

debug ipx ipxwan

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

debug ipx ipxwan

no debug ipx ipxwan

Syntax Description

This command has no arguments or keywords.

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.

Examples

The following is sample output from the **debug ipx ipxwan** command during link startup:

```
Router# debug ipx ipxwan
```

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to up
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
state brought up)]
IPXWAN: state (Sending Timer Requests -> Disconnect) [Serial1/6666:200 (IPX line
state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
state brought up)]

IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 1] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 2] out Serial1/6666:200
IPXWAN: Send TIMER_REQ [seq 0] out Serial1/6666:200

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)]
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)]
```

The following line indicates that the interface has initialized:

```
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, 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) [Serial1/6666:200 (IPX line
state brought down)]
IPXWAN: state (Disconnect -> Sending Timer Requests) [Serial1/6666:200 (IPX line
state brought up)]
```

The following lines indicate that the interface is sending timer requests and waiting for a 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

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

debug ipx nasi {**packets** | **error** | **activity**}

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

Syntax Description	
packets	Displays normal operating messages relating to incoming and outgoing NASI packets. This is the default.
error	Displays messages indicating an error or failure in the protocol processing.
activity	Displays messages relating to internal NASI processing of NASI connections. The activity 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.

Examples

The following is sample output from the **debug ipx nasi** command using the **packet** and **error** keywords:

```
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
```

In the following line, the 0 is the number of the 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 that 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 devices. 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 Commands

Command	Description
debug ipx spx	Displays debugging messages related to the SPX protocol.

debug ipx packet

To display information about packets received, sent, and forwarded, use the **debug ipx packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx packet

no debug ipx packet

Syntax Description

This command has no arguments or keywords.

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.

Examples

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

```
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
```

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 105](#) describes the significant fields shown in the display.

Table 105 *debug ipx packet* Field Descriptions

Field	Description
IPX	Indicates 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.

Table 105 *debug ipx packet Field Descriptions (continued)*

Field	Description
packet received	Router received this packet from a Novell station, possibly through an intermediate router.
gw=183.0000.0c01.5d85	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	Router is attempting to send this packet.

debug ipx routing

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

debug ipx routing {activity | events}

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.

Examples

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

```
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 106](#) describes the significant fields in the display.

Table 106 debug ipx routing Field Descriptions

Field	Description
IPXRIP	IPX RIP packet.
update from 9999.0260.8c6a.1733	Routing update packet 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	Router is sending this IPX routing update packet to address 12FF02:ffff.ffff.ffff through Ethernet interface 1.

Table 106 *debug ipx routing Field Descriptions (continued)*

Field	Description
network 555	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 Commands

Command	Description
debug ipx sap	Displays information about IPX SAP packets.

debug ipx sap

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

debug ipx sap [activity | events]

no debug ipx sap [activity | events]

Syntax Description	activity	(Optional) Provides more detailed output of SAP packets, including displays of services in SAP packets.
	events	(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 substantial amount of output, use it with caution on networks that have many interfaces and large service tables.

Examples

The following is sample output from the **debug ipx sap** command:

```
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
```

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 107 describes the significant fields shown in the display.

Table 107 *debug ipx sap Field Descriptions*

Field	Description
I	Indicates 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 <i>n</i> include: <ul style="list-style-type: none"> 1—General query 2—General response 3—Get Nearest Server request 4—Get Nearest Server response
len 160	Length of this packet (in bytes).
src: 160.000.0c00.070d	Source address of the packet.
dest:160.ffff.ffff.ffff	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 108 describes the significant fields shown in the display.

Table 108 *debug ipx sap Field Descriptions*

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

Table 108 *debug ipx sap Field Descriptions (continued)*

Field	Description
"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
```

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 109](#) describes possible values for the ssoc: field.

Table 109 *debug ipx sap Field Descriptions*

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

Related Commands

Command	Description
debug ipx routing	Displays information on IPX routing packets that the router sends and receives.

debug ipx spoof

To display information about SPX keepalive and IPX watchdog packets when **ipx watchdog** and **ipx spx-spoof** are configured on the router, use the **debug ipx spoof** privileged EXEC command. The **no** form of this command disables debugging output.

debug ipx spoof

no debug ipx spoof

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

Use this command to troubleshoot connections that use SPX spoofing when SPX keepalive spoofing is enabled.

Examples

The following is sample output from the **debug ipx spoof** command:

Router# **debug ipx spoof**

```
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
```

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 for connections that exist in the SPX table but that 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

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

debug ipx spx

no debug ipx spx

Syntax Description

This command has no arguments or keywords.

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.

Examples

The following is sample output from the **debug ipx spx** command:

```
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
```

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
```

 debug ipx spx**Related Commands**

Command	Description
debug ipx nasi	Displays information about the NASI connections.

debug isdn event

To display ISDN events occurring on the user side (on the router) of the ISDN interface, use the **debug isdn event** privileged EXEC command. The **no** form of this command disables debugging output.

debug isdn event

no debug isdn event

Syntax Description

This command has no arguments or keywords.

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.

The ISDN events that can be displayed are Q.931 events (call setup and teardown of ISDN network connections).

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, see Appendix B, “ISDN Switch Types, Codes, and Values.”

Examples

The following is sample output from the **debug isdn event** command of call setup events for an 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
```

The following 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. See Appendix B, “ISDN Switch Types, Codes, and Values,” for information about these values.

```
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

```

The following is sample output from the **debug isdn event** command of call teardown events for a call that has been disconnected by the host side of the connection:

```

Router# debug isdn event

received HOST_DISCONNECT
ISDN Event: Call to 415555121202 was hung up

```

The following is sample output from the **debug isdn event** command of a call teardown event for an outgoing or incoming call that has been disconnected by the ISDN interface on the router side:

```

Router# debug isdn event

ISDN Event: Hangup call to call id 0x8008

```

[Table 110](#) describes the significant fields shown in the display.

Table 110 *debug isdn event Field Descriptions*

Field	Description
Bearer Capability	Indicates the requested bearer service to be provided by the network. See 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 Q.931 specification for details about the possible values associated with each field for which this identifier is relevant.
Channel ID	<p>Channel Identifier. The values and corresponding channels might be identified in several ways:</p> <ul style="list-style-type: none"> Channel ID i=0x0101—Channel B1 Channel ID i=0x0102—Channel B2 <p>ITU-T Q.931 defines the values and channels as exclusive or preferred:</p> <ul style="list-style-type: none"> 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 110 *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 64 Kb/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.

The following is sample output from the **debug isdn event** command of a call teardown event for a call that has passed call screening and then has been hung up by the ISDN interface on the far end side:

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
```

The following is sample output from the **debug isdn event** command 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:

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
```

The following is sample output from the **debug isdn event** command of a call teardown event for an outgoing call that uses a dialer 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
Jan  3 11:41:48.743:          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 is similar to the output of **debug isdn q931**. Refer to the **debug isdn q931** command for detailed field descriptions.

The following is sample output from the **debug isdn event** command of call setup events for a successful callback for 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
```

The following is sample output from the **debug isdn event** command for a callback that was unsuccessful because the router had no dialer map for the calling number:

```
Router# debug isdn event
```

```
BRI0:Caller id 81012345678902 callback - no matching map
```

[Table 111](#) describes the significant fields shown in the display.

Table 111 *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.

The following is sample output from the **debug isdn event** command for a callback that was successful when the dialer profiles DDR feature is configured:

```
*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
*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 112 describes significant fields of call setup events for a successful callback for the sample output from the **debug isdn event** command when the dialer profiles DDR feature is configured.

Table 112 *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 are 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

To display data link layer (Layer 2) access procedures that are taking place at the router on the D channel (LAPD) of its ISDN interface, use the **debug isdn q921** privileged EXEC command. The **no** form of this command disables debugging output.

debug isdn q921

no debug isdn q921

Syntax Description

This command has no arguments or keywords.

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 sent over the B channels that is 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, see Appendix B, "ISDN Switch Types, Codes, and Values."

Examples

The following is sample output from the **debug isdn q921** command for an 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, 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

```

The following is sample output from the **debug isdn q921** command for a 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 a Layer 2 link establishment.

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 need 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).

The following is sample output from the **debug isdn q921** command for an incoming call. It is an incoming SETUP message that assumes that the Layer 2 link is already established to the other side.

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 113 describes the significant fields shown in the display.

Table 113 *debug isdn q921 Field Descriptions*

Field	Description
Jan 3 14:49:47.391	Indicates the date and time at which the frame was sent 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 sent 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 (ASP) 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) 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 from 0 to 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 from 64 to 126.

Table 113 *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 debug isdn q921 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 from 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 from 0 to 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 data-link 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 113 *debug isdn q921 Field Descriptions (continued)*

Field	Description
SPID information i = 0x343135393033383336363031	Indicates the 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 sent 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 re-sent 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 from 64 to 126. The value 127 indicates a broadcast.

debug isdn q931

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

debug isdn q931

no debug isdn q931

Syntax Description

This command has no arguments or keywords.

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 sent 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."

Examples

The following is sample output from the **debug isdn q931** command of a 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)
```

The following is sample output from the **debug isdn q931** command of a 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

```

The following is sample output from the **debug isdn q931** command of a 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

```

The following is sample output from the **debug isdn q931** command of a 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 114](#) describes the significant fields shown in the display.

Table 114 *debug isdn q931 Command Call Setup Procedure Field Descriptions*

Field	Description
TX ->	Indicates that this message is being sent 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.
callref	Indicates the call reference number in hexadecimal notation. 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.

Table 114 *debug isdn q931 Command Call Setup Procedure Field Descriptions (continued)*

Field	Description
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

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

debug isis adj packets

no debug isis adj packets

Syntax Description

This command has no arguments or keywords.

Examples

The following is sample output from the **debug isis adj packets** command:

```
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
```

The following line indicates that the router received an IS-IS hello packet (IIH) on Ethernet interface 0 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 Ethernet interface 1 is 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 Ethernet interface 1 is 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 mpls traffic-eng advertisements

To print information about traffic engineering advertisements in ISIS Link-state advertisement (LSA) messages, use the **debug isis mpls traffic-eng advertisements** EXEC command. To disable debugging output, use the **no** form of this command.

debug isis mpls traffic-eng advertisements

[no debug isis mpls traffic-eng advertisements]

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

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

Examples

In the following example, information about traffic engineering advertisements is printed in ISIS LSA messages:

```
debug isis mpls traffic-eng advertisements
```

```
System ID:Router1.00
Router ID:10.106.0.6
Link Count:1
Link[1]
  Neighbor System ID:Router2.00 (P2P link)
  Interface IP address:10.42.0.6
  Neighbor IP Address:10.42.0.10
  Admin. Weight:10
  Physical BW:155520000 bits/sec
  Reservable BW:5000000 bits/sec
  BW unreserved[0]:2000000 bits/sec, BW unreserved[1]:100000 bits/sec
  BW unreserved[2]:100000 bits/sec, BW unreserved[3]:100000 bits/sec
  BW unreserved[4]:100000 bits/sec, BW unreserved[5]:100000 bits/sec
  BW unreserved[6]:100000 bits/sec, BW unreserved[7]:0 bits/sec
  Affinity Bits:0x00000000
```

Table 115 describes the significant fields shown in the display.

Table 115 *debug isis mpls traffic-eng advertisements Field Descriptions*

Field	Description
System ID	Identification value for the local system in the area.
Router ID	MPLS traffic engineering router ID.

Table 115 *debug isis mpls traffic-eng advertisements Field Descriptions (continued)*

Field	Description
Link Count	Number of links that MPLS traffic engineering advertised.
Neighbor System ID	Identification value for the remote system in an area.
Interface IP address	IPv4 address of the interface.
Neighbor IP Address	IPv4 address of the neighbor.
Admin. Weight	Administrative weight associated with this link.
Physical BW	Bandwidth capacity of the link (in bits per second).
Reservable BW	Amount of reservable bandwidth on this link.
BW unreserved	Amount of bandwidth that is available for reservation.
Affinity Bits	Attribute flags of the link that are being flooded.

debug isis mpls traffic-eng events

To print information about traffic engineering-related ISIS events, use the **debug isis mpls traffic-eng events** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug isis mpls traffic-eng events

no debug isis mpls traffic-eng events

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.0(5)ST	This command was introduced.

Examples In the following example, information is printed about traffic engineering-related ISIS events:

```
debug isis mpls traffic-eng events
```

```
ISIS-RRR:Send MPLS TE Et4/0/1 Router1.02 adjacency down:address 0.0.0.0
ISIS-RRR:Found interface address 10.1.0.6 Router1.02, building subtlv... 58 bytes
ISIS-RRR:Found interface address 10.42.0.6 Router2.00, building subtlv... 64 bytes
ISIS-RRR:Interface address 0.0.0.0 Router1.00 not found, not building subtlv
ISIS-RRR:LSP Router1.02 changed from 0x606BCD30
ISIS-RRR:Mark LSP Router1.02 changed because TLV contents different, code 16
ISIS-RRR:Received 1 MPLS TE links flood info for system id Router1.00
```

debug isis spf statistics

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

debug isis spf statistics

no debug isis spf statistics

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The Intermediate System-to-Intermediate System (IS-IS) Interdomain Routing 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 SPF routing algorithm to build Level 1 and Level 2 routes. The **debug isis spf statistics** command provides information for determining the time required 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.

Examples

The following is sample output from the **debug isis spf statistics** command:

```
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 116](#) describes the significant fields shown in the display.

Table 116 *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 required to place the ISs on the SPT.

Table 116 *debug isis spf statistics Field Descriptions (continued)*

Field	Description
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 the domain.
Complete L2 SPT	Indicates that the algorithm has completed for Level 2 routing.

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, 4 milliseconds were required 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 intradomain topology, 56 milliseconds were required to place 12 Level 2 ISs on the SPT.

debug isis update-packets

To display various sequence number protocol data units (PDUs) and link-state packets that are detected by a router, use the **debug isis update-packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug isis update-packets

no debug isis update-packets

Syntax Description

This command has no arguments or keywords.

Examples

This router has been configured for IS-IS routing. The following is sample output from the **debug isis update-packets** command:

```
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
```

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

```
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 sent:

```
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 tunnel 0 interface:

```
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

To display information associated with the Kerberos Authentication Subsystem, use the **debug kerberos** privileged EXEC command. The **no** form of this command disables debugging output.

debug kerberos

no debug kerberos

Syntax Description

This command has no arguments or keywords.

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.

Examples

The following is part of the sample output from the **debug aaa authentication** command for a Kerberos login attempt that failed. The information indicates that Kerberos is the authentication method used.

```
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
```

The following is sample output from the **debug kerberos** command for a login attempt that was successful. The information indicates that the router sent a request to the KDC and received a valid credential.

```
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
```

The following is sample output from the **debug kerberos** command 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.

```
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
```

The following output shows other failure messages you might see that indicate a configuration problem. The first message indicates that 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 that the router failed to retrieve its own IP address. The third message indicates that 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.

```
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 Commands

Command	Description
debug aaa authentication	Displays information on accountable events as they occur.

debug l2relay events

To start debugging of Layer 2 Relay events, use the **debug l2relay events** command. To disable debugging output, use the **no** form of the command (SGSN D-node only).

debug l2relay events

no debug l2relay events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

The SGSN module uses the proprietary Layer 2 Relay protocol in conjunction with the intra-Serving GPRS Support Node (iSGSN) protocol for communication between the SGSN-datacom (SGSN-D) and SGSN-telecom (SGSN-T) units that comprise the SGSN.

For debugging purposes, it might also be useful to trace Layer 2 Relay packets. To display information about Layer 2 Relay packets, use the **debug l2relay packets** command.

Normally you will not need to use the **debug l2relay events** or **debug l2relay packets** commands. If problems with the SGSN are encountered, Cisco technical support personnel may request that issue the command.



Caution

Because the **debug l2relay events** command generates a substantial amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of Layer 2 Relay events:

```
router# debug l2relay events
```

Related Commands

Command	Description
debug l2relay packets	Displays Layer 2 Relay packets (SGSN D-node only).

debug l2relay packets

To display information about Layer 2 Relay packets, use the **debug l2relay packets** command. To disable debugging output, use the **no** form of the command (SGSN D-node only).

debug l2relay packets

no debug l2relay packets

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(1)GA	This command was introduced.
12.1(3)T	This command was integrated into Cisco IOS Release 12.1(3)T.

Usage Guidelines

Use the **debug l2relay packets** command to display information about Layer 2 Relay packets.

The SGSN module uses the proprietary Layer 2 Relay protocol in conjunction with the intra-Serving GPRS Support Node (iSGSN) protocol for communication between the SGSN-datacom (SGSN-D) and SGSN-telecom (SGSN-T) units that comprise the SGSN.

For debugging purposes, it might also be useful to trace Layer 2 Relay events. To display information about Layer 2 Relay events, use the **debug l2relay events** command.

Normally you will not need to use the **debug l2relay packets** or **debug l2relay events** command. If problems with the SGSN are encountered, Cisco technical support personnel may request that you issue the command.



Caution

Because the **debug l2relay packets** command generates a significant amount of output, use it only when traffic on the GPRS network is low, so other activity on the system is not adversely affected.

Examples

The following example enables the display of Layer 2 Relay packets:

```
router# debug l2relay packets
```

Related Commands

Command	Description
debug ip igmp	Displays Layer 2 Relay events (SGSN D-node only).

debug lane client

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

debug lane client { **all** | **le-arp** | **mpoa** | **packet** | **signaling** | **state** | **topology** } [**interface** *interface*]

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

Syntax Description

all	Displays all debug information related to the LEC.
le-arp	Displays debug information related to the LANE ARP table.
mpoa	Displays debug information to track the following: <ul style="list-style-type: none"> • MPOA specific TLV information in le-arp requests/responses • Elan-id and local segment TLV in lane control frames • When a LANE client is bound to an MPC/MPS
packet	Displays debug information about each packet.
signaling	Displays debug information related to client SVCs.
state	Displays debug information when the state changes.
topology	Displays debug information related to the topology of the emulated LAN (ELAN).
interface <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.

Defaults

If the interface number is not specified, the default will be the number of all the **mpoa lane** clients.

Command History

Release	Modification
12.0(1)T	This command was introduced.

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.

Examples

Sample Displays

The following example shows output for **debug lane client packet** and **debug lane client state** commands for an LEC joining an ELAN named elan1:

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

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

The LEC listens for signalling 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 server (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 00.000000000000000000000000.000000000000.00

```

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 from the **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
```

The following example shows **debug lane client all** command output when an interface with LECS, an LES/BUS, and an LEC is shut down:

```
Router# debug lane client all
```

```
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2: callid      0x60E8B474
LEC ATM1/0.2: cause code 0
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: callid      0x60EB6160
LEC ATM1/0.2: cause code 31
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2: callid      0x60EB7548
LEC ATM1/0.2: cause code 31
LEC ATM1/0.2: sending RELEASE
LEC ATM1/0.2: callid      0x60EB9E48
LEC ATM1/0.2: cause code 31
LEC ATM1/0.2: sending CANCEL
LEC ATM1/0.2: ATM address 47.00918100000000613E5A2F01.006070174820.02
LEC ATM1/0.2: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.3: received RELEASE_COMPLETE
```

```

LEC ATM1/0.3: callid 0x60E8D108
LEC ATM1/0.3: cause code 0
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: callid 0x60EB66D4
LEC ATM1/0.3: cause code 31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3: callid 0x60EB7B8C
LEC ATM1/0.3: cause code 31
LEC ATM1/0.3: sending RELEASE
LEC ATM1/0.3: callid 0x60EBA3BC
LEC ATM1/0.3: cause code 31
LEC ATM1/0.3: sending CANCEL
LEC ATM1/0.3: ATM address 47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.2: received RELEASE_COMPLETE
LEC ATM1/0.2: callid 0x60EB7548
LEC ATM1/0.2: cause code 0
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: callid 0x60EB7B8C
LEC ATM1/0.3: cause code 0
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: callid 0x60EBC458
LEC ATM1/0.1: cause code 0
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: callid 0x60EBD30C
LEC ATM1/0.1: cause code 31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1: callid 0x60EBDD28
LEC ATM1/0.1: cause code 31
LEC ATM1/0.1: sending RELEASE
LEC ATM1/0.1: callid 0x60EBF174
LEC ATM1/0.1: cause code 31
LEC ATM1/0.1: sending CANCEL
LEC ATM1/0.1: ATM address 47.00918100000000613E5A2F01.006070174820.01
LEC ATM1/0.1: state ACTIVE event LEC_SIG_RELEASE_COMP => TERMINATING
LEC ATM1/0.1: received RELEASE_COMPLETE
LEC ATM1/0.1: callid 0x60EBDD28
LEC ATM1/0.1: cause code 0
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: callid 0x60EB6160
LEC ATM1/0.2: cause code 0
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: callid 0x60EB66D4
LEC ATM1/0.3: cause code 0
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: callid 0x60EB9E48
LEC ATM1/0.2: cause code 0
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:  callid      0x60EBA3BC
LEC ATM1/0.3:  cause code  0
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:  callid      0x60EBD30C
LEC ATM1/0.1:  cause code  0
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:  callid      0x60EBF174
LEC ATM1/0.1:  cause code  0
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 address  47.00918100000000613E5A2F01.006070174820.01
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 address  47.00918100000000613E5A2F01.006070174820.02
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 address  47.00918100000000613E5A2F01.006070174820.03
LEC ATM1/0.3:  state IDLE event LEC_LOCAL_DEACTIVATE => IDLE

```

The following output is from the **debug lane client mpoa** command when the **lane** interface is shut down:

```

Router# debug lane client mpoa

Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0.1
Router(config-subif)#shutdown
Router(config-subif)#
00:23:32:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:23:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:23:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#
Router(config-subif)#exit
Router(config)#exit

```

The following output is from the **debug lane client mpoa** command when the **lane** interface is started (not shut down):

```

Router# debug lane client mpoa

Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int atm 1/1/0.1
Router(config-subif)#
Router(config-subif)#

```



```

Router(config-subif)#no shutdown
Router(config-subif)#
00:23:39:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_CONFIG_RSP, num_tlvs 14
00:23:39:LEC ATM1/1/0.1:elan id from LECS set to 300
00:23:39:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_JOIN_RSP, num_tlvs 1
00:23:39:LEC ATM1/1/0.1:elan id from LES set to 300
00:23:39:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:23:39:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
00:23:39:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to up
00:23:39:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:UP
00:25:57:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_ARP_REQ, num_tlvs 1
00:25:57:LEC ATM1/1/0.1:lec_process_dev_type_tlv: lec 47.0091810000000050E
2097801.00500B306440.02
    type MPS, mpc 00.0000000000000000000000000000.000000000000.00
    mps 47.0091810000000050E2097801.00500B306444.00, num_mps_mac 1, mac 0050.0b3
0.6440
00:25:57:LEC ATM1/1/0.1:create mpoa_lec
00:25:57:LEC ATM1/1/0.1:new mpoa_lec 0x617E3118
00:25:57:LEC ATM1/1/0.1:lec_process_dev_type_tlv:type MPS, num _mps_mac
1
00:25:57:LEC ATM1/1/0.1:lec_add_mps:
    remote lec 47.0091810000000050E2097801.00500B306440.02
    mps 47.0091810000000050E2097801.00500B306444.00 num_mps_mac 1, mac 0050.0b30
.6440
00:25:57:LEC ATM1/1/0.1:mpoa_device_change:lec_nsap 47.0091810000000050E20978
01.00500B306440.02, appl_type 5
    mpoa_nsap 47.0091810000000050E2097801.00500B306444.00, opcode 4
00:25:57:LEC ATM1/1/0.1:lec_add_mps:add mac 0050.0b30.6440, mps_mac 0x617E372
C
00:25:57:LEC ATM1/1/0.1:mpoa_device_change:lec_nsap 47.0091810000000050E20978
01.00500B306440.02, appl_type 5
    mpoa_nsap 47.0091810000000050E2097801.00500B306444.00, opcode 5
00:25:57:LEC ATM1/1/0.1: mps_mac 0050.0b30.6440
00:25:57:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:25:57:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A
29AF42D.00
Router(config-subif)#exit
Router(config)#exit

```

The following output is from the **debug lane client mpoa** command when the ATM major interface is shut down:

```

Router# debug lane client mpoa

Router# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int atm 1/1/0
Router(config-if)# shutdown
Router(config-if)#
00:26:28:LANE ATM1/1/0:atm hardware reset
00:26:28:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:26:28:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:28:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:28:%MPOA-5-UPDOWN:MPC mpc2:state changed to down
00:26:28:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0
00:26:30:%LINK-5-CHANGED:Interface ATM1/1/0, changed state to administratively
down
00:26:30:LANE ATM1/1/0:atm hardware reset
00:26:31:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1/0, changed stat
e to down
Router(config-if)#
00:26:31:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0

```

```

00:26:32:LANE ATM1/1/0:atm hardware reset
00:26:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:34:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
Router(config-if)# exit
Router(config)# exit

```

The following output is from the **debug lane client mpoa** command when the ATM major interface is started:

```
Router# debug lane client mpoa
```

```

Router# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# int atm 1/1/0
Router(config-if)# no shutdown
00:26:32:LANE ATM1/1/0:atm hardware reset
00:26:32:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:26:34:%LINK-3-UPDOWN:Interface ATM1/1/0, changed state to down
00:26:34:LANE ATM1/1/0:atm hardware reset
00:26:41:%LINK-3-UPDOWN:Interface ATM1/1/0, changed state to up
00:26:42:%LINEPROTO-5-UPDOWN:Line protocol on Interface ATM1/1/0, changed state to up
00:27:10:%LANE-6-INFO:ATM1/1/0:ILMI prefix add event received
00:27:10:LANE ATM1/1/0:prefix add event for 470091810000000050E2097801 ptr=0x617BFC0C len=13
00:27:10: the current first prefix is now:470091810000000050E2097801
00:27:10:%ATMSSCOP-5-SSCOPINIT:- Intf :ATM1/1/0, Event :Rcv End, State :Active.
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0

00:27:10:%LANE-3-NOREGILMI:ATM1/1/0.1 LEC cannot register 47.0091810000000050E2097801.0050A29AF428.01 with ILMI
00:27:10:%LANE-6-INFO:ATM1/1/0:ILMI prefix add event received
00:27:10:LANE ATM1/1/0:prefix add event for 470091810000000050E2097801 ptr=0x617B8E6C len=13
00:27:10: the current first prefix is now:470091810000000050E2097801
00:27:10:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to down
00:27:10:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:DOWN
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 0

00:27:10:%MPOA-5-UPDOWN:MPC mpc2:state changed to up
00:27:10:LEC ATM1/1/0.1:mpoa_to_lec:appl 6, opcode 1

00:27:12:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_CONFIG_RSP, num_tlvs 14
00:27:12:LEC ATM1/1/0.1:elan id from LECS set to 300
00:27:12:LEC ATM1/1/0.1:lec_process_lane_tlv:msg LANE_JOIN_RSP, num_tlvs 1
00:27:12:LEC ATM1/1/0.1:elan id from LES set to 300
00:27:12:LEC ATM1/1/0.1:lec_append_mpoa_dev_tlv:
00:27:12:LEC ATM1/1/0.1:got mpoa client addr 47.0091810000000050E2097801.0050A29AF42D.00
00:27:12:%LANE-5-UPDOWN:ATM1/1/0.1 elan elan2:LE Client changed state to up
00:27:12:LEC ATM1/1/0.1:lec_inform_mpoa_state_chg:UP
Router(config-if)#exit
Router(config)#exit

```

Related Commands

Command	Description
debug modem traffic	Displays MPC debug information.
debug mpoa server	Displays information about the MPOA server.