

# debug piafs events

To check the debugging messages for PIAFS calls, use the **debug piafs events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug piafs events**

**no debug piafs events**

## Syntax Description

This command has no arguments or keywords.

## Defaults

This command has no default behavior or values.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.2(8)T	This command was introduced on Cisco 803, Cisco 804, and Cisco 813 routers.

## Usage Guidelines

The **debug piafs events** command provides debugging information for the Personal Handyphone System (PHS) Internet Access Forum Standard (PIAFS) calls on the router, including the inband negotiation process.

## Examples

The **debug piafs events** command was configured to provide the following information for PIAFS calls.

```
Router# debug piafs events

02:16:39:PIAFS events debugging is on
02:16:167516180371:PIAFS: RX <- CDAPI :cdapi_route_call Request
02:16:167517398148:PIAFS: RX <- CDAPI :CDAPI_MSG_CONNECT_IND
02:16:171798691839:PIAFS: TX -> CDAPI :CDAPI_MSG_SUBTYPE_ALERT_REQ
02:16:167503724545:PIAFS: TX -> CDAPI :CDAPI_MSG_CONNECT_RESP
02:16:167503765504:PIAFS: TX -> CDAPI :CDAPI_MSG_CONN_ACTIVE_REQ
02:16:167503724544:PIAFS: RX <- CDAPI :CDAPI_MSG_CONN_ACTIVE_IND
02:16:171798691839:PIAFS:Network allotted Channel :B1
02:16:167503765504:PIAFS:Enabling QMC in PIAFS mode for B1
02:16:171798691839:PIAFS:piafs_driver_enable_settings()
02:16:167503765504:PIAFS:The speed is :64
02:16:167503724544:PIAFS:Starting 64 kbps PIAFS Incoming
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:13 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:Updating conf resp num
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:1 RSN:1 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:14 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:2 RSN:2 CRSN:13 SISN:
255]
```

```

02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:15 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:3 RSN:3 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:16 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:4 RSN:4 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:17 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:5 RSN:5 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:18 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:6 RSN:6 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- NEGO_SYNC_REQUEST[GSN:19 RSN:1 CRSN:1 SISN:
255]
02:16:39:PIAFS:TX -> NEGO_SYNC_RECEPTION[GSN:7 RSN:7 CRSN:13 SISN:
255]
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:Piafs layer up & Main FSM set to DATA
02:16:39:PIAFS:Compression v42bis enabled
02:16:39:PIAFS:V42BIS:v42bis_init()
02:16:39:PIAFS:V42BIS:v42bis_init()
02:16:39:PIAFS:V42BIS:Negotiated Values for P1, P2 are - 4096 , 250
02:16:39:PIAFS:Incoming call invoking ISDN_CALL_CONNECT
02:16:39:%LINK-3-UPDOWN:Interface BRI0:1, changed state to up
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80

```

```
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:RX <- CONTROL_REQUEST(comm parameter)[Seq No:0]
02:16:39:PIAFS:Rx Parameters:
02:16:39:PIAFS: Data Protocol:Version 1
02:16:39:PIAFS: Control Protocol:Version 1
02:16:39:PIAFS: RTF value:9
02:16:39:PIAFS: Compression:V.42bis
02:16:39:PIAFS: Frame Length:80
02:16:39:PIAFS: Frame Number:63
02:16:39:PIAFS:TX -> CONTROL_RECEPTION[0]
02:16:39:PIAFS:ACKed all the Rx control parameters
02:16:39:PIAFS:piafs_setmap() tx_map FFFFFFFF
02:16:39:PIAFS:piafs_setmap() rx_map 0
02:16:41:PIAFS:PPP:Autoselect sample 7E
02:16:41:PIAFS:PPP:Autoselect sample 7EFF
02:16:41:PIAFS:PPP:Autoselect sample 7EFF7D
02:16:41:PIAFS:PPP:Autoselect sample 7EFF7D23
02:16:41:PIAFS:piafs_setmap() tx_map FFFFFFFF
02:16:41:PIAFS:piafs_setmap() rx_map 0
02:16:42:PIAFS:piafs_setmap() tx_map A0000
02:16:42:PIAFS:piafs_setmap() rx_map 0
```

Table 191 describes significant fields in the display. The remaining fields are self-explanatory.

**Table 191** *debug piafs events Field Descriptions*

Field	Description
RX <- CDAPI :cdapi_route_call Request	The call distributor application programming interface (CDAPI) in the router receives an ISDN call request from the switch.
RX <- CDAPI :CDAPI_MSG_CONNECT_IND	The CDAPI in the router receives a connection indicator message from the switch.
TX -> CDAPI :CDAPI_MSG_SUBTYPE_ALERT_REQ	The CDAPI in the router transmits an alert request to the switch.
TX -> CDAPI :CDAPI_MSG_CONNECT_RESP	The CDAPI in the router transmits a connect response message to the switch.
TX -> CDAPI :CDAPI_MSG_CONN_ACTIVE_REQ	The CDAPI in the router transmits a connection active request to the switch.
RX <-CDAPI:CDAPI_MSG_CONN_ACTIVE_IND	The CDAPI in the router receives a connection active indicator from the switch.
Enabling QMC in PIAFS mode for B1	QMC (global multichannel parameters) are being enabled in PIAFS mode for the B1 channel.
piafs_driver_enable_settings()	The PIAFS driver is enabling the settings.
Starting 64 kbps PIAFS Incoming	The speed of the transmission in kbps. In this case, the speed is 64 kbps.
RX <- NEGO_SYNC_REQUEST[GSN: RSN: CRSN: SISN:]	The router receives a PIAFS negotiation synchronization request frame from the peer PIAFS device. The frame contains the following: general sequence number (GSN), reception sequence number (RSN), confirmation response sequence number (CRSN), and synchronization initiation sequence number (SISN).
Updating conf resp num	The confirmation response number is being updated.
TX -> NEGO_SYNC_RECEPTION[GSN: RSN: CRSN: SISN: ]	The router transmits a PIAFS negotiation synchronization reception message to the peer PIAFS device. The message includes the GSN, RSN, CRSN, and SISN.
RX <- CONTROL_REQUEST	The router receives a PIAFS control request frame that includes communication parameters.
Rx Parameters	The communication parameters are as follows.
Data Protocol	The version of the data protocol.
Control Protocol	The version of the control protocol.
RTF value	Round-trip frame value.
Compression	The compression standard.
Frame Length	The length of the frame, in bytes.

**Table 191** *debug piafs events Field Descriptions (continued)*

<b>Field</b>	<b>Description</b>
Frame Number	The number of packets per frame.
TX -> CONTROL_RECEPTION	The router transmits a PIAFS control reception frame.
ACKed all the Rx control parameters	The control reception frame acknowledges all the communication parameters that were received from the peer.
Piafs layer up & Main FSM set to DATA	The PIAFS protocol is active on the router. The router is ready to receive data from the peer device.
Compression v42bis enabled	The compression protocol v42bis is enabled.
V42BIS:v42bis_init()	The v42bis compression protocol has been initiated.
V42BIS:Negotiated Values for P1, P2 are - 4096 , 250	In this example, P1 is the total count of encoded words when v42bis compression is enabled. P2 is the maximum letter line length for the V42bis compression.
Incoming call invoking ISDN_CALL_CONNECT	An incoming ISDN call connection message is received.
PPP	The PPP layer on the router becomes active and starts to process the PPP frame from the peer PIAFS device.

# debug pots

To display information on the telephone interfaces, use the **debug pots** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug pots** {**driver** | **csm**} [**1** | **2**]

**no debug pots** {**driver** | **csm**} [**1** | **2**]

## Syntax Description

<b>driver</b>	Displays driver debug information.
<b>csm</b>	Displays CSM debug information.
<b>1</b>	(Optional) Displays information for telephone port 1 only.
<b>2</b>	(Optional) Displays information for telephone port 2 only.

## Command Modes

Privileged EXEC

## Usage Guidelines

The **debug pots** command displays driver and CSM debug information for telephone ports 1 and 2.

## Examples

The following is a sample display from the **debug pots driver 1** command. This sample display indicates that the telephone port driver is not receiving caller ID information from the ISDN line. Therefore, the analog caller ID device attached to the telephone port does not display caller ID information.

```
Router# debug pots driver 1

00:01:51:POTS DRIVER port=1 activate ringer: cadence=0 callerId=Unknown
00:01:51:POTS DRIVER port=1 state=Idle drv_event=RING_EVENT
00:01:51:POTS DRIVER port=1 enter_ringing
00:01:51:POTS DRIVER port=1 cmd=19
00:01:51:POTS DRIVER port=1 activate disconnect
00:01:51:POTS DRIVER port=1 state=Ringling drv_event=DISCONNECT_EVENT
00:01:51:POTS DRIVER port=1 cmd=1A
00:01:51:POTS DRIVER port=1 enter_idle
00:01:51:POTS DRIVER port=1 ts connect: 0 0
00:01:51:POTS DRIVER port=1 cmd=D
00:01:51:POTS DRIVER port=1 report onhook
00:01:51:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:01:51:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:01:51:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:01:51:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:01:53:POTS DRIVER port=1 activate ringer: cadence=0 callerId=Unknown
00:01:53:POTS DRIVER port=1 state=Idle drv_event=RING_EVENT
00:01:53:POTS DRIVER port=1 enter_ringing
00:01:53:POTS DRIVER port=1 cmd=19
00:01:55:POTS DRIVER port=1 cmd=1A
00:02:49:POTS DRIVER port=1 state=Ringling drv_event=OFFHOOK_EVENT
00:02:49:POTS DRIVER port=1 cmd=1A
00:02:49:POTS DRIVER port=1 enter_suspend
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 report offhook
00:02:49:POTS DRIVER port=1 activate connect: endpt=1 calltype=TWO_PARTY_CALL
00:02:49:POTS DRIVER port=1 state=Suspend drv_event=CONNECT_EVENT
```

```

00:02:49:POTS DRIVER port=1 enter_connect: endpt=1 calltype=0
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 ts connect: 1 0
00:02:49:POTS DRIVER port=1 activate connect: endpt=1 calltype=TWO_PARTY_CALL
00:02:49:POTS DRIVER port=1 state=Connect drv_event=CONNECT_EVENT
00:02:49:POTS DRIVER port=1 enter_connect: endpt=1 calltype=0
00:02:49:POTS DRIVER port=1 cmd=A
00:02:49:POTS DRIVER port=1 ts connect: 1 0
00:02:55:POTS DRIVER port=1 state=Connect drv_event=ONHOOK_EVENT
00:02:55:POTS DRIVER port=1 enter_idle
00:02:55:POTS DRIVER port=1 ts connect: 0 0
00:02:55:POTS DRIVER port=1 cmd=D
00:02:55:POTS DRIVER port=1 report onhook
00:02:55:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:02:55:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT
00:02:55:POTS DRIVER port=1 activate tone=SILENCE_TONE
00:02:55:POTS DRIVER port=1 state=Idle drv_event=TONE_EVENT

```

The following is sample display from the **debug pots csm 1** command. This sample display indicates that a dial peer contains an invalid destination pattern (555-1111).

```
Router# debug pots csm 1
```

```

01:57:28:EVENT_FROM_ISDN:dchanidb=0x66CB38, call_id=0x11, ces=0x2 bchan=0x0, event=0x1,
cause=0x0
01:57:28:Dial peer not found, route call to port 1
01:57:28:CSM_PROC_IDLE:CSM_EVENT_ISDN_CALL, call_id=0x11, port=1
01:57:28:Calling number '5551111'
01:57:40:CSM_PROC_RINGING:CSM_EVENT_VDEV_OFFHOOK, call_id=0x11, port=1
01:57:40:EVENT_FROM_ISDN:dchan_idb=0x66CB38, call_id=0x11, ces=0x2 bchan=0x0, event=0x4,
cause=0x0
01:57:40:CSM_PROC_CONNECTING:CSM_EVENT_ISDN_CONNECTED, call_id=0x11, port=1
01:57:47:CSM_PROC_CONNECTING:CSM_EVENT_VDEV_ONHOOK, call_id=0x11, port=1
01:57:201863503872: %ISDN-6-DISCONNECT:Interface BRI0:1 disconnected from unknown, call
lasted 5485 seconds
01:57:47: %ISDN-6-DISCONNECT:Interface BRI0:1 disconnected from unknown, call lasted 5485
seconds
01:57:47:EVENT_FROM_ISDN:dchan_idb=0x66CB38, call_id=0x11, ces=0x2 bchan=0xFFFFFFFF,
event=0x0, cause=0x1
01:57:47:CSM_PROC_NEAR_END_DISCONNECT:CSM_

```

# debug pots csm

To activate events from which an application can determine and display the status and progress of calls to and from POTS ports, use the **debug pots csm** EXEC command.

## debug pots csm

**Syntax Description** This command has no arguments or keywords.

**Command Modes** EXEC

Command History	Release	Modification
	12.1.(2)XF	This command was introduced on the Cisco 800 series routers.

**Examples** To see debug messages, enter the **logging console** global configuration mode command as follows:

```
router(config)# logging console
```

```
router(config)# exit
```

Debug messages are displayed in one of two formats that are relevant to the POTS dial feature:

```
hh:mm:ss: CSM_STATE: CSM_EVENT, call id = ??, port = ?
```

or

```
hh:mm:ss: EVENT_FROM_ISDN:dchan_idb=0x???????, call_id=0x????, ces=? bchan=0x?????????, event=0x?, cause=0x??
```

[Table 192](#) describes the significant fields shown in the display.

**Table 192 debug pots csm Field Descriptions**

Command Elements	Description
hh:mm:ss	Timestamp (in hours, minutes, and seconds).
CSM_STATE	One of the call CSM states listed in <a href="#">Table 193</a> .
CSM_EVENT	One of the CSM events listed in <a href="#">Table 194</a> .
call id	Hexadecimal value from 0x00 to 0xFF.
port	Telephone port 1 or 2.
EVENT_FROM_ISDN	A CSM event. <a href="#">Table 194</a> shows a list of CSM events.
dchan_idb	Internal data structure address.
ces	Connection end point suffix used by ISDN.
bchan	Channel used by the call. A value of 0xFFFFFFFF indicates that a channel is not assigned.

**Table 192** *debug pots csm Field Descriptions (continued)*

Command Elements	Description
event	A hexadecimal value that is translated into a CSM event. <a href="#">Table 195</a> shows a list of events and the corresponding CSM events.
cause	A hexadecimal value that is given to call-progressing events. <a href="#">Table 196</a> shows a list of cause values and definitions.

[Table 193](#) shows the values for CSM states.

**Table 193** *CSM States*

CSM State	Description
CSM_IDLE_STATE	Telephone on the hook.
CSM_RINGING	Telephone ringing.
CSM_SETUP	Setup for outgoing call in progress.
CSM_DIALING	Dialing number of outgoing call.
CSM_IVR_DIALING	Interactive voice response (IVR) for Japanese telephone dialing.
CSM_CONNECTING	Waiting for carrier to connect the call.
CSM_CONNECTED	Call connected.
CSM_DISCONNECTING	Waiting for carrier to disconnect the call.
CSM_NEAR_END_DISCONNECTING	Waiting for carrier to disconnect the call.
CSM_HARD_HOLD	Call on hard hold.
CSM_CONSULTATION_HOLD	Call on consultation hold.
CSM_WAIT_FOR_HOLD	Waiting for carrier to put call on hard hold.
CSM_WAIT_FOR_CONSULTATION_HOLD	Waiting for carrier to put call on consultation hold.
CSM_CONFERENCE	Waiting for carrier to complete call conference.
CSM_TRANSFER	Waiting for carrier to transfer call.
CSM_APPLIC_DIALING	Call initiated from Cisco IOS CLI.

[Table 194](#) shows the values for CSM events.

**Table 194** *CSM Events*

CSM Events	Description
CSM_EVENT_INTER_DIGIT_TIMEOUT	Time waiting for dial digits has expired.
CSM_EVENT_TIMEOUT	Near- or far-end disconnect timeout.
CSM_EVENT_ISDN_CALL	Incoming call.
CSM_EVENT_ISDN_CONNECTED	Call connected.
CSM_EVENT_ISDN_DISCONNECT	Far end disconnected.
CSM_EVENT_ISDN_DISCONNECTED	Call disconnected.

**Table 194 CSM Events (continued)**

CSM Events	Description
CSM_EVENT_ISDN_SETUP	Outgoing call requested.
CSM_EVENT_ISDN_SETUP_ACK	Outgoing call accepted.
CSM_EVENT_ISDN_PROC	Call proceeding and dialing completed.
CSM_EVENT_ISDN_CALL_PROGRESSING	Call being received in band tone.
CSM_EVENT_ISDN_HARD_HOLD	Call on hard hold.
CSM_EVENT_ISDN_HARD_HOLD_REJ	Hold attempt rejected.
CSM_EVENT_ISDN_CHOLD	Call on consultation hold.
CSM_EVENT_ISDN_CHOLD_REJ	Consultation hold attempt rejected.
CSM_EVENT_ISDN_RETRIEVED	Call retrieved.
CSM_EVENT_ISDN_RETRIEVE_REJ	Call retrieval attempt rejected.
CSM_EVENT_ISDN_TRANSFERRED	Call transferred.
CSM_EVENT_ISDN_TRANSFER_REJ	Call transfer attempt rejected.
CSM_EVENT_ISDN_CONFERENCE	Call conference started.
CSM_EVENT_ISDN_CONFERENCE_REJ	Call conference attempt rejected.
CSM_EVENT_ISDN_IF_DOWN	ISDN interface down.
CSM_EVENT_ISDN_INFORMATION	ISDN information element received (used by NTT IVR application).
CSM_EVENT_VDEV_OFFHOOK	Telephone off the hook.
CSM_EVENT_VDEV_ONHOOK	Telephone on the hook.
CSM_EVENT_VDEV_FLASHHOOK	Telephone hook switch has flashed.
CSM_EVENT_VDEV_DIGIT	DTMF digit has been detected.
CSM_EVENT_VDEV_APPLICATION_CALL	Call initiated from Cisco IOS CLI.

Table 195 shows the values for events that are translated into CSM events.

**Table 195 Event Values**

Hexadecimal Value	Event	CSM Event
0x0	DEV_IDLE	CSM_EVENT_ISDN_DISCONNECTED
0x1	DEV_INCALL	CSM_EVENT_ISDN_CALL
0x2	DEV_SETUP_ACK	CSM_EVENT_ISDN_SETUP_ACK
0x3	DEV_CALL_PROC	CSM_EVENT_ISDN_PROC
0x4	DEV_CONNECTED	CSM_EVENT_ISDN_CONNECTED
0x5	DEV_CALL_PROGRESSING	CSM_EVENT_ISDN_CALL_PROGRESSING
0x6	DEV_HOLD_ACK	CSM_EVENT_ISDN_HARD_HOLD
0x7	DEV_HOLD_REJECT	CSM_EVENT_ISDN_HARD_HOLD_REJ
0x8	DEV_CHOLD_ACK	CSM_EVENT_ISDN_CHOLD

**Table 195** Event Values (continued)

Hexadecimal Value	Event	CSM Event
0x9	DEV_CHOLD_REJECT	CSM_EVENT_ISDN_CHOLD_REJ
0xa	DEV_RETRIEVE_ACK	CSM_EVENT_ISDN_RETRIEVED
0xb	DEV_RETRIEVE_REJECT	CSM_EVENT_ISDN_RETRIEVE_REJ
0xc	DEV_CONFR_ACK	CSM_EVENT_ISDN_CONFERENCE
0xd	DEV_CONFR_REJECT	CSM_EVENT_ISDN_CONFERENCE_REJ
0xe	DEV_TRANS_ACK	CSM_EVENT_ISDN_TRANSFERRED
0xf	DEV_TRANS_REJECT	CSM_EVENT_ISDN_TRANSFER_REJ

Table 196 shows cause values that are assigned only to call-progressing events.

**Table 196** Cause Values

Hexadecimal Value	Cause Definitions
0x01	UNASSIGNED_NUMBER
0x02	NO_ROUTE
0x03	NO_ROUTE_DEST
0x04	NO_PREFIX
0x06	CHANNEL_UNACCEPTABLE
0x07	CALL_AWARDED
0x08	CALL_PROC_OR_ERROR
0x09	PREFIX_DIALED_ERROR
0x0a	PREFIX_NOT_DIALED
0x0b	EXCESSIVE_DIGITS
0x0d	SERVICE_DENIED
0x10	NORMAL_CLEARING
0x11	USER_BUSY
0x12	NO_USER_RESPONDING
0x13	NO_USER_ANSWER
0x15	CALL_REJECTED
0x16	NUMBER_CHANGED
0x1a	NON_SELECTED_CLEARING
0x1b	DEST_OUT_OF_ORDER
0x1c	INVALID_NUMBER_FORMAT
0x1d	FACILITY_REJECTED
0x1e	RESP_TO_STAT_ENQ
0x1f	UNSPECIFIED_CAUSE
0x22	NO_CIRCUIT_AVAILABLE

**Table 196 Cause Values (continued)**

Hexadecimal Value	Cause Definitions
0x26	NETWORK_OUT_OF_ORDER
0x29	TEMPORARY_FAILURE
0x2a	NETWORK_CONGESTION
0x2b	ACCESS_INFO_DISCARDED
0x2c	REQ_CHANNEL_NOT_AVAIL
0x2d	PRE_EMPTED
0x2f	RESOURCES_UNAVAILABLE
0x32	FACILITY_NOT_SUBSCRIBED
0x33	BEARER_CAP_INCOMPAT
0x34	OUTGOING_CALL_BARRED
0x36	INCOMING_CALL_BARRED
0x39	BEARER_CAP_NOT_AUTH
0x3a	BEAR_CAP_NOT_AVAIL
0x3b	CALL_RESTRICTION
0x3c	REJECTED_TERMINAL
0x3e	SERVICE_NOT_ALLOWED
0x3f	SERVICE_NOT_AVAIL
0x41	CAP_NOT_IMPLEMENTED
0x42	CHAN_NOT_IMPLEMENTED
0x45	FACILITY_NOT_IMPLEMENT
0x46	BEARER_CAP_RESTRICTED
0x4f	SERV_OPT_NOT_IMPLEMENT
0x51	INVALID_CALL_REF
0x52	CHAN_DOES_NOT_EXIST
0x53	SUSPENDED_CALL_EXISTS
0x54	NO_CALL_SUSPENDED
0x55	CALL_ID_IN_USE
0x56	CALL_ID_CLEARED
0x58	INCOMPATIBLE_DEST
0x5a	SEGMENTATION_ERROR
0x5b	INVALID_TRANSIT_NETWORK
0x5c	CS_PARAMETER_NOT_VALID
0x5f	INVALID_MSG_UNSPEC
0x60	MANDATORY_IE_MISSING
0x61	NONEXISTENT_MSG
0x62	WRONG_MESSAGE

**Table 196 Cause Values (continued)**

Hexadecimal Value	Cause Definitions
0x63	BAD_INFO_ELEM
0x64	INVALID_ELEM_CONTENTS
0x65	WRONG_MSG_FOR_STATE
0x66	TIMER_EXPIRY
0x67	MANDATORY_IE_LEN_ERR
0x6f	PROTOCOL_ERROR
0x7f	INTERWORKING_UNSPEC

**Examples**

This section provides debug output examples for three call scenarios, displaying the sequence of events that occur during a POTS dial call or POTS disconnect call.

**Call Scenario 1**

In this example call scenario, port 1 is on the hook, the application dial is set to call 4085552221, and the far-end successfully connects.

```
Router# debug pots csm

Router# test pots 1 dial 4085552221#

Router#
```

The following screen output shows an event indicating that port 1 is being used by the dial application:

```
01:58:27: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
```

The following screen output shows events indicating that the CSM is receiving the application digits of the number to dial:

```
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:58:27: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
```

The following screen output shows that the telephone connected to port 1 is off the hook:

```
01:58:39: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
```

The following screen output shows a call-proceeding event pair indicating that the router ISDN software has sent the dialed digits to the ISDN switch:

```
01:58:40: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0x0,
event=0x3, cause=0x0
01:58:40: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id =
0x8004, port = 1
```

The following screen output shows the call-progressing event pair indicating that the telephone at the far end is ringing:

```
01:58:40: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x0
01:58:40: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8004, port
= 1
```

The following screen output shows a call-connecting event pair indicating that the telephone at the far end has answered:

```
01:58:48: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x4, cause=0x0
01:58:48: CSM_PROC_CONNECTING: CSM_EVENT_ISDN_CONNECTED, call id = 0x8004, port = 1
```

The following screen output shows a call-progressing event pair indicating that the telephone at the far end has hung up and that the calling telephone is receiving an in-band tone from the ISDN switch:

```
01:58:55: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x10
01:58:55: CSM_PROC_CONNECTED: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8004, port = 1
```

The following screen output shows that the telephone connected to port 1 has hung up:

```
01:58:57: CSM_PROC_CONNECTED: CSM_EVENT_VDEV_ONHOOK, call id = 0x8004, port = 1
```

The following screen output shows an event pair indicating that the call has been terminated:

```
01:58:57: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8004, ces=0x1 bchan=0xFFFFFFFF,
event=0x0, cause=0x0
01:58:57: CSM_PROC_NEAR_END_DISCONNECT: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8004,
port = 1
813_local#
```

## Call Scenario 2

In this example scenario, port 1 is on the hook, the application dial is set to call 4085552221, and the destination number is busy.

```
Router# debug pots csm

Router# test pots 1 dial 4085552221#

Router#
```

The following screen output shows that port 1 is used by the dial application:

```
01:59:42: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
```

The following screen output shows the events indicating that the CSM is receiving the application digits of the number to call:

```
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
01:59:42: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
```

The following screen output shows an event indicating that the telephone connected to port 1 is off the hook:

```
01:59:52: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1
```

The following screen output shows a call-proceeding event pair indicating that the telephone at the far end is busy:

```
01:59:52: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0x0,
event=0x3, cause=0x11
01:59:52: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id = 0x8005, port = 1
```

The following screen output shows a call-progressing event pair indicating that the calling telephone is receiving an in-band busy tone from the ISDN switch:

```
01:59:58: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0xFFFFFFFF,
event=0x5, cause=0x0
01:59:58: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8005, port
= 1
```

The following screen output shows an event indicating that the calling telephone has hung up:

```
02:00:05: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_VDEV_ONHOOK, call id = 0x8005, port = 1
```

The following screen output shows an event pair indicating that the call has been terminated:

```
02:00:05: EVENT_FROM_ISDN:dchan_idb=0x280AF38, call_id=0x8005, ces=0x1 bchan=0xFFFFFFFF,
event=0x0, cause=0x0
02:00:05: CSM_PROC_NEAR_END_DISCONNECT: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8005,
port = 1
```

### Call Scenario 3

In this example call scenario, port 1 is on the hook, the application dial is set to call 408-666-1112, the far end successfully connects, and the command **test pots disconnect** terminates the call:

```
Router# debug pots csm

Router# test pots 1 dial 4086661112

Router#
```

The following screen output follows the same sequence of events as shown in Call Scenario 1:

```
1d03h: CSM_PROC_IDLE: CSM_EVENT_VDEV_APPLICATION_CALL, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_DIGIT, call id = 0x0, port = 1
1d03h: CSM_PROC_APPLIC_DIALING: CSM_EVENT_VDEV_OFFHOOK, call id = 0x0, port = 1

1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
      bchan=0x0, event=0x3, cause=0x0
1d03h: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_PROC, call id = 0x8039, port = 1

1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1
      bchan=0xFFFFFFFF, event=0x5, cause=0x0
```

```
1d03h: CSM_PROC_ENBLOC_DIALING: CSM_EVENT_ISDN_CALL_PROGRESSING, call id = 0x8039,  
      port = 1
```

```
Router# test pots 1 disconnect
```

The **test pots disconnect** command disconnects the call before you physically need to put the telephone back on the hook:

```
1d03h: CSM_PROC_CONNECTING: CSM_EVENT_VDEV_APPLICATION_HANGUP_CALL, call id = 0x8039,  
      port = 1
```

```
1d03h: EVENT_FROM_ISDN:dchan_idb=0x2821F38, call_id=0x8039, ces=0x1  
      bchan=0xFFFFFFFF, event=0x0, cause=0x0
```

```
1d03h: CSM_PROC_DISCONNECTING: CSM_EVENT_ISDN_DISCONNECTED, call id = 0x8039,  
      port = 1
```

```
1d03h: CSM_PROC_DISCONNECTING: CSM_EVENT_TIMEOUT, call id = 0x8039, port = 1
```

# debug ppp

To display information on traffic and exchanges in an internetwork implementing the PPP, use the **debug ppp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug ppp** { **packet** | **negotiation** | **error** | **authentication** | **compression** | **cbcp** }

**no debug ppp** { **packet** | **negotiation** | **error** | **authentication** | **compression** | **cbcp** }

## Syntax Description

<b>packet</b>	Displays PPP packets being sent and received. (This command displays low-level packet dumps.)
<b>negotiation</b>	Displays PPP packets sent during PPP startup, where PPP options are negotiated.
<b>error</b>	Displays protocol errors and error statistics associated with PPP connection negotiation and operation.
<b>authentication</b>	Displays authentication protocol messages, including Challenge Authentication Protocol (CHAP) packet exchanges and Password Authentication Protocol (PAP) exchanges.
<b>compression</b>	Displays information specific to the exchange of PPP connections using MPPC. This command is useful for obtaining incorrect packet sequence number information where MPPC compression is enabled.
<b>cbcp</b>	Displays protocol errors and statistics associated with PPP connection negotiations using MSCB.

## Command Modes

Privileged EXEC

## Usage Guidelines

Use the **debug ppp** command when trying to find the following:

- The Network Control Protocols (NCPs) that are supported on either end of a PPP connection
- Any loops that might exist in a PPP internetwork
- Nodes that are (or are not) properly negotiating PPP connections
- Errors that have occurred over the PPP connection
- Causes for CHAP session failures
- Causes for PAP session failures
- Information specific to the exchange of PPP connections using the Callback Control Protocol (CBCP), used by Microsoft clients
- Incorrect packet sequence number information where MPPC compression is enabled

Refer to Internet RFCs 1331, 1332, and 1333 for details concerning PPP-related nomenclature and protocol information.



### Caution

The **debug ppp compression** command is CPU-intensive and should be used with caution. This command should be disabled immediately after debugging.

## Examples

The following is sample output from the **debug ppp packet** command as seen from the Link Quality Monitor (LQM) side of the connection. This display example depicts packet exchanges under normal PPP operation.

```
Router# debug ppp packet

PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 3 len = 12
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 4 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 4 len = 12
PPP Serial4: O LCP ECHOREP(A) id 4 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 5 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 5 len = 12
PPP Serial4: O LCP ECHOREP(A) id 5 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 6 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 6 len = 12
PPP Serial4: O LCP ECHOREP(A) id 6 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 7 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 7 len = 12
PPP Serial4: O LCP ECHOREP(A) id 7 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

Table 197 describes the significant fields shown in the display.

**Table 197** debug ppp packet Field Descriptions

Field	Description
PPP	PPP debugging output.
Serial4	Interface number associated with this debugging information.
(o), O	Packet was detected as an output packet.
(i), I	Packet was detected as an input packet.
lcp_slqr()	Procedure name; running LQM, send a Link Quality Report (LQR).
lcp_rlqr()	Procedure name; running LQM, received an LQR.
input (C021)	Router received a packet of the specified packet type (in hexadecimal notation). A value of C025 indicates packet of type LQM.
state = OPEN	PPP state; normal state is OPEN.

**Table 197** *debug ppp packet Field Descriptions (continued)*

Field	Description
magic = D21B4	Magic Number for indicated node; when output is indicated, this is the Magic Number of the node on which debugging is enabled. The actual Magic Number depends on whether the packet detected is indicated as I or O.
datagramsize 52	Packet length including header.
code = ECHOREQ(9)	Identifies the type of packet received. Both forms of the packet, string and hexadecimal, are presented.
len = 48	Packet length without header.
id = 3	ID number per Link Control Protocol (LCP) packet format.
pkt type 0xC025	Packet type in hexadecimal notation; typical packet types are C025 for LQM and C021 for LCP.
LCP ECHOREQ(9)	Echo Request; value in parentheses is the hexadecimal representation of the LCP type.
LCP ECHOREP(A)	Echo Reply; value in parentheses is the hexadecimal representation of the LCP type.

To elaborate on the displayed output, consider the partial exchange. This sequence shows that one side is using ECHO for its keepalives and the other side is using LQRs.

Router# **debug ppp packet**

```
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
PPP Serial4: input(C021) state = OPEN code = ECHOREQ(9) id = 3 len = 12
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The first line states that the router with debugging enabled has sent an LQR to the other side of the PPP connection:

```
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The next two lines indicate that the router has received a packet of type C025 (LQM) and provides details about the packet:

```
PPP Serial4(i): pkt type 0xC025, datagramsize 52
PPP Serial4(i): lcp_rlqr() state = OPEN magic = D3454, len = 48
```

The next two lines indicate that the router received an ECHOREQ of type C021 (LCP). The other side is sending ECHOs. The router on which debugging is configured for LQM but also responds to ECHOs.

```
PPP Serial4(i): pkt type 0xC021, datagramsize 16
PPP Serial4: I LCP ECHOREQ(9) id 3 (C) magic D3454
```

Next, the router is detected to have responded to the ECHOREQ with an ECHOREP and is preparing to send out an LQR:

```
PPP Serial4: O LCP ECHOREP(A) id 3 (C) magic D21B4
PPP Serial4(o): lcp_slqr() state = OPEN magic = D21B4, len = 48
```

The following is sample output from the **debug ppp negotiation** command. This is a normal negotiation, where both sides agree on Network Control Program (NCP) parameters. In this case, protocol type IP is proposed and acknowledged.

Router# **debug ppp negotiation**

```
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = 3D567F8 acked (ok)
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
ppp: ipcp_reqci: returning CONFACK.
      (ok)
PPP Serial4: state = ACKSENT fsm_rconfack(8021): rcvd id 4
```

Table 198 describes significant fields shown in the display.

**Table 198 debug ppp Command Negotiation Field Descriptions**

Field	Description
ppp	PPP debugging output.
sending CONFREQ	Router sent a configuration request.
type = 4 (CI_QUALITYTYPE)	Type of LCP configuration option that is being negotiated and a descriptor. A type value of 4 indicates Quality Protocol negotiation; a type value of 5 indicates Magic Number negotiation.
value = C025/3E8	For Quality Protocol negotiation, indicates NCP type and reporting period. In the example, C025 indicates LQM; 3E8 is a hexadecimal value translating to about 10 seconds (in hundredths of a second).
value = 3D56CAC	For Magic Number negotiation, indicates the Magic Number being negotiated.
received config	Receiving node has received the proposed option negotiation for the indicated option type.
acked	Acknowledgment and acceptance of options.
state = ACKSENT	Specific PPP state in the negotiation process.
ipcp_reqci	IPCP notification message; sending CONFACK.
fsm_rconfack (8021)	Procedure fsm_rconfack processes received CONFACKs, and the protocol (8021) is IP.

The first two lines indicate that the router is trying to bring up LCP and will use the indicated negotiation options (Quality Protocol and Magic Number). The value fields are the values of the options themselves. C025/3E8 translates to Quality Protocol LQM. 3E8 is the reporting period (in hundredths of a second). 3D56CAC is the value of the Magic Number for the router.

```
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
```

The next two lines indicate that the other side negotiated for options 4 and 5 as requested and acknowledged both. If the responding end does not support the options, a CONFREJ is sent by the responding node. If the responding end does not accept the value of the option, a CONFNAK is sent with the value field modified.

```
ppp: received config for type = 4 (QUALITYTYPE) acked
ppp: received config for type = 5 (MAGICNUMBER) value = 3D567F8 acked (ok)
```

The next three lines indicate that the router received a CONFACK from the responding side and displays accepted option values. Use the rcvd id field to verify that the CONFREQ and CONFACK have the same ID field.

```
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5
ppp: config ACK received, type = 4 (CI_QUALITYTYPE), value = C025
ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 3D56CAC
```

The next line indicates that the router has IP routing enabled on this interface and that the IPCP NCP negotiated successfully:

```
ppp: ipcp_reqci: returning CONFACK.
```

In the last line, the state of the router is listed as ACKSENT.

```
PPP Serial4: state = ACKSENT fsm_rconfack(C021): rcvd id 5\
```



```

ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44B7010
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44C1488

```

The following is sample output when no response is detected for configuration requests (with both the **debug ppp negotiation** and **debug ppp packet** command enabled):

```
Router# debug ppp negotiation
```

```
Router# debug ppp packet
```

```

ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: 0 LCP CONFREQ(1) id 14 (12) QUALITYTYPE (8) 192 37 0 0 3 232
MAGICNUMBER (6) 4 77 253 200
ppp: TIMEOUT: Time= 44E0980 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: 0 LCP CONFREQ(1) id 15 (12) QUALITYTYPE (8) 192 37 0 0 3 232
MAGICNUMBER (6) 4 77 253 200
ppp: TIMEOUT: Time= 44E1828 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: 0 LCP CONFREQ(1) id 16 (12) QUALITYTYPE (8) 192 37 0 0 3 232
MAGICNUMBER (6) 4 77 253 200
ppp: TIMEOUT: Time= 44E27C8 State= 3
ppp: sending CONFREQ, type = 4 (CI_QUALITYTYPE), value = C025/3E8
ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 44DFDC8
PPP Serial4: 0 LCP CONFREQ(1) id 17 (12) QUALITYTYPE (8) 192 37 0 0 3 232
MAGICNUMBER (6) 4 77 253 200
ppp: TIMEOUT: Time= 44E3768 State= 3

```

The following is sample output from the **debug ppp error** command. These messages might appear when the Quality Protocol option is enabled on an interface that is already running PPP.

```
Router# debug ppp error
```

```

PPP Serial3(i): rlqr receive failure. successes = 15
PPP: myrcvdiffo = 159 peerxmitdiffo = 41091
PPP: myrcvdiffo = 2183 peerxmitdiffo = 1714439
PPP: threshold = 25
PPP Serial4(i): rlqr transmit failure. successes = 15
PPP: myxmitdiffo = 41091 peerrcvdiffo = 159
PPP: myxmitdiffo = 1714439 peerrcvdiffo = 2183
PPP: l->OutLQRs = 1 LastOutLQRs = 1
PPP: threshold = 25
PPP Serial3(i): lqr_protrej() Stop sending LQRs.
PPP Serial3(i): The link appears to be looped back.

```

Table 199 describes the significant fields shown in the display.

**Table 199 debug ppp Error Field Descriptions**

Field	Description
PPP	PPP debugging output.
Serial3(i)	Interface number associated with this debugging information; indicates that this is an input packet.
rlqr receive failure	Request to negotiate the Quality Protocol option is not accepted.
myrcvdiffp = 159	Number of packets received over the time period.
peerxmitdiffp = 41091	Number of packets sent by the remote node over this period.
myrcvdiffo = 2183	Number of octets received over this period.
peerxmitdiffo = 1714439	Number of octets sent by the remote node over this period.
threshold = 25	Maximum error percentage acceptable on this interface. This percentage is calculated by the threshold value entered in the <b>ppp quality number</b> interface configuration command. A value of 100 – <i>number</i> (100 minus <i>number</i> ) is the maximum error percentage. In this case, a <i>number</i> of 75 was entered. This means that the local router must maintain a minimum 75 percent non-error percentage, or the PPP link will be considered down.
OutLQRs = 1	Local router's current send LQR sequence number.
LastOutLQRs = 1	The last sequence number that the remote node side has seen from the local node.

The following is sample output from the **debug ppp authentication** command. Use this debug command to determine why an authentication fails.

```
Router# debug ppp authentication

Serial0: Unable to authenticate. No name received from peer
Serial0: Unable to validate CHAP response. USERNAME pioneer not found.
Serial0: Unable to validate CHAP response. No password defined for USERNAME pioneer
Serial0: Failed CHAP authentication with remote.
Remote message is Unknown name
Serial0: remote passed CHAP authentication.
Serial0: Passed CHAP authentication with remote.
Serial0: CHAP input code = 4 id = 3 len = 48
```

In general, these messages are self-explanatory. Fields that can show optional output are outlined in [Table 200](#).

**Table 200** *debug ppp authentication Field Descriptions*

Field	Description
Serial0	Interface number associated with this debugging information and CHAP access session in question.
USERNAME pioneer not found.	The name <i>pioneer</i> in this example is the name received in the CHAP response. The router looks up this name in the list of usernames that are configured for the router.
Remote message is Unknown name	The following messages can appear: <ul style="list-style-type: none"> <li>• No name received to authenticate</li> <li>• Unknown name</li> <li>• No secret for given name</li> <li>• Short MD5 response received</li> <li>• MD compare failed</li> </ul>
code = 4	Specific CHAP type packet detected. Possible values are as follows: <ul style="list-style-type: none"> <li>• 1—Challenge</li> <li>• 2—Response</li> <li>• 3—Success</li> <li>• 4—Failure</li> </ul>
id = 3	ID number per LCP packet format.
len = 48	Packet length without header.

The following shows sample output from the **debug ppp** command using the **cbcp** keyword. This output depicts packet exchanges under normal PPP operation where the Cisco access server is waiting for the remote PC to respond to the MCB request. The router also has **debug ppp negotiation** and **service timestamps msec** commands enabled.

Router# **debug ppp cbcp**

```
Dec 17 00:48:11.302: As8 MCB: User mscb Callback Number - Client ANY
Dec 17 00:48:11.306: Async8 PPP: O MCB Request(1) id 1 len 9
Dec 17 00:48:11.310: Async8 MCB: O 1 1 0 9 2 5 0 1 0
Dec 17 00:48:11.314: As8 MCB: O Request Id 1 Callback Type Client-Num delay 0
Dec 17 00:48:13.342: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:13.346: Async8 PPP: O MCB Request(1) id 2 len 9
Dec 17 00:48:13.346: Async8 MCB: O 1 2 0 9 2 5 0 1 0
Dec 17 00:48:13.350: As8 MCB: O Request Id 2 Callback Type Client-Num delay 0
Dec 17 00:48:15.370: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:15.374: Async8 PPP: O MCB Request(1) id 3 len 9
Dec 17 00:48:15.374: Async8 MCB: O 1 3 0 9 2 5 0 1 0
Dec 17 00:48:15.378: As8 MCB: O Request Id 3 Callback Type Client-Num delay 0
Dec 17 00:48:17.398: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:17.402: Async8 PPP: O MCB Request(1) id 4 len 9
Dec 17 00:48:17.406: Async8 MCB: O 1 4 0 9 2 5 0 1 0
Dec 17 00:48:17.406: As8 MCB: O Request Id 4 Callback Type Client-Num delay 0
Dec 17 00:48:19.426: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:19.430: Async8 PPP: O MCB Request(1) id 5 len 9
Dec 17 00:48:19.430: Async8 MCB: O 1 5 0 9 2 5 0 1 0
```

```

Dec 17 00:48:19.434: As8 MCB: O Request Id 5 Callback Type Client-Num delay 0
Dec 17 00:48:21.454: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:21.458: Async8 PPP: O MCB Request(1) id 6 len 9
Dec 17 00:48:21.462: Async8 MCB: O 1 6 0 9 2 5 0 1 0
Dec 17 00:48:21.462: As8 MCB: O Request Id 6 Callback Type Client-Num delay 0
Dec 17 00:48:23.482: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:23.486: Async8 PPP: O MCB Request(1) id 7 len 9
Dec 17 00:48:23.490: Async8 MCB: O 1 7 0 9 2 5 0 1 0
Dec 17 00:48:23.490: As8 MCB: O Request Id 7 Callback Type Client-Num delay 0
Dec 17 00:48:25.510: As8 MCB: Timeout in state WAIT_RESPONSE
Dec 17 00:48:25.514: Async8 PPP: O MCB Request(1) id 8 len 9
Dec 17 00:48:25.514: Async8 MCB: O 1 8 0 9 2 5 0 1 0
Dec 17 00:48:25.518: As8 MCB: O Request Id 8 Callback Type Client-Num delay 0
Dec 17 00:48:26.242: As8 PPP: I pkt type 0xC029, datagramsize 18
Dec 17 00:48:26.246: Async8 PPP: I MCB Response(2) id 8 len 16
Dec 17 00:48:26.250: Async8 MCB: I 2 8 0 10 2 C C 1 32 34 39 32 36 31 33 0
Dec 17 00:48:26.254: As8 MCB: Received response
Dec 17 00:48:26.258: As8 MCB: Response CBK-Client-Num 2 12 12, addr 1-2492613
Dec 17 00:48:26.262: Async8 PPP: O MCB Ack(3) id 9 len 16
Dec 17 00:48:26.266: Async8 MCB: O 3 9 0 10 2 C C 1 32 34 39 32 36 31 33 0
Dec 17 00:48:26.270: As8 MCB: O Ack Id 9 Callback Type Client-Num delay 12
Dec 17 00:48:26.270: As8 MCB: Negotiated MCB with peer
Dec 17 00:48:26.390: As8 LCP: I TERMREQ [Open] id 4 len 8 (0x00000000)
Dec 17 00:48:26.390: As8 LCP: O TERMACK [Open] id 4 len 4
Dec 17 00:48:26.394: As8 MCB: Peer terminating the link
Dec 17 00:48:26.402: As8 MCB: Initiate Callback for mscb at 2492613 using Async

```

The following is sample output from the **debug ppp compression** command with **service timestamps** enabled and shows a typical PPP packet exchange between the router and Microsoft client where the MPPC header sequence numbers increment correctly:

```
Router# debug ppp compression
```

```

00:04:14: BR0:1 MPPC: Decompress - hdr/exp_cc# 0x2003/0x0003
00:04:14: BR0:1 MPPC: Decompress - hdr/exp_cc# 0x2004/0x0004
00:04:14: BR0:1 MPPC: Decompress - hdr/exp_cc# 0x2005/0x0005
00:04:14: BR0:1 MPPC: Decompress - hdr/exp_cc# 0x2006/0x0006
00:04:14: BR0:1 MPPC: Decompress - hdr/exp_cc# 0x2007/0x0007

```

Table 201 describes the fields for the **debug ppp compression** output.

**Table 201** debug ppp compression Field Descriptions

Field	Description
<i>interface</i>	Interface enabled with MPPC.
Decomp - hdr/	Decompression header and bit settings.
exp_cc#	Expected coherency count.
0x2003	Received sequence number.
0x0003	Expected sequence number.

The following shows sample output from **debug ppp negotiation** and **debug ppp error** commands, which can be used to troubleshoot initial PPP negotiation and setup errors. This example shows a virtual interface (virtual interface 1) during normal PPP operation and CCP negotiation.

```
Router# debug ppp negotiation error

Vt1 PPP: Unsupported or un-negotiated protocol. Link arp
VPDN: Chap authentication succeeded for p5200
Vil PPP: Phase is DOWN, Setup
Vil VPDN: Virtual interface created for dinesh@cisco.com
Vil VPDN: Set to Async interface
Vil PPP: Phase is DOWN, Setup
Vil VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
Vil CCP: Re-Syncing history using legacy method
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
Vil PPP: Treating connection as a dedicated line
Vil PPP: Phase is ESTABLISHING, Active Open
Vil LCP: O CONFREQ [Closed] id 1 len 25
Vil LCP:   ACCM 0x000A0000 (0x0206000A0000)
Vil LCP:   AuthProto CHAP (0x0305C22305)
Vil LCP:   MagicNumber 0x000FB69F (0x0506000FB69F)
Vil LCP:   PFC (0x0702)
Vil LCP:   ACFC (0x0802)
Vil VPDN: Bind interface direction=2
Vil PPP: Treating connection as a dedicated line
Vil LCP: I FORCED CONFREQ len 21
Vil LCP:   ACCM 0x000A0000 (0x0206000A0000)
Vil LCP:   AuthProto CHAP (0x0305C22305)
Vil LCP:   MagicNumber 0x12A5E4B5 (0x050612A5E4B5)
Vil LCP:   PFC (0x0702)
Vil LCP:   ACFC (0x0802)
Vil VPDN: PPP LCP accepted sent & rcv CONFACK
Vil PPP: Phase is AUTHENTICATING, by this end
Vil CHAP: O CHALLENGE id 1 len 27 from "1_4000"
Vil CHAP: I RESPONSE id 20 len 37 from "dinesh@cisco.com"
Vil CHAP: O SUCCESS id 20 len 4
Vil PPP: Phase is UP
Vil IPCP: O CONFREQ [Closed] id 1 len 10
Vil IPCP:   Address 15.2.2.3 (0x03060F020203)
Vil CCP: O CONFREQ [Not negotiated] id 1 len 10
Vil CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vil IPCP: I CONFREQ [REQsent] id 1 len 34
Vil IPCP:   Address 0.0.0.0 (0x030600000000)
Vil IPCP:   PrimaryDNS 0.0.0.0 (0x810600000000)
Vil IPCP:   PrimaryWINS 0.0.0.0 (0x820600000000)
Vil IPCP:   SecondaryDNS 0.0.0.0 (0x830600000000)
Vil IPCP:   SecondaryWINS 0.0.0.0 (0x840600000000)
Vil IPCP: Using the default pool
Vil IPCP: Pool returned 11.2.2.5
Vil IPCP: O CONFREQ [REQsent] id 1 len 16
Vil IPCP:   PrimaryWINS 0.0.0.0 (0x820600000000)
Vil IPCP:   SecondaryWINS 0.0.0.0 (0x840600000000)
Vil CCP: I CONFREQ [REQsent] id 1 len 15
Vil CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vil CCP:   Stacker history 1 check mode EXTENDED (0x1105000104)
Vil CCP: Already accepted another CCP option, rejecting this STACKER
Vil CCP: O CONFREQ [REQsent] id 1 len 9
Vil CCP:   Stacker history 1 check mode EXTENDED (0x1105000104)
Vil IPCP: I CONFACK [REQsent] id 1 len 10
Vil IPCP:   Address 15.2.2.3 (0x03060F020203)
Vil CCP: I CONFACK [REQsent] id 1 len 10
Vil CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
Vil CCP: I CONFREQ [ACKrcvd] id 2 len 10
Vil CCP:   MS-PPC supported bits 0x00000001 (0x120600000001)
```

```
Vi1 CCP: O CONFACK [ACKrcvd] id 2 len 10
Vi1 CCP: MS-PPC supported bits 0x00000001 (0x120600000001)
Vi1 CCP: State is Open
Vi1 IPCP: I CONFREQ [ACKrcvd] id 2 len 22
Vi1 IPCP: Address 0.0.0.0 (0x030600000000)
Vi1 IPCP: PrimaryDNS 0.0.0.0 (0x810600000000)
Vi1 IPCP: SecondaryDNS 0.0.0.0 (0x830600000000)
Vi1 IPCP: O CONFNAK [ACKrcvd] id 2 len 22
Vi1 IPCP: Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP: PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP: SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: I CONFREQ [ACKrcvd] id 3 len 22
Vi1 IPCP: Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP: PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP: SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: O CONFACK [ACKrcvd] id 3 len 22
Vi1 IPCP: Address 11.2.2.5 (0x03060B020205)
Vi1 IPCP: PrimaryDNS 171.69.1.148 (0x8106AB450194)
Vi1 IPCP: SecondaryDNS 171.69.2.132 (0x8306AB450284)
Vi1 IPCP: State is Open
Vi1 IPCP: Install route to 11.2.2.5
```

# debug ppp bap

To display general BACP transactions, use the **debug ppp bap** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug ppp bap** [error | event | negotiation]

**no debug ppp bap** [error | event | negotiation]

## Syntax Description

<b>error</b>	(Optional) Displays local errors.
<b>event</b>	(Optional) Displays information about protocol actions and transitions between action states (pending, waiting, idle) on the link.
<b>negotiation</b>	(Optional) Displays successive steps in negotiations between peers.

## Command Modes

Privileged EXEC

## Usage Guidelines

Do not use this command when memory is scarce or in very high traffic situations.

## Examples

The following types of events generate the debug messages displayed in the figures in this section:

- A dial attempt failed.
- A BACP group was created.
- A BACP group was removed.
- The precedence of the group changed.
- Attempting to dial a number.
- Received a BACP message.
- Discarding a BACP message.
- Received an unknown code.
- Cannot find the appropriate BACP group on input.
- Displaying the response type.
- Incomplete mandatory options notification.
- Invalid outgoing message type.
- Unable to build an output message.
- Sending a BACP message.
- Details about the sent message (type of message, its identifier, the virtual access interface that sent it).

The following is sample output from the **debug ppp bap** command:

```
Router# debug ppp bap

BAP Virtual-Access1: group "laudrup" (2) (multilink) without precedence created

BAP laudrup: sending CallReq, id 2, len 38 on BRI3:1 to remote
BAP Virtual-Access1: received CallRsp, id 2, len 13
BAP laudrup: CallRsp, id 2, ACK
BAP laudrup: attempt1 to dial 19995776677 on BRI3
  ---> reason BAP - Multilink bundle overloaded
BAP laudrup: sending StatusInd, id 2, len 44 on Virtual-Access1 to remote
BAP Virtual-Access1: received StatusRsp, id 2, len 1
BAP laudrup: StatusRsp, id 2, ACK
```

[Table 202](#) describes some basic information about the group, the events, and the sent-message details.

**Table 202** *debug ppp bap* Field Descriptions

Field	Description
BAP Virtual-Access1:	Identifier of the virtual access interface in use.
group "laudrup"	Name of the BACP group.
sending CallReq	Action initiated; in this case, sending a call request.
on BRI3:1 to remote	Physical interface being used.
BAP laudrup: attempt1 to dial 19995776677 on BRI3	Call initiated, number being dialed, and physical interface being used.
---> reason BAP - Multilink bundle overloaded	Reason for initiating the BACP call.
BAP laudrup: sending StatusInd, id 2, len 44 on Virtual-Access1 to remote	Details about the sent message: It was a status indication message, had identifier 2, had a BACP datagram length 44, and was sent on virtual access interface 1. You can display information about the virtual access interface by using the <b>show interfaces virtual-access EXEC</b> command. (The length shown at the end of each negotiated option includes the 2-byte type and length header.)

The **debug ppp bap event** command might show state transitions and protocol actions, in addition to the basic **debug ppp bap** command.

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

```
Router# debug ppp bap event

BAP laudrup: Idle --> AddWait
BAP laudrup: AddWait --> AddPending
BAP laudrup: AddPending --> Idle
```

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

```
Router# debug ppp bap event
```

```
Peer does not support a message type  
No response to a particular request  
No response to all request retransmissions  
Not configured to initiate link addition  
Expected action by peer has not occurred  
Exceeded number of retries  
No links available to call out  
Unable to provide phone numbers for callback  
Maximum number of links in the group  
Minimum number of links in the group  
Unable to process link addition at present  
Unable to process link removal at present  
Not configured/unable to initiate link removal  
Link addition completed notification  
Link addition failed notification  
Determination of location of the group config  
Link with specified discriminator not in group  
Link removal failed  
Call failure with status  
Failed to dial specified number  
Discarding retransmission  
Unable to find received identifier  
Received StatusInd when no call pending  
Discarding message with no phone delta  
Unable to send message in particular state  
Received a zero identifier  
Request has precedence
```

The error messages displayed might be added to the basic output when the **debug ppp bap error** command is used. Because the errors are very rare, you might never see these messages.

```
Router# debug ppp bap error
```

```
Unable to find appropriate request for received response  
Invalid message type of queue  
Received request is not part of the group  
Add link attempt failed to locate group  
Remove link attempt failed to locate group  
Unable to inform peer of link addition  
Changing of precedence cannot locate group  
Received short header/illegal length/short packet  
Invalid configuration information length  
Unable to NAK incomplete options  
Unable to determine current number of links  
No interface list to dial on  
Attempt to send invalid data  
Local link discriminator is not in group  
Received response type is incorrect for identifier
```

The messages displayed might be added to the basic output when the **debug ppp bap negotiation** command is used:

```
Router# debug ppp bap negotiation

BAP laudrup: adding link speed 64 kbps for type 0x1 len 5
BAP laudrup: adding reason "User initiated addition", len 25
BAP laudrup: CallRsp, id 4, ACK
BAP laudrup: link speed 64 kbps for types 0x1, len 5 (ACK)
BAP laudrup: phone number "1: 0 2: ", len 7 (ACK)
BAP laudrup: adding call status 0, action 0 len 4
BAP laudrup: adding 1 phone numbers "1: 0 2: " len 7
BAP laudrup: adding reason "Successfully added link", len 25
BAP laudrup: StatusRsp, id 4, ACK
```

Additional negotiation messages might also be displayed for the following:

```
Received BAP message
Sending message
Decode individual options for send/receive
Notification of invalid options
```

The following shows additional reasons for a particular BAP action that might be displayed in an “adding reason” line of the **debug ppp bap negotiation** command output:

```
"Outgoing add request has precedence"
"Outgoing remove request has precedence"
"Unable to change request precedence"
"Unable to determine valid phone delta"
"Attempting to add link"
"Link addition is pending"
"Attempting to remove link"
"Link removal is pending"
"Precedence of peer marked CallReq for no action"
"Callback request rejected due to configuration"
"Call request rejected due to configuration"
"No links of specified type(s) available"
"Drop request disallowed due to configuration"
"Discriminator is invalid"
"No response to call requests"
"Successfully added link"
"Attempt to dial destination failed"
"No interfaces present to dial out"
"No dial string present to dial out"
"Mandatory options incomplete"
"Load has not exceeded threshold"
"Load is above threshold"
"Currently attempting to dial destination"
"No response to CallReq from race condition"
```

Table 203 describes the reasons for a BACP Negotiation Action.

**Table 203 Explanation of Reasons for BACP Negotiation Action**

Reason	Explanation
“Outgoing add request has precedence”	Received a CallRequest or CallbackRequest while we were waiting on a CallResponse or CallbackResponse to a sent request. We are the favored peer from the initial BACP negotiation, so we are issuing a NAK to our peer request.
“Outgoing remove request has precedence”	Received a LinkDropQueryRequest while waiting on a LinkDropQueryResponse to a sent request. We are the favored peer from the initial BACP negotiation, therefore we are issuing a NAK to our peer request.
“Unable to change request precedence”	Received a CallRequest, CallbackRequest, or LinkDropQueryRequest while waiting on a LinkDropQueryResponse to a sent request. Our peer is deemed to be the favored peer from the initial BACP negotiation and we were unable to change the status of our outgoing request in response to the favored request, so we are issuing a NAK. (This is an internal error and should never be seen.)
“Unable to determine valid phone delta”	Received a CallRequest from our peer but are unable to provide the required phone delta for the response, so we are issuing a NAK. (This is an internal error and should never be seen.)
“Attempting to add link”	Received a LinkDropQueryRequest while attempting to add a link; a NAK is issued.
“Link addition is pending”	Received a LinkDropQueryRequest, CallRequest, or CallbackRequest while attempting to add a link as the result of a previous operation; a NAK is issued in the response.
“Attempting to remove link”	Received a CallRequest or CallbackRequest while attempting to remove a link; a NAK is issued.
“Link removal is pending”	Received a CallRequest, CallbackRequest, or LinkDropQueryRequest while attempting to remove a link as the result of a previous operation; a NAK is issued in the response.
“Precedence of peer marked CallReq for no action”	Received an ACK to a previously unfavored CallRequest; we are issuing a CallStatusIndication to inform our peer that there will be no further action on our part as per this response.
“Callback request rejected due to configuration”	Received a CallbackRequest but we are configured not to accept them; a REJECT is issued to our peer.
“Call request rejected due to configuration”	Received a CallRequest but we are configured not to accept them; a REJECT is issued to our peer.
“No links of specified type(s) available”	We received a CallRequest but no links of the specified type and speed are available; a NAK is issued.
“Drop request disallowed due to configuration”	Received a LinkDropQueryRequest but we are configured not to accept them; a NAK is issued to our peer.

**Table 203 Explanation of Reasons for BACP Negotiation Action (continued)**

Reason	Explanation
“Discriminator is invalid”	Received a LinkDropQueryRequest but the local link discriminator is not contained within the bundle; a NAK is issued.
“No response to call requests”	After no response to our CallRequest message, a CallStatusIndication is sent to the peer informing that no more action will be taken on behalf of this operation.
“Successfully added link”	Sent as part of the CallStatusIndication informing our peer that we successfully completed the addition of a link to the bundle as the result of the transmission of a CallRequest or the reception of a CallbackRequest.
“Attempt to dial destination failed”	Sent as part of the CallStatusIndication informing our peer that we failed in an attempt to add a link to the bundle as the result of the transmission of a CallRequest or the reception of a CallbackRequest. The retry field with the CallStatusIndication informs the peer of our intentions.
“No interfaces present to dial out”	There are no available interfaces to dial out on to attempt to add a link to the bundle, and we will not retry the dial attempt.
“No dial string present to dial out”	We do not have a dial string to dial out with to attempt to add a link to the bundle, and we are not going to retry the dial attempt. (This is an internal error and should never be seen.)
“Mandatory options incomplete”	Received a CallRequest, CallbackRequest, LinkDropQueryRequest, or CallStatusIndication and the mandatory options are not present, so a NAK is issued in the response. (A CallStatusResponse is an ACK, however).
“Load has not exceeded threshold”	Received a CallRequest or CallbackRequest but we are issuing a NAK in the response. We are monitoring the load of the bundle, and so we determine when links should be added to the bundle.
“Load is above threshold”	Received a LinkDropQueryRequest but we are issuing a NAK in the response. We are monitoring the load of the bundle, and so we determine when links should be removed from the bundle.
“Currently attempting to dial destination”	Received a CallbackRequest which is a retransmission of one that we previously ACK'd and are dialing the number suggested in the request. We are issuing an ACK because we did so previously, even though our peer never saw the previous response.
“No response to CallReq from race condition”	We issued a CallRequest but failed to receive a response, and we are issuing a CallStatusIndication to inform our peer of our intention not to proceed with the operation.

# debug ppp multilink events

To display information about events affecting multilink groups established for BACP, use the **debug ppp multilink events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug ppp multilink events**

**no debug ppp multilink events**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## Usage Guidelines



### Caution

Do not use this command when memory is scarce or in very high traffic situations.

## Examples

The following is sample output from the **debug ppp multilink events** command:

```
Router# debug ppp multilink events
```

```
MLP laudrup: established BAP group 4 on Virtual-Access1, physical BRI3:1
MLP laudrup: removed BAP group 4
```

Other event messages include the following:

```
Unable to find bundle for BAP group identifier
Unable to find physical interface to start BAP
Unable to create BAP group
Attempt to start BACP when inactive or running
Attempt to start BACP on non-MLP interface
Link protocol has gone down, removing BAP group
Link protocol has gone down, BAP not running or present
```

[Table 204](#) describes the significant fields shown in the display.

**Table 204** *debug ppp multilink events Field Descriptions*

Field	Description
MLP laudrup	Name of the multilink group.
established BAP group 4	Internal identifier. The same identifiers are used in the <b>show ppp bap group</b> command output.
Virtual-Access1	Dynamic access interface number.
physical BRI3:1	Bundle was established from a call on this interface.
removed BAP group 4	When the bundle is removed, the associated BACP group (with its ID) is also removed.

# debug ppp multilink fragments

To display information about individual multilink fragments and important multilink events, use the **debug ppp multilink fragments** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug ppp multilink fragments**

**no debug ppp multilink fragments**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

## Usage Guidelines



### Caution

The **debug ppp multilink fragments** command has some memory overhead and should not be used when memory is scarce or in very high traffic situations.

## Examples

The following is sample output from the **debug ppp multilink fragments** command when used with the **ping** EXEC command. The debug output indicates that a multilink PPP packet on interface BRI 0 (on the B channel) is an input (I) or output (O) packet. The output also identifies the sequence number of the packet and the size of the fragment.

```
Router# debug ppp multilink fragments

Router# ping 7.1.1.7
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 7.1.1.7, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/34/36 ms
Router#
2:00:28: MLP BRI0: B-Channel 1: O seq 80000000: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000001: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000001: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000000: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000002: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000003: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000003: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000002: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000004: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000005: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000005: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000004: size 58
2:00:28: MLP BRI0: B-Channel 1: O seq 80000006: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000007: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000007: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000006: size 58
```

```
2:00:28: MLP BRI0: B-Channel 1: O seq 80000008: size 58
2:00:28: MLP BRI0: B-Channel 2: O seq 40000009: size 59
2:00:28: MLP BRI0: B-Channel 2: I seq 40000009: size 59
2:00:28: MLP BRI0: B-Channel 1: I seq 80000008: size 58
```

# debug pppatm

To enable debug reports for PPP over ATM (PPPoA) events, errors, and states, either globally or conditionally, on an interface or virtual circuit (VC), use the **debug pppatm** command in privileged EXEC mode. To disable the reports, use the **no** form of this command.

```
debug pppatm {event | error | state} [interface atm interface-number [subinterface-number]] vc
  {[vpi/vci]vci | virtual-circuit-name}
```

```
no debug pppatm {event | error | state} [interface atm interface-number [subinterface-number]]
  vc {[vpi/vci] | virtual-circuit-name}
```

## Syntax Description

<b>event</b>	PPPoA events.
<b>error</b>	PPPoA errors.
<b>state</b>	PPPoA state.
<b>interface atm</b> <i>interface-number</i> [ <i>subinterface-number</i> ]	(Optional) Specifies a particular ATM interface by interface number and optionally a subinterface number separated by a period.
<b>vc</b> [ <i>vpi</i> / <i>vci</i> ] <i>virtual-circuit-name</i>	(Optional) Virtual circuit (VC) keyword followed by a virtual path identifier (VPI), virtual channel identifier (VCI), and VC name. A slash mark is required after the VPI.

## Defaults

No default behavior or values.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.2(13)T	This command was introduced.

## Usage Guidelines

Each specific PPPoA debug report must be requested on a separate command line; see the “Examples” section.

## Examples

The following is example output of a PPPoA session with event, error, and state debug reports enabled on ATM interface 1/0.10:

```
Router# debug pppatm event interface atm1/0.10
Router# debug pppatm error interface atm1/0.10
Router# debug pppatm state interface atm1/0.10

00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = Clear Session
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = Disconnecting
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = AAA gets dynamic attrs
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = AAA gets dynamic attrs
00:03:08: PPPATM: ATM1/0.10 0/101 [1], Event = SSS Cleanup
```

```

00:03:08: PPPATM: ATM1/0.10 0/101 [0], State = DOWN
00:03:08: PPPATM: ATM1/0.10 0/101 [0], Event = Up Pending
00:03:16: PPPATM: ATM1/0.10 0/101 [0], Event = Up Dequeued
00:03:16: PPPATM: ATM1/0.10 0/101 [0], Event = Processing Up
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = Access IE allocated
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = Set Pkts to SSS
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets retrived attrs
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets nas port details
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = AAA unique id allocated
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = No AAA method list set
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = SSS Request
00:03:16: PPPATM: ATM1/0.10 0/101 [2], State = NAS_PORT_POLICY_INQUIRY
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = SSS Msg Received = 1
00:03:16: PPPATM: ATM1/0.10 0/101 [2], State = PPP_START
00:03:16: PPPATM: ATM1/0.10 0/101 [2], Event = PPP Msg Received = 1
00:03:16: PPPATM: ATM1/0.10 0/101 [2], State = LCP_NEGOTIATION
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = PPP Msg Received = 4
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = HW Switch support FORW = 0
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = Access IE get nas port
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = AAA gets dynamic attrs
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = PPP Msg Received = 5
00:03:27: PPPATM: ATM1/0.10 0/101 [2], Event = Set Pkts to SSS
00:03:27: PPPATM: ATM1/0.10 0/101 [2], State = FORWARDED

```

Table 205 describes the significant fields shown in the display.

**Table 205** *debug pppatm* Field Descriptions

Field	Description
Event	Reports PPPoA events for use by Cisco engineering technical assistance personnel.
State	Reports PPPoA states for use by Cisco engineering technical assistance personnel.

#### Related Commands

Command	Description
<b>atm pppatm passive</b>	Places an ATM subinterface into passive mode.
<b>show pppatm summary</b>	Displays PPPoA session counts.

# debug pppoe

To display debugging information for PPP over Ethernet (PPPoE) sessions, use the **debug pppoe** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug pppoe { data | errors | events | packets } [rmac remote-mac-address | interface type number
[vc {[vpi]/vci | vc-name}]]
```

```
no debug pppoe { data | errors | events | packets } [rmac remote-mac-address |
interface type number [vc {[vpi]/vci | vc-name}]]
```

## Syntax Description

<b>data</b>	Displays data packets of PPPoE sessions.
<b>errors</b>	Displays PPPoE protocol errors that prevent a session from being established, or displays errors that cause an established session to be closed.
<b>events</b>	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
<b>packets</b>	Displays each PPPoE protocol packet that is exchanged.
<b>rmac</b> <i>remote-mac-address</i>	(Optional) Remote MAC address. Debugging information for PPPoE sessions sourced from this address will be displayed.
<b>interface</b> <i>type number</i>	(Optional) Interface for which PPPoE session debugging information will be displayed.
<b>vc</b>	(Optional) Displays debugging information for PPPoE sessions for a specific permanent virtual circuit (PVC).
<i>vpi</i>	(Optional) ATM network virtual path identifier (VPI) for the PVC. In the absence of the slash (/) and a <i>vpi</i> value, the <i>vpi</i> value defaults to 0.
<i>vci</i>	(Optional) ATM network virtual channel identifier (VCI) for the PVC.
<i>vc-name</i>	(Optional) Name of the PVC.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
12.2(13)T	This command was introduced. This command replaces the <b>debug vpdn pppoe-data</b> , <b>debug vpdn pppoe-error</b> , <b>debug vpdn pppoe-events</b> , and <b>debug vpdn pppoe-packet</b> commands available in previous Cisco IOS releases.
12.2(15)T	This command was modified to display debugging information on a per-MAC address, per-interface, and per-VC basis.

**Examples**

The following examples show sample output from the **debug pppoe** command:

```
Router# debug pppoe events interface atm1/0.10 vc 101
```

```
PPPoE protocol events debugging is on
Router#
00:41:55:PPPoE 0:I PADI R:00b0.c2e9.c470 L:ffff.ffff.ffff 0/101 ATM1/0.10
00:41:55:PPPoE 0:O PADO, R:00b0.c2e9.c470 L:0001.c9f0.0c1c 0/101 ATM1/0.10
00:41:55:PPPoE 0:I PADR R:00b0.c2e9.c470 L:0001.c9f0.0c1c 0/101 ATM1/0.10
00:41:55:PPPoE :encap string prepared
00:41:55:[3]PPPoE 3:Access IE handle allocated
00:41:55:[3]PPPoE 3:pppoe SSS switch updated
00:41:55:[3]PPPoE 3:AAA unique ID allocated
00:41:55:[3]PPPoE 3:No AAA accounting method list
00:41:55:[3]PPPoE 3:Service request sent to SSS
00:41:55:[3]PPPoE 3:Created R:0001.c9f0.0c1c L:00b0.c2e9.c470 0/101 ATM1/0.10
00:41:55:[3]PPPoE 3:State REQ_NASPORT Event MORE_KEYS
00:41:55:[3]PPPoE 3:O PADS R:00b0.c2e9.c470 L:0001.c9f0.0c1c 0/101 ATM1/0.10
00:41:55:[3]PPPoE 3:State START_PPP Event DYN_BIND
00:41:55:[3]PPPoE 3:data path set to PPP
00:41:57:[3]PPPoE 3:State LCP_NEGO Event PPP_LOCAL
00:41:57:PPPoE 3/SB:Sent vtemplate request on base Vi2
00:41:57:[3]PPPoE 3:State CREATE_VA Event VA_RESP
00:41:57:[3]PPPoE 3:Vi2.1 interface obtained
00:41:57:[3]PPPoE 3:State PTA_BIND Event STAT_BIND
00:41:57:[3]PPPoE 3:data path set to Virtual Access
00:41:57:[3]PPPoE 3:Connected PTA
```

```
Router# debug pppoe errors interface atm1/0.10
```

```
PPPoE protocol errors debugging is on
Router#
00:44:30:PPPoE 0:Max session count(1) on mac(00b0.c2e9.c470) reached.
00:44:30:PPPoE 0:Over limit or Resource low. R:00b0.c2e9.c470 L:ffff.ffff.ffff 0/101
ATM1/0.10
```

[Table 206](#) describes significant fields shown in the displays.

**Table 206** debug pppoe Field Descriptions

Field	Description
PPPoE	PPPoE debug message header.
0:	PPPoE session ID.
I PADI	Incoming PPPoE Active Discovery Initiation packet.
R:	Remote MAC address.
L:	Local MAC address.
0/101	Virtual path identifier (VPI)/virtual channel identifier (VCI) of the PVC.
ATM1/0.10	Interface type and number.
O PADO	Outgoing PPPoE Active Discovery Offer packet.
I PADR	Incoming PPPoE Active Discovery Request packet.

Table 206 debug pppoe Field Descriptions (continued)

Field	Description
[3]	Unique user session ID. The same ID is used for identifying sessions across different applications such as PPPoE, PPP, Layer 2 Tunneling Protocol (L2TP), and Subscriber Service Switch (SSS). The same session ID appears in the output for the <b>show pppoe session</b> , <b>show sss session</b> , and <b>show vpdn session</b> commands.
PPPoE 3	PPPoE session ID.
Created	PPPoE session is created.
O PADS	Outgoing PPPoE Active Discovery Session-confirmation packet.
Connected PTA	PPPoE session is established.
Max session count(1) on mac(00b0.c2e9.c470) reached	PPPoE session is rejected because of per-MAC session limit.

## Related Commands

Command	Description
<b>encapsulation aal5autopp virtual-template</b>	Enables PPPoA/PPPoE autosense.
<b>pppoe enable</b>	Enables PPPoE sessions on an Ethernet interface or subinterface.
<b>protocol pppoe (ATM VC)</b>	Enables PPPoE sessions to be established on PVCs.
<b>show pppoe session</b>	Displays information about active PPPoE sessions.
<b>show sss session</b>	Displays Subscriber Service Switch session status.
<b>show vpdn session</b>	Displays session information about L2TP, L2F protocol, and PPPoE tunnels in a VPDN.

# debug priority

To display priority queueing output, use the **debug priority** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug priority**

**no debug priority**

---

**Syntax Description**

This command has no arguments or keywords.

---

**Command Modes**

Privileged EXEC

---

**Examples**

The following example shows how to enable priority queueing output:

```
Router# debug priority
```

```
Priority output queueing debugging is on
```

The following is sample output from the **debug priority** command when the Frame Relay PVC Interface Priority Queueing (FR PIPQ) feature is configured on serial interface 0:

```
Router# debug priority
```

```
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 100 -> high
00:49:05:PQ:Serial0 output (Pk size/Q 24/0)
00:49:05:PQ:Serial0 dlci 200 -> medium
00:49:05:PQ:Serial0 output (Pk size/Q 24/1)
00:49:05:PQ:Serial0 dlci 300 -> normal
00:49:05:PQ:Serial0 output (Pk size/Q 24/2)
00:49:05:PQ:Serial0 dlci 400 -> low
00:49:05:PQ:Serial0 output (Pk size/Q 24/3)
```

---

**Related Commands**

Command	Description
<b>debug custom-queue</b>	Displays custom queueing output.

# debug proxy h323 statistics

To enable proxy RTP statistics, use the **debug proxy h323 statistics** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.**debug proxy h323 statistics**

**no debug proxy h323 statistics**

---

**Syntax Description** This command has no arguments or keywords.

---

**Command Modes** Privileged EXEC

---

<b>Command History</b>	<b>Release</b>	<b>Modification</b>
	11.3(2)NA	This command was introduced.

---

---

**Usage Guidelines** Enter the **show proxy h323 detail-call** EXEC command to see the statistics.

# debug pvcd

To display the PVC Discovery events and ILMI MIB traffic used when discovering PVCs, use the **debug pvcd** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug pvcd**

**no debug pvcd**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## Usage Guidelines

This command is primarily used by Cisco technical support representatives.

## Examples

The following is sample output from the **debug pvcd** command:

```
Router# debug pvcd
```

```
PVCD: PVCD enabled w/ Subif
PVCD(2/0): clearing event queue
PVCD: 2/0 Forgetting discovered PVCs...
PVCD: Removing all dynamic PVCs on 2/0
PVCD: Restoring MIXED PVCs w/ default parms on 2/0
PVCD: Marking static PVCs as UNKNWN on 2/0
PVCD: Marking static PVC 0/50 as UNKNWN on 2/0 ...
PVCD: Trying to discover PVCs on 2/0...
PVCD: pvcd_discoverPVCs
PVCD: pvcd_ping
PVCD: fPortEntry.5.0 = 2
PVCD: pvcd_getPeerVccTableSize
PVCD: fLayerEntry.5.0 = 13
PVCD:end allocating VccTable size 13
PVCD: pvcd_getPeerVccTable
PVCD:***** 2/0: getNext on fVccEntry = NULL TYPE/VALUE numFiledS = 19 numVccs = 13
PVCD: Creating Dynamic PVC 0/33 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/33: DYNAMIC
PVCD: After _create_pvc() VC 0/33: DYNAMIC0/33 on 2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/34 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/34: DYNAMIC
PVCD: After _create_pvc() VC 0/34: DYNAMIC0/34 on 2/0 : UBR PCR -1
PVCD: Creating Dynamic PVC 0/44 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/44: DYNAMIC
PVCD: After _create_pvc() VC 0/44: DYNAMIC0/44 on 2/0 : UBR PCR = -1
PVCD: PVC 0/50 with INHERITED_QOSTYPE
PVCD: _oi_state_change ( 0/50, 1 = ILMI_VC_UP )
PVCD: Creating Dynamic PVC 0/60 on 2/0
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/60: DYNAMIC
PVCD: After _create_pvc() VC 0/60: DYNAMIC0/60 on 2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/80 on 2/0
```

```
PVCD(2/0): Before _update_inheritance() and _create_pvc() VC 0/80: DYNAMIC
PVCD: After _create_pvc() VC 0/80: DYNAMIC0/80 on 2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/99 on 2/0
```

# debug qlc error

To display quality link line control (QLLC) errors, use the **debug qlc error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug qlc error**

**no debug qlc error**

---

## Syntax Description

This command has no arguments or keywords.

---

## Command Modes

Privileged EXEC

---

## Usage Guidelines

This command helps you track down errors in the QLLC interactions with X.25 networks. Use the **debug qlc error** command in conjunction with the **debug x25 all** command to see the connection. The data shown by this command only flows through the router on the X.25 connection. Some forms of this command can generate a substantial amount of output and network traffic.

---

## Examples

The following is sample output from the **debug qlc error** command:

```
Router# debug qlc error
```

```
%QLLC-3-GENERRMSG: qlc_close - bad qlc pointer Caller 00407116 Caller 00400BD2  
QLLC 4000.1111.0002: NO X.25 connection. Discarding XID and calling out
```

The following line indicates that the QLLC connection was closed:

```
%QLLC-3-GENERRMSG: qlc_close - bad qlc pointer Caller 00407116 Caller 00400BD2
```

The following line shows the virtual MAC address of the failed connection:

```
QLLC 4000.1111.0002: NO X.25 connection. Discarding XID and calling out
```

# debug qlc event

To enable debugging of QLLC events, use the **debug qlc event** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug qlc event**

**no debug qlc event**

---

**Syntax Description**

This command has no arguments or keywords.

---

**Command Modes**

Privileged EXEC

---

**Usage Guidelines**

Use the **debug qlc event** command to display primitives that might affect the state of a QLLC connection. An example of these events is the allocation of a QLLC structure for a logical channel indicator when an X.25 call has been accepted with the QLLC call user data. Other examples are the receipt and transmission of LAN explorer and XID frames.

---

**Examples**

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

```
Router# debug qlc event

QLLC: allocating new qlc lci 9
QLLC: tx POLLING TEST, da 4001.3745.1088, sa 4000.1111.0001
QLLC: rx explorer response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
QLLC: gen NULL XID, da c001.3745.1088, sa 4000.1111.0001, rif 0830.1A91.1901.A040, dsap 4,
ssap 4
QLLC: rx XID response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
```

The following line indicates that a new QLLC data structure has been allocated:

```
QLLC: allocating new qlc lci 9
```

The following lines show transmission and receipt of LAN explorer or test frames:

```
QLLC: tx POLLING TEST, da 4001.3745.1088, sa 4000.1111.0001
QLLC: rx explorer response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
```

The following lines show XID events:

```
QLLC: gen NULL XID, da c001.3745.1088, sa 4000.1111.0001, rif 0830.1A91.1901.A040, dsap 4,
ssap 4
QLLC: rx XID response, da 4000.1111.0001, sa c001.3745.1088, rif 08B0.1A91.1901.A040
```

# debug qlc packet

To display QLLC events and QLLC data packets, use the **debug qlc packet** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug qlc packet**

**no debug qlc packet**

## Syntax Description

This command has no arguments or keywords.

## Command Modes

Privileged EXEC

## Usage Guidelines

This command helps you to track down errors in the QLLC interactions with X.25 networks. The data shown by this command only flows through the router on the X25 connection. Use the **debug qlc packet** command in conjunction with the **debug x25 all** command to see the connection and the data that flows through the router.

## Examples

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

```
Router# debug qlc packet

14:38:05: Serial2/5 QLLC I: Data Packet.-RSP 9 bytes.
14:38:07: Serial2/6 QLLC I: Data Packet.-RSP 112 bytes.
14:38:07: Serial2/6 QLLC O: Data Packet. 128 bytes.
14:38:08: Serial2/6 QLLC I: Data Packet.-RSP 9 bytes.
14:38:08: Serial2/6 QLLC I: Data Packet.-RSP 112 bytes.
14:38:08: Serial2/6 QLLC O: Data Packet. 128 bytes.
14:38:08: Serial2/6 QLLC I: Data Packet.-RSP 9 bytes.
14:38:12: Serial2/5 QLLC I: Data Packet.-RSP 112 bytes.
14:38:12: Serial2/5 QLLC O: Data Packet. 128 bytes.
```

The following lines indicate that a packet was received on the interfaces:

```
14:38:05: Serial2/5 QLLC I: Data Packet.-RSP 9 bytes.
14:38:07: Serial2/6 QLLC I: Data Packet.-RSP 112 bytes.
```

The following lines show that a packet was sent on the interfaces:

```
14:38:07: Serial2/6 QLLC O: Data Packet. 128 bytes.
14:38:12: Serial2/5 QLLC O: Data Packet. 128 bytes.
```

# debug qlc state

To enable debugging of QLLC events, use the **debug qlc state** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug qlc state**

**no debug qlc state**

---

**Syntax Description** This command has no arguments or keywords.

---

**Command Modes** Privileged EXEC

---

**Usage Guidelines** Use the **debug qlc state** command to show when the state of a QLLC connection has changed. The typical QLLC connection goes from states ADM to SETUP to NORMAL. The NORMAL state indicates that a QLLC connection exists and is ready for data transfer.

---

**Examples** The following is sample output from the **debug qlc state** command:

```
Router# debug qlc state

Serial2 QLLC O: QSM-CMD
Serial2: X25 O D1 DATA (5) Q 8 lci 9 PS 4 PR 3
QLLC: state ADM -> SETUP
Serial2: X25 I D1 RR (3) 8 lci 9 PR 5
Serial2: X25 I D1 DATA (5) Q 8 lci 9 PS 3 PR 5
Serial2 QLLC I: QUA-RSPQLLC: addr 00, ctl 73

QLLC: qsetupstate: recvd qua rsp
QLLC: state SETUP -> NORMAL
```

The following line indicates that a QLLC connection attempt is changing state from ADM to SETUP:

```
QLLC: state ADM -> SETUP
```

The following line indicates that a QLLC connection attempt is changing state from SETUP to NORMAL:

```
QLLC: state SETUP -> NORMAL
```

# debug qlc timer

To display QLLC timer events, use the **debug qlc timer** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug qlc timer**

**no debug qlc timer**

---

## Syntax Description

This command has no arguments or keywords.

---

## Command Modes

Privileged EXEC

---

## Usage Guidelines

The QLLC process periodically cycles and checks status of itself and its partner. If the partner is not found in the desired state, a LAPB primitive command is re-sent until the partner is in the desired state or the timer expires.

---

## Examples

The following is sample output from the **debug qlc timer** command:

```
Router# debug qlc timer
```

```
14:27:24: Qllc timer lci 257, state ADM retry count 0 Caller 00407116 Caller 00400BD2  
14:27:34: Qllc timer lci 257, state NORMAL retry count 0  
14:27:44: Qllc timer lci 257, state NORMAL retry count 1  
14:27:54: Qllc timer lci 257, state NORMAL retry count 1
```

The following line of output shows the state of a QLLC partner on a given X.25 logical channel identifier:

```
14:27:24: Qllc timer lci 257, state ADM retry count 0 Caller 00407116 Caller 00400BD2
```

Other messages are informational and appear every ten seconds.

## debug qlc x25

To display X.25 packets that affect a QLLC connection, use the **debug qlc x25** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug qlc x25**

**no debug qlc x25**

**Syntax Description** This command has no arguments or keywords.

**Command Modes** Privileged EXEC

**Usage Guidelines** This command is helpful to track down errors in the QLLC interactions with X.25 networks. Use the **debug qlc x25** command in conjunction with the **debug x25 events** or **debug x25 all** commands to see the X.25 events between the router and its partner.

**Examples** The following is sample output from the **debug qlc x25** command:

```
Router# debug qlc x25

15:07:23: QLLC X25 notify lci 257 event 1
15:07:23: QLLC X25 notify lci 257 event 5
15:07:34: QLLC X25 notify lci 257 event 3 Caller 00407116 Caller 00400BD2
15:07:35: QLLC X25 notify lci 257 event 4
```

[Table 207](#) describes fields of output.

**Table 207** *debug qlc x.25 Field Descriptions*

Field	Description
15:07:23	Displays the time of day.
QLLC X25 notify 257	Indicates that this is a QLLC X25 message.
event <n>	Indicates the type of event, <i>n</i> . Values for <i>n</i> can be as follows: <ul style="list-style-type: none"> <li>1—Circuit is cleared</li> <li>2—Circuit has been reset</li> <li>3—Circuit is connected</li> <li>4—Circuit congestion has cleared</li> <li>5—Circuit has been deleted</li> </ul>

# debug radius

To display information associated with RADIUS, use the **debug radius** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug radius [brief | hex]**

**no debug radius [brief | hex]**

## Syntax Description

<b>brief</b>	(Optional) Displays abbreviated debug output.
<b>hex</b>	(Optional) Displays debug output in hexadecimal notation.

## Defaults

Debugging output in ASCII format is enabled by default.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
11.2(1)T	This command was introduced.
12.2(11)T	The <b>brief</b> and <b>hex</b> keywords were added. The default output format became ASCII rather than hexadecimal.

## Usage Guidelines

RADIUS is a distributed security system that secures networks against unauthorized access. Cisco supports RADIUS under the authentication, authorization, and accounting (AAA) security system. When RADIUS is used on the router, you can use the **debug radius** command to display detailed debugging and troubleshooting information in ASCII format. Use the **debug radius brief** command for abbreviated output displaying client/server interaction and minimum packet information. Use the **debug radius hex** command to display packet dump information that has not been truncated in hex format.

## Examples

The following is sample output from the **debug radius** command:

```
Router# debug radius

Radius protocol debugging is on
Radius packet hex dump debugging is off
Router#
00:02:50: RADIUS: ustruct sharecount=3
00:02:50: RADIUS: radius_port_info() success=0 radius_nas_port=1
00:02:50: RADIUS: Initial Transmit ISDN 0:D:23 id 0 10.0.0.1:1824, Accounting-Request, len
358
00:02:50: RADIUS:  NAS-IP-Address      [4]  6  10.0.0.0
00:02:50: RADIUS:  Vendor, Cisco        [26] 19  VT=02 TL=13 ISDN 0:D:23
00:02:50: RADIUS:  NAS-Port-Type       [61] 6  Async
00:02:50: RADIUS:  User-Name           [1] 12  "4085554206"
00:02:50: RADIUS:  Called-Station-Id  [30] 7  "52981"
00:02:50: RADIUS:  Calling-Station-Id [31] 12 "4085554206"
00:02:50: RADIUS:  Acct-Status-Type   [40] 6  Start
```

```

00:02:50: RADIUS: Service-Type          [6] 6 Login
00:02:50: RADIUS: Vendor, Cisco         [26] 27 VT=33 TL=21 h323-gw-id=5300_43.
00:02:50: RADIUS: Vendor, Cisco         [26] 55 VT=01 TL=49
h323-incoming-conf-id=8F3A3163 B4980003 0 29BD0
00:02:50: RADIUS: Vendor, Cisco         [26] 31 VT=26 TL=25 h323-call-origin=answer
00:02:50: RADIUS: Vendor, Cisco         [26] 32 VT=27 TL=26 h323-call-type=Telephony
00:02:50: RADIUS: Vendor, Cisco         [26] 57 VT=25 TL=51 h323-setup-time=*16:02:48.681
PST Fri Dec 31 1999
00:02:50: RADIUS: Vendor, Cisco         [26] 46 VT=24 TL=40 h323-conf-id=8F3A3163
B4980003 0 29BD0
00:02:50: RADIUS: Acct-Session-Id       [44] 10 "00000002"
00:02:50: RADIUS: Delay-Time            [41] 6 0
00:02:51: RADIUS: Received from id 0 1.7.157.1:1824, Accounting-response, len 20
00:02:51: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4085274206
00:03:01: RADIUS: ustruct sharecount=3
00:03:01: Radius: radius_port_info() success=0 radius_nas_port=1
00:03:01: RADIUS: Initial Transmit ISDN 0:D:23 id 1 1.7.157.1:1823, Access-Request, len
171
00:03:01: RADIUS: NAS-IP-Address        [4] 6 10.0.0.0
00:03:01: RADIUS: Vendor, Cisco         [26] 19 VT=02 TL=13 ISDN 0:D:23
00:03:01: RADIUS: NAS-Port-Type         [61] 6 Async
00:03:01: RADIUS: User-Name             [1] 8 "123456"
00:03:01: RADIUS: Vendor, Cisco         [26] 46 VT=24 TL=40 h323-conf-id=8F3A3163
B4980003 0 29BD0
00:03:01: RADIUS: Calling-Station-Id   [31] 12 "4085554206"
00:03:01: RADIUS: User-Password         [2] 18 *
00:03:01: RADIUS: Vendor, Cisco         [26] 36 VT=01 TL=30 h323-ivr-out=transactionID:0
00:03:01: RADIUS: Received from id 1 1.7.157.1:1823, Access-Accept, len 115
00:03:01: RADIUS: Service-Type          [6] 6 Login
00:03:01: RADIUS: Vendor, Cisco         [26] 29 VT=101 TL=23 h323-credit-amount=45
00:03:01: RADIUS: Vendor, Cisco         [26] 27 VT=102 TL=21 h323-credit-time=33
00:03:01: RADIUS: Vendor, Cisco         [26] 26 VT=103 TL=20 h323-return-code=0
00:03:01: RADIUS: Class                 [25] 7 6C6F63616C
00:03:01: RADIUS: saved authorization data for user 62321E14 at 6233D258
00:03:13: %ISDN-6-DISCONNECT: Interface Serial0:22 disconnected from 4085274206, call
lasted 22 seconds
00:03:13: RADIUS: ustruct sharecount=2
00:03:13: Radius: radius_port_info() success=0 radius_nas_port=1
00:03:13: RADIUS: Sent class "local" at 6233D2C4 from user 62321E14
00:03:13: RADIUS: Initial Transmit ISDN 0:D:23 id 2 1.7.157.1:1824, Accounting-Request,
len 775
00:03:13: RADIUS: NAS-IP-Address        [4] 6 10.0.0.0
00:03:13: RADIUS: Vendor, Cisco         [26] 19 VT=02 TL=13 ISDN 0:D:23
00:03:13: RADIUS: NAS-Port-Type         [61] 6 Async
00:03:13: RADIUS: User-Name             [1] 8 "123456"
00:03:13: RADIUS: Called-Station-Id     [30] 7 "52981"
00:03:13: RADIUS: Calling-Station-Id   [31] 12 "4085274206"
00:03:13: RADIUS: Acct-Status-Type     [40] 6 Stop
00:03:13: RADIUS: Class                 [25] 7 6C6F63616C
00:03:13: RADIUS: Undebuggable         [45] 6 00000001
00:03:13: RADIUS: Service-Type          [6] 6 Login
00:03:13: RADIUS: Vendor, Cisco         [26] 27 VT=33 TL=21 h323-gw-id=5300_43.
00:03:13: RADIUS: Vendor, Cisco         [26] 55 VT=01 TL=49
h323-incoming-conf-id=8F3A3163 B4980003 0 29BD0
00:03:13: RADIUS: Vendor, Cisco         [26] 31 VT=26 TL=25 h323-call-origin=answer
00:03:13: RADIUS: Vendor, Cisco         [26] 32 VT=27 TL=26 h323-call-type=Telephony
00:03:13: RADIUS: Vendor, Cisco         [26] 57 VT=25 TL=51 h323-setup-time=*16:02:48.681
PST Fri Dec 31 1999
00:03:13: RADIUS: Vendor, Cisco         [26] 59 VT=28 TL=53
h323-connect-time=*16:02:48.946 PST Fri Dec 31 1999
00:03:13: RADIUS: Vendor, Cisco         [26] 62 VT=29 TL=56in=0
00:03:13: RADIUS: Vendor, Cisco         [26] 23 VT=01 TL=17 pre-bytes-out=0
00:03:13: RADIUS: Vendor, Cisco         [26] 21 VT=01 TL=15 pre-paks-in=0
00:03:13: RADIUS: Vendor, Cisco         [26] 22 VT=01 TL=16 pre-paks-out=0

```

```

00:03:13: RADIUS: Vendor, Cisco [26] 22 VT=01 TL=16 nas-rx-speed=0
00:03:13: RADIUS: Vendor, Cisco [26] 22 VT=01 TL=16 nas-tx-speed=0
00:03:13: RADIUS: Delay-Time [41] 6 0
00:03:13: RADIUS: Received from id 2 1.7.157.1:1824, Accounting-response, len 20
h323-disconnect-time=*16:03:11.306 PST Fri Dec 31 1999
00:03:13: RADIUS: Vendor, Cisco [26] 32 VT=30 TL=26 h323-disconnect-cause=10
00:03:13: RADIUS: Vendor, Cisco [26] 28 VT=31 TL=22 h323-voice-quality=0
00:03:13: RADIUS: Vendor, Cisco [26] 46 VT=24 TL=40 h323-conf-id=8F3A3163
B4980003 0 29BD0
00:03:13: RADIUS: Acct-Session-Id [44] 10 "00000002"
00:03:13: RADIUS: Acct-Input-Octets [42] 6 0
00:03:13: RADIUS: Acct-Output-Octets [43] 6 88000
00:03:13: RADIUS: Acct-Input-Packets [47] 6 0
00:03:13: RADIUS: Acct-Output-Packets [48] 6 550
00:03:13: RADIUS: Acct-Session-Time [46] 6 22
00:03:13: RADIUS: Vendor, Cisco [26] 30 VT=01 TL=24 subscriber=RegularLine
00:03:13: RADIUS: Vendor, Cisco [26] 35 VT=01 TL=29 h323-ivr-out=Tariff:Unknown
00:03:13: RADIUS: Vendor, Cisco [26] 22 VT=01 TL=16 pre-bytes-

```

The following is sample output from the **debug radius brief** command:

```
Router# debug radius brief
```

```

Radius protocol debugging is on
Radius packet hex dump debugging is off
Radius protocol in brief format debugging is on
00:05:21: RADIUS: Initial Transmit ISDN 0:D:23 id 6 10.0.0.1:1824, Accounting-Request, len
358
00:05:21: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4085274206
00:05:26: RADIUS: Retransmit id 6
00:05:31: RADIUS: Tried all servers.
00:05:31: RADIUS: No valid server found. Trying any viable server
00:05:31: RADIUS: Tried all servers.
00:05:31: RADIUS: No response for id 7
00:05:31: RADIUS: Initial Transmit ISDN 0:D:23 id 8 10.0.0.0:1823, Access-Request, len 171
00:05:36: RADIUS: Retransmit id 8
00:05:36: RADIUS: Received from id 8 1.7.157.1:1823, Access-Accept, len 115
00:05:47: %ISDN-6-DISCONNECT: Interface Serial0:22 disconnected from 4085274206, call
lasted 26 seconds
00:05:47: RADIUS: Initial Transmit ISDN 0:D:23 id 9 10.0.0.1:1824, Accounting-Request, len
775
00:05:47: RADIUS: Received from id 9 1.7.157.1:1824, Accounting-response, len 20

```

The following example shows **debug radius hex** output:

```
Router# debug radius hex
```

```

Radius protocol debugging is on
Radius packet hex dump debugging is on
Router#
17:26:52: RADIUS: ustruct sharecount=3
17:26:52: Radius: radius_port_info() success=0 radius_nas_port=1
17:26:52: RADIUS: Initial Transmit ISDN 0:D:23 id 10 10.0.0.1:1824, Accounting-Request,
len 361
17:26:52: Attribute 4 6 01081D03
17:26:52: Attribute 26 19 00000009020D4953444E20303A443A3233
17:26:52: Attribute 61 6 00000000
17:26:52: Attribute 1 12 34303835323734323036
17:26:52: Attribute 30 7 3532393831
17:26:52: Attribute 31 12 34303835323734323036
17:26:52: Attribute 40 6 00000001
17:26:52: Attribute 6 6 00000001
17:26:52: Attribute 26 27 000000092115683332332D67772D69643D353330305F34332E

```

```

17:26:52:      Attribute 26 57
000000090133683332332D696E636F6D696E672D636F6E662D69643D3846334133313633204234393830303046
20302033424537314238
17:26:52:      Attribute 26 31
000000091A19683332332D63616C6C2D6F726967696E3D616E73776572
17:26:52:      Attribute 26 32
000000091B1A683332332D63616C6C2D747970653D54656C6570686F6E79
17:26:52:      Attribute 26 56
000000091932683332332D73657475702D74696D653D2A30393A32363A35322E3838302050535420536174204A
616E20312032303030
17:26:52:      Attribute 26 48
00000009182A683332332D636F6E662D69643D3846334133313633204234393830303046203020334245373142
38
17:26:52:      Attribute 44 10 3030303030303035
17:26:52:      Attribute 41 6 00000000
17:26:52: %ISDN-6-CONNECT: Interface Serial0:22 is now connected to 4085274206
17:26:52: RADIUS: Received from id 10 10.0.0.1:1824, Accounting-response, len 20
17:27:01: RADIUS: ustruct sharecount=3
17:27:01: Radius: radius_port_info() success=0 radius_nas_port=1
17:27:01: RADIUS: Initial Transmit ISDN 0:D:23 id 11 10.0.0.0:1823, Access-Request, len
173
17:27:01:      Attribute 4 6 01081D03
17:27:01:      Attribute 26 19 00000009020D4953444E20303A443A3233
17:27:01:      Attribute 61 6 00000000
17:27:01:      Attribute 1 8 313233343536
17:27:01:      Attribute 26 48
00000009182A683332332D636F6E662D69643D3846334133313633204234393830303046203020334245373142
38
17:27:01:      Attribute 31 12 34303835323734323036
17:27:01:      Attribute 2 18 C980D8D0E9A061B3D783C61AA6F27214
17:27:01:      Attribute 26 36
00000009011E683332332D6976722D6F75743D7472616E73616374696F6E49443A33
17:27:01: RADIUS: Received from id 11 1.7.157.1:1823, Access-Accept, len 115
17:27:01:      Attribute 6 6 00000001
17:27:01:      Attribute 26 29 000000096517683332332D6372656469742D616D6F756E743D3435
17:27:01:      Attribute 26 27 000000096615683332332D6372656469742D74696D653D3333
17:27:01:      Attribute 26 26 000000096714683332332D72657475726E2D636F64653D30
17:27:01:      Attribute 25 7 6C6F63616C
17:27:01: RADIUS: saved authorization data for user 61AA0698 at 6215087C
17:27:09: %ISDN-6-DISCONNECT: Interface Serial0:22 disconnected from 4085554206, call
lasted 17 seconds
17:27:09: RADIUS: ustruct sharecount=2
17:27:09: Radius: radius_port_info() success=0 radius_nas_port=1
17:27:09: RADIUS: Sent class "local" at 621508E8 from user 61AA0698
17:27:09: RADIUS: Initial Transmit ISDN 0:D:23 id 12 1.7.157.1:1824, Accounting-Request,
len 776
17:27:09:      Attribute 4 6 01081D03
17:27:09:      Attribute 26 19 00000009020D4953444E20303A443A3233
17:27:09:      Attribute 61 6 00000000
17:27:09:      Attribute 1 8 313233343536
17:27:09:      Attribute 30 7 3532393831
17:27:09:      Attribute 31 12 34303835323734323036
17:27:09:      Attribute 40 6 00000002
17:27:09:      Attribute 25 7 6C6F63616C
17:27:09:      Attribute 45 6 00000001
17:27:09:      Attribute 6 6 00000001
17:27:09:      Attribute 26 27 000000092115683332332D67772D69643D353330305F34332E
17:27:09:      Attribute 26 57
000000090133683332332D696E636F6D696E672D636F6E662D69643D3846334133313633204234393830303046
20302033424537314238
17:27:09:      Attribute 26 31
000000091A19683332332D63616C6C2D6F726967696E3D616E73776572
17:27:09:      Attribute 26 32
000000091B1A683332332D63616C6C2D747970653D54656C6570686F6E79

```

```

17:27:09:      Attribute 26 56
000000091932683332332D73657475702D74696D653D2A30393A32363A35322E3838302050535420536174204A
616E20312032303030
17:27:09:      Attribute 26 58
000000091C34683332332D636F6E6E6563742D74696D653D2A30393A32363A35322E3930372050535420536174
204A616E20312032303030
17:27:09:      Attribute 26 61
000000091D37683332332D646973636F6E6E6563742D74696D653D2A30393A32373A31302E3133372050535420
536174204A616E20312032303030
17:27:09:      Attribute 26 32
000000091E1A683332332D646973636F6E6E6563742D63617573653D3130
17:27:09:      Attribute 26 28 000000091F16683332332D766F6963652D7175616C6974793D30
17:27:09:      Attribute 26 48
00000009182A683332332D636F6E662D69643D3846334133313633204234393830303046203020334245373142
38
17:27:09:      Attribute 44 10 3030303030303035
17:27:09:      Attribute 42 6 00000000
17:27:09:      Attribute 43 6 00012CA0
17:27:09:      Attribute 47 6 00000000
17:27:09:      Attribute 48 6 000001E1
17:27:09:      Attribute 46 6 00000011
17:27:09:      Attribute 26 30 000000090118737562736372696265723D526567756C61724C696E65
17:27:09:      Attribute 26 35
00000009011D683332332D6976722D6F75743D5461726966663A556E6B6E6F776E
17:27:09:      Attribute 26 22 0000000901107072652D62797465732D696E3D30
17:27:09:      Attribute 26 23 0000000901117072652D62797465732D6F75743D30
17:27:09:      Attribute 26 21 00000009010F7072652D70616B732D696E3D30
17:27:09:      Attribute 26 22 0000000901107072652D70616B732D6F75743D30
17:27:09:      Attribute 26 22 0000000901106E61732D72782D73706565643D30
17:27:09:      Attribute 26 22 0000000901106E61732D74782D73706565643D30
17:27:09:      Attribute 41 6 00000000
17:27:09: RADIUS: Received from id 12 10.0.0.1:1824, Accounting-response, len 20

```

**Related Commands**

Command	Description
<a href="#">debug aaa accounting</a>	Displays information on accountable events as they occur.
<a href="#">debug aaa authentication</a>	Displays information on AAA/TACACS+ authentication.

# debug ras

To display the types and addressing of Registration, Admission and Status (RAS) messages sent and received, use the **debug ras** command in privileged EXEC mode. To disable debug output, use the **no** form of this command.

**debug ras**

**no debug ras**

## Syntax Description

This command has no keywords or arguments.

## Defaults

This command is disabled by default.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.2(2)XB1	This command was implemented on the Cisco AS5850
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

## Usage Guidelines

Use the **debug ras** command to display the types and addressing of RAS messages sent and received. The debug output lists the message type using mnemonics defined in International Telecommunications Union-Telecommunication (ITU-T) specification H.225.

## Examples

In the following output, gateway GW13.cisco.com sends a RAS registration request (RRQ) message to gatekeeper GK15.cisco.com at IP address 10.9.53.15. GW13.cisco.com then receives a registration confirmation (RCF) message from the gatekeeper. If there is no response, it could mean that the gatekeeper is offline or improperly addressed. If you receive a reject (RRJ) message, it could mean that the gatekeeper is unable to handle another gateway or that the registration information is incorrect.

```
Router# debug ras

*Mar 13 19:53:34.231:      RASLib::ras_sendto:msg length 105 from
                        10.9.53.13:8658 to 10.9.53.15:1719
*Mar 13 19:53:34.231:      RASLib::RASSendRRQ:RRQ (seq# 36939) sent
                        to 10.9.53.15
*Mar 13 19:53:34.247:      RASLib::RASRecvData:successfully rcvd
                        message of length 105 from 10.9.53.15:1719
*Mar 13 19:53:34.251:      RASLib::RASRecvData:RCF (seq# 36939) rcvd
                        from [10.9.53.15:1719] on sock [0x6168356C
```

# debug redundancy

To enable the display of events for troubleshooting redundant DSCs, use the **debug redundancy** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug redundancy {all | ui | clk | hub}
```

```
no debug redundancy {all | ui | clk | hub}
```

## Syntax Description

<b>all</b>	Displays all available information on redundant DSCs, including that specified by the following options in this table.
<b>ui</b>	Displays information on the user interface of the redundant DSCs.
<b>clk</b>	Displays information on the clocks of the redundant DSCs.
<b>hub</b>	Displays information on the BIC hub of the redundant DSCs. The hub is the Fast Ethernet link between the router and the DSC.

## Defaults

The command is disabled by default.

## Command Modes

Privileged EXEC

## Command History

Release	Modification
11.3(6)AA	This command was introduced.

## Usage Guidelines

This command is issued from the router shelf console.

## Examples

The output from this command consists of event announcements that can be used by authorized troubleshooting personnel.