2 LSP
ISIS-Update: Delete link 888.8800.0181.00 from L2 LSP 1600.8906.4022.00-00, seq E
```

The following lines indicate that the NSAP identified as 8888.8800.0181.00 was added to the Level 2 LSP 1600.8906.4022.00-00. The new sequence number associated with this LSP is 0x10.

```
ISIS-Update: Updating L1 LSP
ISIS-Update: Sending L1 CSNP on Ethernet0
ISIS-Update: Sending L2 CSNP on Ethernet0
ISIS-Update: Add link 8888.8800.0181.00 to L2 LSP 1600.8906.4022.00-00, new seq 10,
len 91
```

The following line indicates that the router sent Level 2 LSP 1600.8906.4022.00-00 with sequence number 0x10 on Tunnel0:

```
ISIS-Update: Sending L2 LSP 1600.8906.4022.00-00, seq 10, ht 1198 on Tunnel0
```

The following lines indicates that a Level 2 LSP could not be transmitted because it was recently transmitted:

```
ISIS-Update: Sending L2 CSNP on Tunnel0
ISIS-Update: Updating L2 LSP
ISIS-Update: Rate limiting L2 LSP 1600.8906.4022.00-00, seq 11 (Tunnel0)
```

The following lines indicate that a Level 2 partial sequence number PDU (PSNP) has been received on Tunnel0:

```
ISIS-Update: Updating L1 LSP
ISIS-Update: Rec L2 PSNP from 8888.8800.0181.00 (Tunnel0)
```

The following line indicates that a Level 2 PSNP with an entry for Level 2 LSP 1600.8906.4022.00-00 has been received. This output is an acknowledgment that a previously sent LSP was received without an error.

```
ISIS-Update: PSNP entry 1600.8906.4022.00-00, seq 10, ht 1196
```

## debug kerberos

Use the **debug kerberos EXEC** command to display information associated with the Kerberos Authentication Subsystem. The **no** form of this command disables debugging output.

**[no] debug kerberos**

### Usage Guidelines

Kerberos is a security system that authenticates users and services without passing a cleartext password over the network. Cisco supports Kerberos under the Authentication, Authorization, and Accounting (AAA) security system.

Use the **debug aaa authentication** command to get a high-level view of login activity. When Kerberos is used on the router, you can use the **debug kerberos** command for more detailed debugging information.

### Sample Displays

Figure 2-160 shows part of the **debug aaa authentication** command output for a Kerberos login attempt that failed. The information indicates that Kerberos is the authentication method used.

**Figure 2-160 Sample Debug AAA Authentication Output—Kerberos Login Attempt**

```
Router# debug aaa authentication
AAA/AUTHEN/START (116852612): Method=KRB5
AAA/AUTHEN (116852612): status = GETUSER
AAA/AUTHEN/CONT (116852612): continue_login
AAA/AUTHEN (116852612): status = GETUSER
AAA/AUTHEN (116852612): Method=KRB5
AAA/AUTHEN (116852612): status = GETPASS
AAA/AUTHEN/CONT (116852612): continue_login
AAA/AUTHEN (116852612): status = GETPASS
AAA/AUTHEN (116852612): Method=KRB5
AAA/AUTHEN (116852612): password incorrect
AAA/AUTHEN (116852612): status = FAIL
```

Figure 2-161 shows part of the **debug kerberos** command output for a login attempt that was successful. The information indicates that the router sent a request to the KDC and received a valid credential.

**Figure 2-161 Sample Debug Kerberos Output—Successful Login**

```
Router# debug kerberos
Kerberos: Requesting TGT with expiration date of 820911631
Kerberos: Sent TGT request to KDC
Kerberos: Received TGT reply from KDC
Kerberos: Received valid credential with endtime of 820911631
```

Figure 2-162 shows part of the **debug kerberos** command output for a login attempt that failed. The information indicates that the router sent a request to the KDC and received a reply, but the reply did not contain a valid credential.

**Figure 2-162 Sample Debug Kerberos Output—Failed Login**

```
Router# debug kerberos

Kerberos: Requesting TGT with expiration date of 820911731
Kerberos: Sent TGT request to KDC
Kerberos: Received TGT reply from KDC
Kerberos: Received invalid credential.
AAA/AUTHEN (425003829): password incorrect
```

Figure 2-163 shows other failure messages you might see that indicate a configuration problem. The first message indicates the router failed to find the default Kerberos realm, therefore the process failed to build a message to send to the KDC. The second message indicates the router failed to retrieve its own IP address. The third message indicates the router failed to retrieve the current time. The fourth message indicates the router failed to find or create a credentials cache for a user, which is usually caused by low memory availability.

**Figure 2-163 Sample Debug Kerberos Output—Miscellaneous Messages**

```
Router# debug kerberos

Kerberos: authentication failed when parsing name

Kerberos: authentication failed while getting my address

Kerberos: authentication failed while getting time of day

Kerberos: authentication failed while allocating credentials cache
```

#### Related Command

**debug aaa authentication**

## debug lane client

Use the **debug lane client** EXEC command to display information about a LAN Emulation Client (LEC). The **no** form of this command disables debugging output.

```
[no] debug lane client {all | le-arp | packet | signaling | state | topology} [interface interface]
```

### Syntax Description

<b>all</b>	Displays all debug information related to the LEC.
<b>le-arp</b>	Displays debug information related to the LANE ARP table.
<b>packet</b>	Displays debug information about each packet.
<b>signaling</b>	Displays debug information related to client SVCs.
<b>state</b>	Displays debug information when the state changes.
<b>topology</b>	Displays debug information related to the topology of the emulated LAN.
<b>interface</b> <i>interface</i>	(Optional) Limits the debugging output to messages that relate to a particular interface or subinterface. If you enter this command multiple times with different interfaces, the last interface entered will be the one used to filter the messages.

### Usage Guidelines

The **debug lane client all** command can generate a large amount of output. Use a limiting keyword or specify a subinterface to decrease the amount of output and focus on the information you need.

### Sample Displays

Figure 2-164 shows sample output for **debug lane client packet** and **debug lane client state** commands for an LEC joining an emulated LAN (ELAN) called *elan1*.

**Figure 2-164 Sample Debug LANE Client Output—Client Joining ELAN**

```
Router# debug lane client packet
Router# debug lane client state
```

The LEC listens for signaling calls to its ATM address (Initial State).

```
LEC ATM2/0.1: sending LISTEN
LEC ATM2/0.1: listen on 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1: received LISTEN
```

The LEC calls the LAN Emulation Configuration Server (LECS) and attempts to set up the Configure Direct VC (LECS Connect Phase).

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1: callid 0x6114D174
LEC ATM2/0.1: called party 39.020304050607080910111213.00000CA05B43.00
LEC ATM2/0.1: calling_party 39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the LECS. The Configure Direct VC is established.

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:  callid          0x6114D174
LEC ATM2/0.1:  vcd            148
```

The LEC sends a CONFIG REQUEST to the LECS on the Configure Direct VC (Configuration Phase).

```
LEC ATM2/0.1: sending LANE_CONFIG_REQ on VCD 148
LEC ATM2/0.1:  SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:  SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  LAN Type       2
LEC ATM2/0.1:  Frame size     2
LEC ATM2/0.1:  LAN Name       elan1
LEC ATM2/0.1:  LAN Name size  5
```

The LEC receives a CONFIG RESPONSE from the LECS on the Configure Direct VC.

```
LEC ATM2/0.1: received LANE_CONFIG_RSP on VCD 148
LEC ATM2/0.1:  SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:  SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  LAN Type       2
LEC ATM2/0.1:  Frame size     2
LEC ATM2/0.1:  LAN Name       elan1
LEC ATM2/0.1:  LAN Name size  5
```

The LEC releases the Configure Direct VC.

```
LEC ATM2/0.1: sending RELEASE
LEC ATM2/0.1:  callid          0x6114D174
LEC ATM2/0.1:  cause code     31
```

The LEC receives a RELEASE\_COMPLETE from the LECS.

```
LEC ATM2/0.1: received RELEASE_COMPLETE
LEC ATM2/0.1:  callid          0x6114D174
LEC ATM2/0.1:  cause code     16
```

The LEC calls the LAN Emulation Server (LES) and attempts to set up the Control Direct VC (Join/Registration Phase).

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:  callid          0x61167110
LEC ATM2/0.1:  called party   39.020304050607080910111213.00000CA05B41.01
LEC ATM2/0.1:  calling_party  39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the LES. The Control Direct VC is established.

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:  callid          0x61167110
LEC ATM2/0.1:  vcd            150
```

The LEC sends a JOIN REQUEST to the LES on the Control Direct VC.

```
LEC ATM2/0.1: sending LANE_JOIN_REQ on VCD 150
LEC ATM2/0.1:  Status         0
LEC ATM2/0.1:  LECID          0
LEC ATM2/0.1:  SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:  SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  LAN Type       2
LEC ATM2/0.1:  Frame size     2
LEC ATM2/0.1:  LAN Name       elan1
LEC ATM2/0.1:  LAN Name size  5
```

The LEC receives a SETUP request from the LES to set up the Control Distribute VC.

```
LEC ATM2/0.1: received SETUP
LEC ATM2/0.1:  callid      0x6114D174
LEC ATM2/0.1:  called party 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  calling_party 39.020304050607080910111213.00000CA05B41.01
```

The LEC responds to the LES call setup with a CONNECT.

```
LEC ATM2/0.1: sending CONNECT
LEC ATM2/0.1:  callid      0x6114D174
LEC ATM2/0.1:  vcd          151
```

A CONNECT\_ACK is received from the ATM switch. The Control Distribute VC is established.

```
LEC ATM2/0.1: received CONNECT_ACK
```

The LEC receives a JOIN response from the LES on the Control Direct VC.

```
LEC ATM2/0.1: received LANE_JOIN_RSP on VCD 150
LEC ATM2/0.1:  Status      0
LEC ATM2/0.1:  LECID       1
LEC ATM2/0.1:  SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:  SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  LAN Type      2
LEC ATM2/0.1:  Frame size    2
LEC ATM2/0.1:  LAN Name      elan1
LEC ATM2/0.1:  LAN Name size 5
```

The LEC sends an LE\_ARP request to the LES to obtain the broadcast-and-unknown (BUS) ATM NSAP address (BUS Connect).

```
LEC ATM2/0.1: sending LANE_ARP_REQ on VCD 150
LEC ATM2/0.1:  SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:  SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  TARGET MAC address ffff.ffff.ffff
LEC ATM2/0.1:  TARGET ATM address 00.0000000000000000000000000000.000000000000.00
```

The LEC receives its own LE\_ARP request via the LES over the Control Distribute VC.

```
LEC ATM2/0.1: received LANE_ARP_RSP on VCD 151
LEC ATM2/0.1:  SRC MAC address 0000.0ca0.5b40
LEC ATM2/0.1:  SRC ATM address 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  TARGET MAC address ffff.ffff.ffff
LEC ATM2/0.1:  TARGET ATM address 39.020304050607080910111213.00000CA05B42.01
```

The LEC calls the BUS and attempts to set up the Multicast Send VC.

```
LEC ATM2/0.1: sending SETUP
LEC ATM2/0.1:  callid      0x6114D354
LEC ATM2/0.1:  called party 39.020304050607080910111213.00000CA05B42.01
LEC ATM2/0.1:  calling_party 39.020304050607080910111213.00000CA05B40.01
```

The LEC receives a CONNECT response from the BUS. The Multicast Send VC is established.

```
LEC ATM2/0.1: received CONNECT
LEC ATM2/0.1:  callid      0x6114D354
LEC ATM2/0.1:  vcd          153
```

The LEC receives a SETUP request from the BUS to set up the Multicast Forward VC.

```
LEC ATM2/0.1: received SETUP
LEC ATM2/0.1:  callid      0x610D4230
LEC ATM2/0.1:  called party 39.020304050607080910111213.00000CA05B40.01
LEC ATM2/0.1:  calling_party 39.020304050607080910111213.00000CA05B42.01
```

The LEC responds to the BUS call setup with a CONNECT.

```
LEC ATM2/0.1: sending CONNECT
LEC ATM2/0.1:  callid          0x610D4230
LEC ATM2/0.1:  vcd            154
```

A CONNECT\_ACK is received from the ATM switch. The Multicast Forward VC is established.

```
LEC ATM2/0.1: received CONNECT_ACK
```

The LEC moves into the OPERATIONAL state.

```
%LANE-5-UPDOWN: ATM2/0.1 elan elan1: LE Client changed state to up
```

The following output is the from **show lane client** command after the LEC joins the emulated LAN as shown in the **debug lane client** output:

```
Router# show lane client

LE Client ATM2/0.1 ELAN name: elan1 Admin: up State: operational
Client ID: 1 LEC up for 1 minute 2 seconds
Join Attempt: 1
HW Address: 0000.0ca0.5b40 Type: token ring Max Frame Size: 4544
Ring:1 Bridge:1 ELAN Segment ID: 2048
ATM Address: 39.020304050607080910111213.00000CA05B40.01

VCD  rxFrames  txFrames  Type      ATM Address
  0      0         0  configure 39.020304050607080910111213.00000CA05B43.00
142    1         2  direct   39.020304050607080910111213.00000CA05B41.01
143    1         0  distribute 39.020304050607080910111213.00000CA05B41.01
145    0         0  send     39.020304050607080910111213.00000CA05B42.01
146    1         0  forward  39.020304050607080910111213.00000CA05B42.01
```

Figure 2-165 shows sample **debug lane client all** output when an interface with an LECS, an LES/BUS, and an LEC is shut down.

**Figure 2-165 Sample Debug LANE Client Output—Interface Shutdown**

```
Router# debug lane client all

LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2:  callid0x60E8B474
LEC ATM1/0.2:  cause code0
LEC ATM1/0.2:  action A_PROCESS_REL_COMP
LEC ATM1/0.2:  action A_TEARDOWN_LEC
LEC ATM1/0.2:  sending RELEASE
LEC ATM1/0.2:  callid0x60EB6160
LEC ATM1/0.2:  cause code31
LEC ATM1/0.2:  sending RELEASE
LEC ATM1/0.2:  callid0x60EB7548
LEC ATM1/0.2:  cause code31
LEC ATM1/0.2:  sending RELEASE
LEC ATM1/0.2:  callid0x60EB9E48
LEC ATM1/0.2:  cause code31
LEC ATM1/0.2:  sending CANCEL
LEC ATM1/0.2:  ATM address47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2:  state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3:  callid0x60E8D108
LEC ATM1/0.3:  cause code0
LEC ATM1/0.3:  action A_PROCESS_REL_COMP
LEC ATM1/0.3:  action A_TEARDOWN_LEC
LEC ATM1/0.3:  sending RELEASE
```

```
LEC ATM1/0.3: callid0x60EB66D4
LEC ATM1/0.3: cause code31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3: callid0x60EB7B8C
LEC ATM1/0.3: cause code31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3: callid0x60EBA3BC
LEC ATM1/0.3: cause code31
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3: ATM address47.0091810000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2: callid0x60EB7548
LEC ATM1/0.2: cause code0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3: callid0x60EB7B8C
LEC ATM1/0.3: cause code0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1: callid0x60EBC458
LEC ATM1/0.1: cause code0
LEC ATM1/0.1: action A_PROCESS_REL_COMP
LEC ATM1/0.1: action A_TEARDOWN_LEC
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1: callid0x60EBD30C
LEC ATM1/0.1: cause code31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1: callid0x60EBDD28
LEC ATM1/0.1: cause code31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1: callid0x60EBF174
LEC ATM1/0.1: cause code31
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1: ATM address47.0091810000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1: callid0x60EBDD28
LEC ATM1/0.1: cause code0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2: callid0x60EB6160
LEC ATM1/0.2: cause code0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3: callid0x60EB66D4
LEC ATM1/0.3: cause code0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2: callid0x60EB9E48
LEC ATM1/0.2: cause code0
LEC ATM1/0.2: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.2: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.3: received RELEASE_COMPLETE
LEC ATM1/0.3: callid0x60EBA3BC
LEC ATM1/0.3: cause code0
LEC ATM1/0.3: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.3: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1: callid0x60EBD30C
```

```
LEC ATM1/0.1: cause code0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1: callid0x60EBF174
LEC ATM1/0.1: cause code0
LEC ATM1/0.1: action A_PROCESS_TERM_REL_COMP
LEC ATM1/0.1: state TERMINATING event LEC_SIG_RELEASE_COMP => IDLE
LEC ATM1/0.2: received CANCEL
LEC ATM1/0.2: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.3: received CANCEL
LEC ATM1/0.3: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.1: received CANCEL
LEC ATM1/0.1: state IDLE event LEC_SIG_CANCEL => IDLE
LEC ATM1/0.1: action A_SHUTDOWN_LEC
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1: ATM address47.00918100000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
LEC ATM1/0.2: action A_SHUTDOWN_LEC
LEC ATM1/0.2: sending CANCEL
LEC ATM1/0.2: ATM address47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
LEC ATM1/0.3: action A_SHUTDOWN_LEC
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3: ATM address47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state IDLE event LEC_LOCAL_DEACTIVATE => IDLE
```

## debug lane config

Use the **debug lane config** EXEC command to display information about a LANE configuration server. The **no** form of this command disables debugging output.

**[no] debug lane config {all | events | packets}**

### Syntax Description

<b>all</b>	Display all debug messages related to the LANE configuration server. The output includes both the <b>events</b> and <b>packets</b> types of output.
<b>events</b>	Display only messages related to significant LANE configuration server events.
<b>packets</b>	Display information on each packet sent or received by the LANE configuration server.

### Usage Guidelines

The **debug lane config** output is intended to be used primarily by a Cisco technical support representative.

### Sample Display

Figure 2-166 shows sample **debug lane config all** output when an interface with an LECS, an LES/BUS, and an LEC is shut down.

**Figure 2-166 Sample Debug LANE Config All Output**

```
Router# debug lane config all

LECS EVENT ATM1/0: processing interface down transition
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: elan 'xxx' LES 47.00918100000000613E5A2F01.006070174821.01 callId
0x60CE0F58 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE0F58 cause 31
LECS EVENT ATM1/0: elan 'yyy' LES 47.00918100000000613E5A2F01.006070174821.02 callId
0x60CE2104 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2104 cause 31
LECS EVENT ATM1/0: elan 'zzz' LES 47.00918100000000613E5A2F01.006070174821.03 callId
0x60CE2DC8 deliberately being disconnected
LECS EVENT ATM1/0: sending RELEASE for call 0x60CE2DC8 cause 31
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT ATM1/0: placed de-register address 0x60EC4F28
(47.007900000000000000000000.00A03E000001.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
```

## debug lane config

---

```
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE0F58 cause 0
LECS EVENT ATM1/0: call 0x60CE0F58 cleaned up
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2104 cause 0
LECS EVENT ATM1/0: call 0x60CE2104 cleaned up
LECS EVENT ATM1/0: ATM_RELEASE_COMPLETE received: callId 0x60CE2DC8 cause 0
LECS EVENT ATM1/0: call 0x60CE2DC8 cleaned up
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: UNKNOWN/UNSET: signalling DE-registered
LECS EVENT ATM1/0: placed de-register address 0x60E8A824
(47.00918100000000613E5A2F01.006070174823.00) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: placed de-register address 0x60EC5C08
(47.00918100000000613E5A2F01.006070174823.99) request with signalling
LECS EVENT ATM1/0: ilmiDeRegisterAddress: sendSetRequestToILMI failure; interface down ?
LECS EVENT ATM1/0: tearing down all connexions
LECS EVENT ATM1/0: All calls to/from LECSs are being released
LECS EVENT: config server 56 killed
```

## debug lane finder

Use the **debug lane finder** EXEC command to display information about the finder internal state machine. The **no** form of this command disables debugging output.

**[no] debug lane finder**

### Usage Guidelines

The **debug lane finder** output is intended to be used primarily by a Cisco technical support representative.

### Sample Display

Figure 2-167 shows sample **debug lane finder** output when an interface with an LECS, an LES/BUS, and an LEC is shut down.

**Figure 2-167 Sample Debug LANE Finder Output**

```
Router# debug lane finder

LECS FINDER AT1/0.3: user request 1819 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER AT1/0: finder state machine started
LECS FINDER AT1/0: time to perform a getNext on the ILMI
LECS FINDER AT1/0: LECS 47.00918100000000613E5A2F01.006070174823.00 deleted
LECS FINDER AT1/0: ilmi_client_request failed, answering all users
LECS FINDER AT1/0: answering all requests now
LECS FINDER AT1/0: responded to user request 1819
LECS FINDER AT1/0: number of remaining requests still to be processed: 0
LECS FINDER AT1/0.2: user request 1820 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER AT1/0: finder state machine started
LECS FINDER AT1/0: time to perform a getNext on the ILMI
LECS FINDER AT1/0: ilmi_client_request failed, answering all users
LECS FINDER AT1/0: answering all requests now
LECS FINDER AT1/0: responded to user request 1820
LECS FINDER AT1/0: number of remaining requests still to be processed: 0
LECS FINDER AT1/0.1: user request 1821 of type GET_MASTER_LECS_ADDRESS queued up
LECS FINDER AT1/0: finder state machine started
LECS FINDER AT1/0: time to perform a getNext on the ILMI
LECS FINDER AT1/0: ilmi_client_request failed, answering all users
LECS FINDER AT1/0: answering all requests now
LECS FINDER AT1/0: responded to user request 1821
LECS FINDER AT1/0: number of remaining requests still to be processed: 0
```

## debug lane server

Use the **debug lane server** EXEC command to display information about a LANE server. The **no** form of this command disables debugging output.

**[no] debug lane server [interface interface]**

### Syntax Description

**interface interface** (Optional) Limits the debugging output to messages that relate to a particular interface or subinterface. If you enter this command multiple times with different interfaces, the last interface entered will be the one used to filter debug messages.

### Usage Guidelines

The **debug lane server** output is intended to be used primarily by a Cisco technical support representative.

The **debug lane server** command can generate a large amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

### Sample Display

Figure 2-168 shows sample **debug lane server** output when an interface with an LECS, an LES/BUS, and an LEC is shut down.

**Figure 2-168 Sample Debug LANE Server Output**

```
Router# debug lane server

LES ATM1/0.1: lsv_lecsAccessSigCB called with callId 0x60CE124C, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.1: disconnected from the master LECS
LES ATM1/0.1: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: lsv_lecsAccessSigCB called with callId 0x60CE29E0, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.2: disconnected from the master LECS
LES ATM1/0.2: should have been connected, will reconnect in 3 seconds
LES ATM1/0.3: lsv_lecsAccessSigCB called with callId 0x60EB1940, opcode
ATM_RELEASE_COMPLETE
LES ATM1/0.3: disconnected from the master LECS
LES ATM1/0.3: should have been connected, will reconnect in 3 seconds
LES ATM1/0.2: elan yyy client 1 lost control distribute
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 state change Oper -> Term
LES ATM1/0.3: elan zzz client 1 lost control distribute
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 state change Oper -> Term
LES ATM1/0.2: elan yyy client 1 lost MC forward
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 lost MC forward
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 lost control distribute
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 state change Oper -> Term
LES ATM1/0.1: elan xxx client 1 lost MC forward
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
```

```
LES ATM1/0.2: elan yyy client 1 released control direct
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1 released control direct
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1 MC forward released
LES ATM1/0.2: elan yyy client 1: lsv_kill_client called
LES ATM1/0.2: elan yyy client 1: freeing client structures
LES ATM1/0.2: elan yyy client 1 unregistered 0060.7017.4820
LES ATM1/0.2: elan yyy client 1 destroyed
LES ATM1/0.3: elan zzz client 1 MC forward released
LES ATM1/0.3: elan zzz client 1: lsv_kill_client called
LES ATM1/0.3: elan zzz client 1: freeing client structures
LES ATM1/0.3: elan zzz client 1 unregistered 0060.7017.4820
LES ATM1/0.3: elan zzz client 1 destroyed
LES ATM1/0.1: elan xxx client 1 released control direct
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1 MC forward released
LES ATM1/0.1: elan xxx client 1: lsv_kill_client called
LES ATM1/0.1: elan xxx client 1: freeing client structures
LES ATM1/0.1: elan xxx client 1 unregistered 0060.7017.4820
LES ATM1/0.1: elan xxx client 1 destroyed
LES ATM1/0.1: elan xxx major interface state change
LES ATM1/0.1: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.1: shutting down
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: elan xxx: LES/BUS state change operational -> terminating
LES ATM1/0.1: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy major interface state change
LES ATM1/0.2: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.2: shutting down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: elan yyy: LES/BUS state change operational -> terminating
LES ATM1/0.2: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz major interface state change
LES ATM1/0.3: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.3: shutting down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: elan zzz: LES/BUS state change operational -> terminating
LES ATM1/0.3: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: lsv_kill_lesbus called
LES ATM1/0.1: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.1: elan xxx: stopped listening on addresses
LES ATM1/0.1: elan xxx: all clients killed
LES ATM1/0.1: elan xxx: multicast groups killed
LES ATM1/0.1: elan xxx: addresses de-registered from ilmi
LES ATM1/0.1: elan xxx: LES/BUS state change terminating -> down
LES ATM1/0.1: elan xxx: administratively down
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: lsv_kill_lesbus called
LES ATM1/0.2: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.2: elan yyy: stopped listening on addresses
LES ATM1/0.2: elan yyy: all clients killed
LES ATM1/0.2: elan yyy: multicast groups killed
LES ATM1/0.2: elan yyy: addresses de-registered from ilmi
LES ATM1/0.2: elan yyy: LES/BUS state change terminating -> down
LES ATM1/0.2: elan yyy: administratively down
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: lsv_kill_lesbus called
LES ATM1/0.3: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.3: elan zzz: stopped listening on addresses
LES ATM1/0.3: elan zzz: all clients killed
```

## debug lane server

---

```
LES ATM1/0.3: elan zzz: multicast groups killed
LES ATM1/0.3: elan zzz: addresses de-registered from ilmi
LES ATM1/0.3: elan zzz: LES/BUS state change terminating -> down
LES ATM1/0.3: elan zzz: administratively down
LES ATM1/0.3: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.2: cleanupLeCsAccess: discarding all validation requests
LES ATM1/0.1: cleanupLeCsAccess: discarding all validation requests
```

## debug lane signaling

Use the **debug lane signaling EXEC** command to display information about LANE Server and BUS switched virtual circuits (SVCs). The **no** form of this command disables debugging output.

**[no] debug lane signaling [interface interface]**

### Syntax Description

**interface interface** (Optional) Limits the debugging output to messages that relate to a particular interface or subinterface. If you enter this command multiple times with different interfaces, the last interface entered will be the one used to filter debug messages.

### Usage Guidelines

The **debug lane signaling** output is intended to be used primarily by a Cisco technical support representative.

The **debug lane signaling** command can generate a large amount of output. Specify a subinterface to decrease the amount of output and focus on the information you need.

### Sample Display

Figure 2-169 shows sample **debug lane signaling** output when an interface with an LECS, an LES/BUS, and an LEC is shut down.

**Figure 2-169 Sample Debug LANE Signaling Output**

```
Router# debug lane signaling

LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB565C cause 0 lv 0x60E8D348
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D348 from mcg 0x60E97E84
LANE SIG ATM1/0.2: timer for lv 0x60E8D348 stopped
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D468 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D3D8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: sent ATM_RELEASE request for lv 0x60E8D2B8 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB5CA0 cause 0 lv 0x60E8BEF4
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8BEF4 from mcg 0x60E9A37C
LANE SIG ATM1/0.3: timer for lv 0x60E8BEF4 stopped
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8C014 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BF84 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.3: sent ATM_RELEASE request for lv 0x60E8BE64 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB9040 cause 0 lv 0x60E8D468
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.2: lane_sig_mc_release: breaking lv 0x60E8D468 from mcg 0x60E97EC8
LANE SIG ATM1/0.2: timer for lv 0x60E8D468 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB97D4 cause 0 lv 0x60E8C014
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.3: lane_sig_mc_release: breaking lv 0x60E8C014 from mcg 0x60E9A3C0
LANE SIG ATM1/0.3: timer for lv 0x60E8C014 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBCEB8 cause 0 lv 0x60EBBAF0
lvstate LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBAF0 from mcg 0x60E8F51C
LANE SIG ATM1/0.1: timer for lv 0x60EBBAF0 stopped
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBC10 in state LANE_VCC_CONNECTED
```

```
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBB80 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: sent ATM_RELEASE request for lv 0x60EBBA60 in state LANE_VCC_CONNECTED
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBEB00 cause 0 lv 0x60EBBC10
lvstate LANE_VCC_DROP_SENT
LANE SIG ATM1/0.1: lane_sig_mc_release: breaking lv 0x60EBBC10 from mcg 0x60E8F560
LANE SIG ATM1/0.1: timer for lv 0x60EBBC10 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60E8B174 cause 0 lv 0x60E8D2B8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D2B8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60E8B990 cause 0 lv 0x60E8BE64
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BE64 stopped
LANE SIG ATM1/0.2: received ATM_RELEASE_COMPLETE callid 0x60EB7FE0 cause 0 lv 0x60E8D3D8
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.2: timer for lv 0x60E8D3D8 stopped
LANE SIG ATM1/0.3: received ATM_RELEASE_COMPLETE callid 0x60EB8554 cause 0 lv 0x60E8BF84
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.3: timer for lv 0x60E8BF84 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBB6D4 cause 0 lv 0x60EBBA60
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBA60 stopped
LANE SIG ATM1/0.1: received ATM_RELEASE_COMPLETE callid 0x60EBE24C cause 0 lv 0x60EBBB80
lvstate LANE_VCC_RELEASE_SENT
LANE SIG ATM1/0.1: timer for lv 0x60EBBB80 stopped
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.2: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.3: sent ATM_CANCEL_NSAP request for lv 0x0 in state NULL_VCC_POINTER
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.1: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.2: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
LANE SIG ATM1/0.3: received ATM_CANCEL_NSAP for nsap
00.000000000000050000000000.000000000000.00
```

## debug lapb

Use the **debug lapb** EXEC command to display all traffic for interfaces using Link Access Procedure, Balanced (LAPB) encapsulation. The **no** form of this command disables debugging output.

**[no] debug lapb**

### Usage Guidelines

This command displays information on the X.25 Layer 2 protocol. It is useful to users who are familiar with the LAPB protocol.

You can use the **debug lapb** command to determine why X.25 interfaces or LAPB connections are going up and down. It is also useful for identifying link problems, as evidenced when **show interfaces** command displays a high number of rejects or frame errors over the X.25 link.



**Caution** Because the **debug lapb** command generates a lot of output, use it when the aggregate of all LAPB traffic on X.25 and LAPB interfaces is fewer than five frames per second.

### Sample Display

Figure 2-170 shows sample **debug lapb** output. (The numbers 1 through 7 at the top of the display have been added in order to aid documentation.)

**Figure 2-170 Sample Debug LAPB Output**

```

1          2 3 4 5 6 7
Serial0: LAPB I CONNECT (5) IFRAME P 2 1
Serial0: LAPB O REJSENT (2) REJ F 3
Serial0: LAPB O REJSENT (5) IFRAME 0 3
Serial0: LAPB I REJSENT (2) REJ (C) 7
Serial0: LAPB I DISCONNECT (2) SABM P
Serial0: LAPB O CONNECT (2) UA F
Serial0: LAPB O CONNECT (5) IFRAME 0 0
Serial0: LAPB T1 CONNECT 357964 0

```

In Figure 2-170 each line of output describes a LAPB event. There are two types of LAPB events: frame events (when a frame enters or exits the LAPB) and timer events. In Figure 2-170, the last line describes a timer event; all of the other lines describe frame events. Table 2-83 describes the first seven fields shown in Figure 2-170.

**Table 2-83 Debug LAPB Field Descriptions**

Field	Description
First field (1)	Interface type and unit number reporting the frame event.
Second field (2)	Protocol providing the information.

**Table 2-83      Debug LAPB Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
Third field (3)	<p>Frame event type. Possible values follow:</p> <ul style="list-style-type: none"> <li>• I—Frame input</li> <li>• O—Frame output</li> <li>• T1—T1 timer expired</li> <li>• T3—Interface outage timer expired</li> <li>• T4—Idle link timer expired</li> </ul>
Fourth field (4)	<p>State of the protocol when the frame event occurred. Possible values follow:</p> <ul style="list-style-type: none"> <li>• BUSY (RNR frame received)</li> <li>• CONNECT</li> <li>• DISCONNECT</li> <li>• DISCSENT (disconnect sent)</li> <li>• ERROR (FRMR frame sent)</li> <li>• REJSENT (reject frame sent)</li> <li>• SABMSENT (SABM frame sent)</li> </ul>
Fifth field (5)	<p>In a frame event, this value is the size of the frame (in bytes). In a timer event, this value is the current timer value (in milliseconds).</p>
Sixth field (6)	<p>In a frame event, this value is the frame type name. Possible values for frame type names follow:</p> <ul style="list-style-type: none"> <li>• DISC—Disconnect</li> <li>• DM—Disconnect mode</li> <li>• FRMR—Frame reject</li> <li>• IFRAME—Information frame</li> <li>• ILLEGAL—Illegal LAPB frame</li> <li>• REJ—Reject</li> <li>• RNR—Receiver not ready</li> <li>• RR—Receiver ready</li> <li>• SABM—Set asynchronous balanced mode</li> <li>• SABME—Set asynchronous balanced mode, extended</li> <li>• UA—Unnumbered acknowledgment</li> </ul> <p>In a T1 timer event, this value is the number of retransmissions already attempted.</p>

**Table 2-83 Debug LAPB Field Descriptions (Continued)**

Field	Description
Seventh field (7) (This field will not print if the frame control field is required to appear as either a command or a response, and that frame type is correct.)	<p>This field is only present in frame events. It describes the frame type identified by the LAPB address and Poll/Final bit. Possible values are as follows:</p> <ul style="list-style-type: none"> <li>• (C)—Command frame</li> <li>• (R)—Response frame</li> <li>• P—Command/Poll frame</li> <li>• F—Response/Final frame</li> <li>• /ERR—Command/Response type is invalid for the control field. An ?ERR generally means that the DTE/DCE assignments are not correct for this link.</li> <li>• BAD-ADDR—Address field is neither Command nor Response</li> </ul>

A timer event only displays the first six fields of **debug lapb** output. For frame events, however, the fields that follow the sixth field document the LAPB control information present in the frame. Depending on the value of the frame type name shown in the sixth field, these fields may or may not appear. Descriptions of the fields following the first six fields shown in Figure 2-170 follow.

After the Poll/Final indicator, depending on the frame type, three different types of LAPB control information can be printed.

For information frames, the value of the N(S) field and the N(R) field will be printed. The N(S) field of an information frame is the sequence number of that frame, so this field will rotate between 0 and 7 for (modulo 8 operation) or 0 and 127 (for modulo 128 operation) for successive outgoing information frames and (under normal circumstances) also will rotate for incoming information frame streams. The N(R) field is a “piggybacked” acknowledgment for the incoming information frame stream; it informs the other end of the link what sequence number is expected next.

RR, RNR, and REJ frames have an N(R) field, so the value of that field is printed. This field has exactly the same significance that it does in an information frame.

For the FRMR frame, the error information is decoded to display the rejected control field, V(R) and V(S) values, the Response/Command flag, and the error flags WXYZ.

In the following example, the output shows an idle link timer action (T4) where the timer expires twice on an idle link, with the value of T4 set to five seconds:

```
Serial2: LAPB T4 CONNECT 255748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
Serial2: LAPB T4 CONNECT 260748
Serial2: LAPB O CONNECT (2) RR P 5
Serial2: LAPB I CONNECT (2) RR F 5
```

The next example shows an interface outage timer expiration (T3):

```
Serial2: LAPB T3 DISCONNECT 273284
```

The following example output shows an error condition when no DCE to DTE connection exists. Note that if a frame has only one valid type (for example, a SABM can only be a command frame), a received frame that has the wrong frame type will be flagged as a receive error (R/ERR in the following output). This feature makes misconfigured links (DTE-DTE or DCE-DCE) easy to spot. Other, less common errors will be highlighted too, such as a too-short or too-long frame, or an invalid address (neither command nor response).

```
Serial2: LAPB T1 SABMSENT 1026508 1
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
Serial2: LAPB T1 SABMSENT 1029508 2
Serial2: LAPB O SABMSENT (2) SABM P
Serial2: LAPB I SABMSENT (2) SABM (R/ERR)
```

The output in the next example shows the router is misconfigured and has a standard (modulo 8) interface connected to an extended (modulo 128) interface. This condition is indicated by the SABM balanced mode and SABME balanced mode extended messages appearing on the same interface.

```
Serial2: LAPB T1 SABMSENT 1428720 0
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
Serial2: LAPB T1 SABMSENT 1431720 1
Serial2: LAPB O SABMSENT (2) SABME P
Serial2: LAPB I SABMSENT (2) SABM P
```

## debug lat packet

Use the **debug lat packet** EXEC command to display information on all LAT events. The **no** form of this command disables debugging output.

**[no] debug lat packet**

### Usage Guidelines

For each datagram (packet) received or transmitted, a message is logged to the console.

---

**Note** This command severely impacts LAT performance and is intended for troubleshooting use only.

---

### Sample Display

Figure 2-171 shows sample **debug lat packet** output.

**Figure 2-171 Sample Debug LAT Packet Output**

```
Router# debug lat packet

LAT: I int=Ethernet0, src=0000.0c01.0509, dst=0900.2b00.000f, type=0, M=0, R=0
LAT: I int=Ethernet0, src=0800.2b11.2d13, dst=0000.0c01.7876, type=A, M=0, R=0
LAT: O dst=0800.2b11.2d13, int=Ethernet0, type= A, M=0, R=0, len= 20, next 0 ref 1
```

The second line of output in Figure 2-171 describes a packet that is input to the router. Table 2-84 describes the fields in this line.

**Table 2-84 Debug LAT Packet Field Descriptions**

Field	Description
LAT:	Indicates that this display shows LAT debugging output.
I	Indicates that this line of output describes a packet that is input to the router (I) or output from the router (O).
int = Ethernet0	Indicates the interface on which the packet event took place.
src = 0800.2b11.2d13	Indicates the source address of the packet.
dst = 0000.0c01.7876	Indicates the destination address of the packet.
type = A	Indicates the message type (in hexadecimal). Possible values are as follows: <ul style="list-style-type: none"> <li>• 0 = Run Circuit</li> <li>• 1 = Start Circuit</li> <li>• 2 = Stop Circuit</li> <li>• A = Service Announcement</li> <li>• C = Command</li> <li>• D = Status</li> <li>• E = Solicit Information</li> <li>• F = Response Information</li> </ul>

The third line of output in Figure 2-171 describes a packet that is output from the router. Table 2-85 describes the last three fields in this line.

**Table 2-85      Debug LAT Packet Field Descriptions**

<b>Field</b>	<b>Description</b>
len= 20	Indicates the length (hexadecimal) of the packet in bytes.
next 0	Indicates the link on transmit queue.
ref 1	Indicates the count of packet users.

## debug lex rcmd

Use the **debug lex rcmd EXEC** command to debug LAN Extender remote commands. The **no** form of this command disables debugging output.

**[no] debug lex rcmd**

### Sample Display

Figure 2-172 shows sample **debug lex rcmd** output.

**Figure 2-172 Sample Debug LEX Rcmd Output**

```
Router# debug lex rcmd

LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
LEX-RCMD: REJ received
LEX-RCMD: illegal CODE field received in header: <number>
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
LEX-RCMD: Lex0 is not bound to a serial interface
LEX-RCMD: encapsulation failure
LEX-RCMD: timeout for Lex0: "lex priority-group" command
LEX-RCMD: re-transmitting Lex0: "lex priority-group" command
LEX-RCMD: lex_setup_and_send called with invalid parameter
LEX-RCMD: bind occurred on shutdown LEX interface
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
LEX-RCMD: No active Lex interface found for unbind
```

Explanations for individual lines of output from Figure 2-172 follow.

The following output indicates that a LAN Extender remote command packet was received on a serial interface that is not bound to a LAN Extender interface:

```
LEX-RCMD: "shutdown" command received on unbound serial interface- Serial0
```

This message can occur for any of the LAN Extender remote commands. Possible causes of this message are as follows:

- FLEX state machine software error
- Serial line momentarily goes down, which is detected by the host but not by FLEX

The following output indicates that a LAN Extender remote command response has been received. The hexadecimal values are for internal use only.

```
LEX-RCMD: Lex0: "inventory" command received
Rcvd rcmd: FF 03 80 41 41 13 00 1A 8A 00 00 16 01 FF 00 00
Rcvd rcmd: 00 02 00 00 07 5B CD 15 00 00 0C 01 15 26
```

The following output indicates that when the host router originates a LAN Extender remote command to FLEX, it generates an 8-bit identifier which is used to associate a command with its corresponding response:

```
LEX-RCMD: ACK or response received on Serial0 without a corresponding ID
```

This message could be displayed for any of the following reasons:

- FLEX was very busy at the time that the command arrived and could not send an immediate response. The command timed out on the host router and then FLEX finally sent the response.
- Transmission error.
- Software error.

Possible responses to Config-Request are Config-ACK, Config-NAK, and Config-Rej. The following output shows that some of the options in the Config-Request are not recognizable or are not acceptable to FLEX due to transmission errors or software errors:

```
LEX-RCMD: REJ received
```

The following output shows that a LAN Extender remote command response was received but that the CODE field in the header was incorrect:

```
LEX-RCMD: illegal CODE field received in header: <number>
```

The following output indicates that a LAN Extender remote command response was received but that it had an incorrect length field. This message can occur for any of the LAN Extender remote commands.

```
LEX-RCMD: illegal length for Lex0: "lex input-type-list"
```

The following output shows that a host router was about to send a remote command when the serial link went down:

```
LEX-RCMD: Lex0 is not bound to a serial interface
```

The following output shows that the serial interface's encapsulation routine failed to encapsulate the remote command datagram because the LEX-NCP was not in the OPEN state. Due to the way the PPP state machine is implemented, it is normal to see a single encapsulation failure for each remote command that gets sent at bind time.

```
LEX-RCMD: encapsulation failure
```

The following output shows that the timer expired for the given remote command without having received a response from the FLEX device. This message can occur for any of the LAN Extender remote commands.

```
LEX-RCMD: timeout for Lex0: "lex priority-group" command
```

This message could be displayed for any of the following reasons:

- FLEX too busy to respond
- Transmission failure
- Software error

The following output indicates that the host is retransmitting the remote command after a timeout:

```
LEX-RCMD: re-transmitting Lex0: "lex priority-group" command
```

The following output indicates that an illegal parameter was passed to the `lex_setup_and_send` routine. This message could be displayed for due to a host software error.

```
LEX-RCMD: lex_setup_and_send called with invalid parameter
```

The following output is informational and shows when a bind occurs on a shutdown interface:

```
LEX-RCMD: bind occurred on shutdown LEX interface
```

The following output shows that LEX-NCP reached the open state and a bind operation was attempted with the FLEX's MAC address, but no free LAN Extender interfaces were found that were configured with that MAC address. This output can occur when the network administrator does not configure a LAN Extender interface with the correct MAC address.

```
LEX-RCMD: Serial0- No free Lex interface found with negotiated MAC address 0000.0c00.d8db
```

The following output shows that the serial line that was bound to the LAN Extender interface went down and the unbind routine was called, but when the list of active LAN Extender interfaces was searched, the LAN Extender interface corresponding to the serial interface was not found. This output usually occurs because of a host software error.

```
LEX-RCMD: No active Lex interface found for unbind
```

## debug list

Use the **debug list** EXEC command to filter debugging information on a per-interface or per-access list basis. The **no** form of this command turns off the list filter.

```
debug list [list] [interface]  
no debug list
```

### Syntax Description

<i>list</i>	(Optional) An access list number in the range of 1100–1199.
<i>interface</i>	(Optional) Interface type. Allowed values include <ul style="list-style-type: none"><li>• <b>channel</b>—IBM Channel interface</li><li>• <b>ethernet</b>—IEEE 802.3</li><li>• <b>fdi</b>—ANSI X3T9.5</li><li>• <b>null</b>—Null interface</li><li>• <b>serial</b>—Serial</li><li>• <b>tokenring</b>—IEEE 802.5</li><li>• <b>tunnel</b>—Tunnel interface</li></ul>

### Usage Guidelines

The **debug list** command is used with other debug commands for specific protocols and interfaces to filter the amount of debug information that is displayed. In particular, this command is designed to filter specific physical unit (PU) output from bridging protocols. The **debug list** command is supported with the following commands:

- **debug llc2 errors**
- **debug llc2 packets**
- **debug llc2 state**
- **debug rif**
- **debug sdlc**
- **debug token ring**

---

**Note** All **debug** commands that support access-list filtering use access lists in the range 1100-1199. The access list numbers shown in the examples are merely samples of valid numbers.

---

## Examples

To use **debug list** on only the first of several LLC2 connections, use the **show llc2** command to display the active connections:

```
Router# show llc2

Sdl1cVirtualRing2008 DTE: 4000.2222.22c7 4000.1111.111c 04 04 state NORMAL
Sdl1cVirtualRing2008 DTE: 4000.2222.22c8 4000.1111.1120 04 04 state NORMAL
Sdl1cVirtualRing2008 DTE: 4000.2222.22c1 4000.1111.1104 04 04 state NORMAL
```

Next, configure an extended bridging access list, numbered 1103, for the connection you want to filter:

```
access-list 1103 permit 4000.1111.111c 0000.0000.0000 4000.2222.22c7 0000.0000.0000 0xc
2 eq 0x404
```

The convention for LLC **debug list** filtering is to use dmac=6 bytes, smac=6 bytes, dsap\_offset = 12, and ssap\_offset = 13.

Finally, you invoke the following debug commands:

```
Router# debug list 1103
Router# debug llc2 packet

LLC2 Packets debugging is on
for access list: 1103
```

To use **debug list** for SDLC connections, with the exception of address 04, create access list 1102 to deny the specific address and permit all others:

```
access-list 1102 deny 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000 0xc 1
eq 0x4
access-list 1102 permit 0000.0000.0000 0000.0000.0000 0000.0000.0000 0000.0000.0000
```

The convention is to use dmac = 0.0.0, smac = 0.0.0, and sdlc\_frame\_offset = 12.

Invoke the following debug commands:

```
Router# debug list 1102
Router# debug sdlc

SDLC link debugging is on
for access list: 1102
```

To enable SDLC debugging (or debugging for any of the other supported protocols) for a specific interface rather than for all interfaces on a router, use the following commands:

```
Router# debug list serial 0
Router# debug sdlc

SDLC link debugging is on
for interface: Serial0
```

To enable Token Ring debugging between two MAC address, 0000.3018.4acd and 0000.30e0.8250, configure an extended bridging access list 1106:

```
access-list 1106 permit 0000.3018.4acd 8000.0000.0000 0000.30e0.8250 8000.0000.0000
access-list 1106 permit 0000.30e0.8250 8000.0000.0000 0000.3018.4acd 8000.0000.0000
```

Invoke the following debug commands:

```
Router# debug list 1106
Router# debug token ring

Token Ring Interface debugging is on
for access list: 1106
```

To enable RIF debugging for a single MAC address, configure an access list 1109:

```
access-list 1109 permit permit 0000.0000.0000 ffff.ffff.ffff 4000.2222.22c6  
0000.0000.0000
```

Invoke the following debug commands:

```
Router# debug list 1109  
Router# debug rif  
RIF update debugging is on  
for access list: 1109
```

### Related Commands

- debug llc2 errors**
- debug llc2 packet**
- debug llc2 state**
- debug rif**
- debug sdlc**
- debug token ring**

## debug llc2 dynwind

Use the **debug llc2 dynwind** EXEC command to display changes to the dynamic window over Frame Relay. The **no** form of this command disables debugging output.

**[no] debug llc2 dynwind**

### Sample Display

Figure 2-173 shows sample **debug llc2 dynwind** output.

**Figure 2-173 Sample Debug LLC2 Dynwind Output**

```
Router# debug llc2 dynwind

LLC2/DW: BECN received! event REC_I_CMD, Window size reduced to 4
LLC2/DW: 1 consecutive I-frame(s) received without BECN
LLC2/DW: 2 consecutive I-frame(s) received without BECN
LLC2/DW: 3 consecutive I-frame(s) received without BECN
LLC2/DW: 4 consecutive I-frame(s) received without BECN
LLC2/DW: 5 consecutive I-frame(s) received without BECN
LLC2/DW: Current working window size is 5
```

In this example, the router receives a backward explicit congestion notification (BECN) and reduces the window size to four. After receiving five consecutive I-frames without a BECN, the router increases the window size to five.

### Related Commands

**debug llc2 errors**  
**debug llc2 packet**  
**debug llc2 state**

## debug llc2 errors

Use the **debug llc2 errors** EXEC command to display Logical Link Control, type 2 (LLC2) protocol error conditions or unexpected input. The **no** form of this command disables debugging output.

**[no] debug llc2 errors**

### Sample Display

Figure 2-174 shows sample **debug llc2 errors** output from a router ignoring an incorrectly configured device.

**Figure 2-174 Sample Debug LLC2 Errors Output**

```
Router# debug llc2 errors

LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
LLC: admstate: 4000.1014.0001 0000.0000.0000 04 04 REC_RR_RSP
```

Each line of output contains the remote MAC address, the local MAC address, the remote service access point (SAP), and the local SAP. In this example, the router receives unsolicited RR frames marked as responses.

### Related Commands

**debug list**  
**debug llc2 dynwind**  
**debug llc2 packet**  
**debug llc2 state**

## debug llc2 packet

Use the **debug llc2 packet** EXEC command to display all input and output from the Logical Link Control, type 2 (LLC2) protocol stack. This command also displays information about some error conditions as well as internal interactions between the Common Link Services (CLS) layer and the LLC2 layer. The **no** form of this command disables debugging output.

**[no] debug llc2 packet**

### Sample Display

Figure 2-175 shows sample **debug llc2 packet** output from the router sending ping data back and forth to another router.

**Figure 2-175 Sample Debug LLC2 Packet Output**

```
Router# debug llc2 packet

LLC: llc2_input
401E54F0:                10400000                .@..
401E5500: 303A90CF 0006F4E1 2A200404 012B5E  0:..O..ta* ...+
LLC: i REC_RR_CMD N(R)=21 p/f=1
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:                0040 0006F4E1 2A200000                .@..ta* ..
401E5620: 303A90CF 04050129 00                N 0:..O...). 2012
LLC: llc_sendframe
4022E3A0:                0040 0006F4E1                .@..ta
4022E3B0: 2A200000 303A90CF 04042A28 2C000202  * ..0:..O...*(,....
4022E3C0: 00050B90 A02E0502 FF0003D1 004006C1  .... ..Q.@.A
4022E3D0: D7C9D5C  0.128
   C400130A C1D7D7D5 4BD5F2F0 WIUGD...AWWUKUrp
4022E3E0: F1F30000 011A6071 00010860 D7027000  qs....`q...`W.p.
4022E3F0: 00003B00 1112FF01 03000243 6973636F  ..;.....Cisco
4022E400: 20494F53 69                IOSi
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt I N(S)=21 N(R)=20 p/f=0 size=90
LLC: llc2_input
401E5620:                10400000 303A90CF                .@...:O
401E5630: 0006F4E1 2A200404 282C2C00 02020004  ..ta* ..(,.....
401E5640: 03902000 1112FF01 03000243 6973636F  .. ..Cisco
401E5650: 20494F53 A0                IOS
LLC: i REC_I_CMD N(R)=22 N(S)=20 V(R)=20 p/f=0
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_I_CMD (1)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(S)=20 V(R)=20
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_I_CMD N(R)=44
LLC: INFO: 0006.f4e1.2a20 0000.303a.90cf 04 04 v(r) 20
```

Explanations of representative lines in Figure 2-175 follow.

The first three lines indicate that the router has received some input from the link.

```
LLC: llc2_input
401E54F0:                10400000                .@..
401E5500: 303A90CF 0006F4E1 2A200404 012B5E  0:..O..ta* ...+
```

The next line indicates that this input was an RR command with the poll bit set. The other router has received sequence number 21 and is waiting for the final bit.

```
LLC: i REC_RR_CMD N(R)=21 p/f=1
```

The next two lines contain the MAC addresses of the sender and receiver as well as the state of the router when it received this frame.

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 NORMAL REC_RR_CMD (3)
LLC (rs): 0006.f4e1.2a20 0000.303a.90cf 04 04 REC_RR_CMD N(R)=42
```

The next four lines indicate that the router is transmitting a response with the final bit set.

```
LLC: 0006.f4e1.2a20 0000.303a.90cf 04 04 txmt RR_RSP N(R)=20 p/f=1
LLC: llc_sendframe
401E5610:          0040 0006F4E1 2A200000          .@..ta* ..
401E5620: 303A90CF 04050129 00          N 0:..O...). 2012
```

### Related Commands

- debug list**
- debug llc2 dynwind**
- debug llc2 errors**
- debug llc2 state**

## debug llc2 state

Use the **debug llc2 state** EXEC command to display state transitions of the Logical Link Control, type 2 (LLC2) protocol. The **no** form of this command disables debugging output.

**[no] debug llc2 state**

### Usage Guidelines

Refer to the ISO/IEC standard 8802-2 for definitions and explanations of **debug llc2 state** output.

### Sample Display

Figure 2-176 shows sample **debug llc2 state** output when a router disables and enables an interface.

**Figure 2-176 Sample Debug LLC2 State Output**

```
Router# debug llc2 state

LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> AWAIT (P_TIMER_EXP)
LLC(rs): 0006.f4e1.2a20 0000.303a.90cf 04 04, AWAIT -> D_CONN (P_TIMER_EXP)
LLC: cleanup 0006.f4e1.2a20 0000.303a.90cf 04 04, UNKNOWN (17)
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, ADM -> SETUP (CONN_REQ)
LLC: normalstate: set_local_busy 0006.f4e1.2a20 0000.303a.90cf 04 04
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, NORMAL -> BUSY (SET_LOCAL_BUSY)
LLC: Connection established: 0006.f4e1.2a20 0000.303a.90cf 04 04, success
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, SETUP -> BUSY (SET_LOCAL_BUSY)
LLC: busystate: 0006.f4e1.2a20 0000.303a.90cf 04 04 local busy cleared
LLC (stsw): 0006.f4e1.2a20 0000.303a.90cf 04 04, BUSY -> NORMAL (CLEAR_LOCAL_BUSY)
```

### Related Commands

**debug list**  
**debug llc2 dynwind**  
**debug llc2 errors**  
**debug llc2 packet**

## debug lnm events

Use the **debug lnm events** EXEC command to display any unusual events that occur on a Token Ring network. These events include stations reporting errors or error thresholds being exceeded. The **no** form of this command disables debugging output.

**[no] debug lnm events**

### Sample Display

Figure 2-177 shows sample **debug lnm events** output.

**Figure 2-177 Sample Debug LNM Events Output**

```
Router# debug lnm events

IBMNM3: Adding 0000.3001.1166 to error list
IBMNM3: Station 0000.3001.1166 going into preweight condition
IBMNM3: Station 0000.3001.1166 going into weight condition
IBMNM3: Removing 0000.3001.1166 from error list
LANMGR0: Beaconsing is present on the ring
LANMGR0: Ring is no longer beaconsing
IBMNM3: Beaconsing, Postmortem Started
IBMNM3: Beaconsing, heard from 0000.3000.1234
IBMNM3: Beaconsing, Postmortem Next Stage
IBMNM3: Beaconsing, Postmortem Finished
```

Explanations for the messages shown in Figure 2-177 follow.

The following message indicates that station 0000.3001.1166 reported errors and has been added to the list of stations reporting errors. This station is located on Ring 3.

```
IBMNM3: Adding 0000.3001.1166 to error list
```

The following message indicates that station 0000.3001.1166 has passed the “early warning” threshold for error counts:

```
IBMNM3: Station 0000.3001.1166 going into preweight condition
```

The following message indicates that station 0000.3001.1166 is experiencing a severe number of errors:

```
IBMNM3: Station 0000.3001.1166 going into weight condition
```

The following message indicates that the error counts for station 0000.3001.1166 have all decayed to zero, so this station is being removed from the list of stations that have reported errors:

```
IBMNM3: Removing 0000.3001.1166 from error list
```

The following message indicates that Ring 0 has entered failure mode. This ring number is assigned internally.

```
LANMGR0: Beaconsing is present on the ring
```

The following message indicates that Ring 0 is no longer in failure mode. This ring number is assigned internally.

```
LANMGR0: Ring is no longer beaconsing
```

The following message indicates that the router is beginning its attempt to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. The router attempts to contact stations that were part of the fault domain to detect whether they are still operating on the ring.

```
IBMNM3: Beaconing, Postmortem Started
```

The following message indicates that the router is attempting to determine whether or not any stations left the ring during the automatic recovery process for the last beaconing failure. It received a response from station 0000.3000.1234, one of the two stations in the fault domain.

```
IBMNM3: Beaconing, heard from 0000.3000.1234
```

The following message indicates that the router is attempting to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It is initiating another attempt to contact the two stations in the fault domain.

```
IBMNM3: Beaconing, Postmortem Next Stage
```

The following message indicates that the router has attempted to determine whether any stations left the ring during the automatic recovery process for the last beaconing failure. It has successfully heard back from both stations that were part of the fault domain.

```
IBMNM3: Beaconing, Postmortem Finished
```

Explanations follow for other messages that the **debug lnm events** command can generate.

The following message indicates that the router is out of memory:

```
LANMGR: memory request failed, find_or_build_station()
```

The following message indicates that Ring 3 is experiencing a large number of errors that cannot be attributed to any individual station:

```
IBMNM3: Non-isolating error threshold exceeded
```

The following message indicates that a station (or stations) on Ring 3 are receiving frames faster than they can be processed:

```
IBMNM3: Adapters experiencing congestion
```

The following message indicates that the beaconing has lasted for over 1 minute and is considered a “permanent” error:

```
IBMNM3: Beaconing, permanent
```

The following message indicates that the beaconing lasted for less than 1 minute. The router is attempting to determine whether either station in the fault domain left the ring.

```
IBMNM: Beaconing, Destination Started
```

In the preceding line of output, the following can replace “Started”: “Next State”, “Finished”, “Timed out”, and “Cannot find station *n*”.

## debug lnm llc

Use the **debug lnm llc** EXEC command to display all communication between the router/bridge and the LAN Network Managers (LNMs) that have connections to it. The **no** form of this command disables debugging output.

**[no] debug lnm llc**

### Usage Guidelines

One line is displayed for each message sent or received.

### Sample Display

Figure 2-178 shows sample **debug lnm llc** output.

**Figure 2-178 Sample Debug LNM LLC Output**

```
Router# debug lnm llc

IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
IBMNM: Sending LRM LAN Manager Accepted to 1000.5ade.0d8a on link 0.
IBMNM: sending LRM New Reporting Link Established to 1000.5a79.dbf8 on link 1.
IBMNM: Determining new controlling LNM
IBMNM: Sending Report LAN Manager Control Shift to 1000.5ade.0d8a on link 0.
IBMNM: Sending Report LAN Manager Control Shift to 1000.5a79.dbf8 on link 1.

IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
IBMNM: Sending Report Bridge Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Request REM Status from 1000.5ade.0d8a.
IBMNM: Sending Report REM Status to 1000.5ade.0d8a on link 0.
IBMNM: Bridge 001-2-00A received Set Bridge Parameters from 1000.5ade.0d8a.
IBMNM: Sending Bridge Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending Bridge Params Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Bridge 001-2-00A received Set REM Parameters from 1000.5ade.0d8a.
IBMNM: Sending REM Parameters Set to 1000.5ade.0d8a on link 0.
IBMNM: sending REM Parameters Changed Notification to 1000.5a79.dbf8 on link 1.
IBMNM: Received LRM Set Reporting Point frame from 1000.5ade.0d8a.
IBMNM: found bridge: 001-1-00A, addresses: 0000.3080.2d79 4000.3080.2d7
```

As Figure 2-178 indicates, **debug lnm llc** output can vary somewhat in format. Table 2-86 describes significant fields shown in the first line of output in Figure 2-178.

**Table 2-86 Debug LNM LLC Field Descriptions**

Field	Description
IBMNM:	This line of output displays LLC-level debugging information.
Received	The router received a frame. The other possible value is Sending, to indicate that the router is sending a frame.

**Table 2-86 Debug LNM LLC Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
LRM	The function of the LLC-level software that is communicating: <ul style="list-style-type: none"> <li>• CRS—Configuration Report Server</li> <li>• LBS—LAN Bridge Server</li> <li>• LRM—LAN Reporting Manager</li> <li>• REM—Ring Error Monitor</li> <li>• RPS—Ring Parameter Server</li> <li>• RS—Ring Station</li> </ul>
Set Reporting Point	Name of the specific frame that the router sent or received. Possible values include the following: <ul style="list-style-type: none"> <li>• Bridge Counter Report</li> <li>• Bridge Parameters Changed Notification</li> <li>• Bridge Parameters Set</li> <li>• CRS Remove Ring Station</li> <li>• CRS Report NAUN Change</li> <li>• CRS Report Station Information</li> <li>• CRS Request Station Information</li> <li>• CRS Ring Station Removed</li> <li>• LRM LAN Manager Accepted</li> <li>• LRM Set Reporting Point</li> <li>• New Reporting Link Established</li> <li>• REM Forward MAC Frame</li> <li>• REM Parameters Changed Notification</li> <li>• REM Parameters Set</li> <li>• Report Bridge Status</li> <li>• Report LAN Manager Control Shift</li> <li>• Report REM Status</li> <li>• Request Bridge Status</li> <li>• Request REM Status</li> <li>• Set Bridge Parameters</li> <li>• Set REM Parameters</li> </ul>
from 1000.5ade.0d8a	If the router has received the frame, this address is the source address of the frame. If the router is sending the frame, this address is the destination address of the frame.

Explanations for other types of messages shown in Figure 2-178 follow.

The following message indicates that the lookup for the bridge with which the LAN Manager was requesting to communicate was successful:

```
IBMNM: found bridge: 001-2-00A, addresses: 0000.3040.a630 4000.3040.a630
```

The following message is self-explanatory:

```
IBMNM: Opening connection to 1000.5ade.0d8a on TokenRing0
```

The following message indicates that a LAN Manager has connected or disconnected from an internal bridge and that the router computes which LAN Manager is allowed to change parameters:

```
IBMNM: Determining new controlling LNM
```

The following line of output indicates which bridge in the router is the destination for the frame:

```
IBMNM: Bridge 001-2-00A received Request Bridge Status from 1000.5ade.0d8a.
```

## debug lnm mac

Use the **debug lnm mac** EXEC command to display all management communication between the router/bridge and all stations on the local Token Rings. The **no** form of this command disables debugging output.

**[no] debug lnm mac**

### Usage Guidelines

One line is displayed for each message sent or received.

### Sample Display

Figure 2-179 shows sample **debug lnm mac** output.

**Figure 2-179 Sample Debug LNM MAC Output**

```
Router# debug lnm mac

LANMGR0: RS received request address from 4000.3040.a670.
LANMGR0: RS sending report address to 4000.3040.a670.
LANMGR0: RS received request state from 4000.3040.a670.
LANMGR0: RS sending report state to 4000.3040.a670.
LANMGR0: RS received request attachments from 4000.3040.a670.
LANMGR0: RS sending report attachments to 4000.3040.a670.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: CRS received report NAUN change from 0000.3040.a630.
LANMGR2: RS start watching ring poll.
LANMGR0: CRS received report NAUN change from 0000.3040.a630.
LANMGR0: RS start watching ring poll.
LANMGR2: REM received report soft error from 0000.3040.a630.
LANMGR0: REM received report soft error from 0000.3040.a630.
LANMGR2: RS received ring purge from 0000.3040.a630.
LANMGR2: RS received AMP from 0000.3040.a630.
LANMGR2: RS received SMP from 0000.3080.2d79.
LANMGR2: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR2: RS start watching ring poll.
LANMGR0: RS received ring purge from 0000.3040.a630.
LANMGR0: RS received AMP from 0000.3040.a630.
LANMGR0: RS received SMP from 0000.3080.2d79.
LANMGR0: CRS received report NAUN change from 1000.5ade.0d8a.
LANMGR0: RS start watching ring poll.
LANMGR2: RS received SMP from 1000.5ade.0d8a.
LANMGR2: RPS received request initialization from 1000.5ade.0d8a.
LANMGR2: RPS sending initialize station to 1000.5ade.0d8a.
```

Table 2-87 describes significant fields shown in the first line of output in Figure 2-179.

**Table 2-87      Debug LNM MAC Field Descriptions**

Field	Description
LANMGR0:	LANMGR indicates that this line of output displays MAC-level debugging information. 0 indicates the number of the Token Ring interface associated with this line of debugging output.
RS	Indicates which function of the MAC-level software is communicating: <ul style="list-style-type: none"> <li>• CRS—Configuration Report Server</li> <li>• REM—Ring Error Monitor</li> <li>• RPS—Ring Parameter Server</li> <li>• RS—Ring Station</li> </ul>
received	Indicates that the router received a frame. The other possible value is “sending”, to indicate that the router is sending a frame.
request address	Indicates the name of the specific frame that the router sent or received. Possible values include the following: <ul style="list-style-type: none"> <li>• AMP</li> <li>• initialize station</li> <li>• report address</li> <li>• report attachments</li> <li>• report nearest active upstream neighbor (NAUN) change</li> <li>• report soft error</li> <li>• report state</li> <li>• request address</li> <li>• request attachments</li> <li>• request initialization</li> <li>• request state</li> <li>• ring purge</li> <li>• SMP</li> </ul>
from 4000.3040.a670	Indicates the source address of the frame, if the router has received the frame. If the router is sending the frame, this address is the destination address of the frame.

As Figure 2-179 indicates, all **debug lnm mac** messages follow the format described in Table 2-87 except the following:

```
LANMGR2: RS start watching ring poll
LANMGR2: RS stop watching ring poll
```

These messages indicate that the router starts and stops receiving AMP and SMP frames. These frames are used to build a current picture of which stations are on the ring.

## debug local-ack state

Use the **debug local-ack state EXEC** command to display the new and the old state conditions whenever there is a state change in the local acknowledgment state machine. The **no** form of this command disables debugging output.

**[no] debug local-ack state**

### Sample Display

Figure 2-180 shows sample **debug local-ack state** output.

**Figure 2-180 Sample Debug Local-Ack State Output**

```
Router# debug local-ack state

LACK_STATE: 2370300, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2370304, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2373816, hashp 2AE628, old state = connected, new state = disconnected
LACK_STATE: 2489548, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2489548, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
LACK_STATE: 2490132, hashp 2AE628, old state = connected, new state = awaiting
linkdown response
LACK_STATE: 2490140, hashp 2AE628, old state = awaiting linkdown response,
new state = disconnected
LACK_STATE: 2497640, hashp 2AE628, old state = disconn, new state = awaiting
LLC2 open to finish
LACK_STATE: 2497644, hashp 2AE628, old state = awaiting LLC2 open to finish,
new state = connected
```

Table 2-88 describes significant fields shown in Figure 2-180.

**Table 2-88 Debug Local-Ack State Field Descriptions**

Field	Description
LACK_STATE:	Indication that this packet describes a state change in the local acknowledgment state machine.
2370300	System clock.
hashp 2AE628	Internal control block pointer used by technical support staff for debugging purposes.
old state = disconn	The old state condition in the local acknowledgment state machine. Possible values include the following: <ul style="list-style-type: none"> <li>• Disconn (disconnected)</li> <li>• awaiting LLC2 open to finish</li> <li>• connected</li> <li>• awaiting linkdown response</li> </ul>

**Table 2-88      Debug Local-Ack State Field Descriptions (Continued)**

<b>Field</b>	<b>Description</b>
new state = awaiting LLC2 open to finish	The new state condition in the local acknowledgment state machine. Possible values include the following: <ul style="list-style-type: none"><li>• Disconn (disconnected)</li><li>• awaiting LLC2 open to finish</li><li>• connected</li><li>• awaiting linkdown response</li></ul>