

debug arap

Use the **debug arap** EXEC command to display AppleTalk Remote Access Protocol (ARAP) events. The **no** form of this command disables debugging output.

```
[no] debug arap {internal | memory | mnp4 | v42bis} [linenum [aux | console | tty | vty]]
```

Syntax Description

internal	Debugs internal ARA packets.
memory	Debugs memory allocation for ARA.
mnp4	Debugs low-level asynchronous serial protocol.
v42bis	Debugs V.42bis compression.
<i>linenum</i>	(Optional) Line number. The number ranges from 0 to 999, depending on what type of line is selected.
aux	(Optional) Auxiliary line.
console	(Optional) Primary terminal line.
tty	(Optional) Physical terminal asynchronous line.
vty	(Optional) Virtual terminal line.

Usage Guidelines

Use the **debug arap** command with the **debug callback** command on access servers to debug dial-in and callback events.

Use the **debug modem** command to help catch problems related to ARAP auto-detection (that is, **autoselect arap**). These problems are very common and are most often caused by modems, which are the most common cause of failure in ARAP connection and configuration sessions.

Sample Display

The following is sample output from the **debug arap internal** command:

```
Router# debug arap internal

ARAP: ----- SRVVERSION -----
ARAP: ----- ACKing 0 -----
ARAP: ----- AUTH_CHALLENGE -----
arapsec_local_account setting up callback
ARAP: ----- ACKing 1 -----
ARAP: ----- AUTH_RESPONSE -----
arap_startup initiating callback ARAP 2.0
ARAP: ----- CALLBACK -----
TTY7 Callback process initiated, user: dialback dialstring 40
TTY7 Callback forced wait = 4 seconds
TTY7 ARAP Callback Successful - await exec/autoselect pickup
TTY7: Callback in effect
ARAP: ----- STARTINFOFROMSERVER -----
ARAP: ----- ACKing 0 -----
ARAP: ----- ZONELISTINFO -----
```

debug arap

```
ARAP: ----- ZONELISTINFO -----  
ARAP: ----- ZONELISTINFO -----  
ARAP: ----- ZONELISTINFO -----  
ARAP: ----- ZONELISTINFO -----
```

Related Commands

debug callback
debug modem

debug arp

Use the **debug arp** EXEC command to display information on Address Resolution Protocol (ARP) transactions. The **no** form of this command disables debugging output.

[no] debug arp

Usage Guidelines

Use this command when some nodes on a TCP/IP network are responding, but others are not. It shows whether the router is sending ARPs and whether it is receiving ARPs.

Sample Display

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

```
Router# debug arp

IP ARP: sent req src 172.16.22.7 0000.0c01.e117, dst 172.16.22.96 0000.0000.0000
IP ARP: rcvd rep src 172.16.22.96 0800.2010.b908, dst 172.16.22.7
IP ARP: rcvd req src 172.16.6.10 0000.0c00.6fa2, dst 172.16.6.62
IP ARP: rep filtered src 172.16.22.7 aa92.1b36.a456, dst 255.255.255.255 ffff.ffff.ffff
IP ARP: rep filtered src 172.16.9.7 0000.0c00.6b31, dst 172.16.22.7 0800.2010.b908
```

In the output, each line of output represents an ARP packet that the router sent or received.

Explanations for the individual lines of output follow.

The first line indicates that the router at IP address 172.16.22.7 and MAC address 0000.0c01.e117 sent an ARP request for the MAC address of the host at 172.16.22.96. The series of zeros (0000.0000.0000) following this address indicate that the router is currently unaware of the MAC address.

```
IP ARP: sent req src 172.16.22.7 0000.0c01.e117, dst 172.16.22.96 0000.0000.0000
```

The second line indicates that the router at IP address 172.16.22.7 receives a reply from the host at 172.16.22.96 indicating that its MAC address is 0800.2010.b908:

```
IP ARP: rcvd rep src 172.16.22.96 0800.2010.b908, dst 172.16.22.7
```

The third line indicates that the router receives an ARP request from the host at 172.16.6.10 requesting the MAC address for the host at 172.16.6.62:

```
IP ARP: rcvd req src 172.16.6.10 0000.0c00.6fa2, dst 172.16.6.62
```

The fourth line indicates that another host on the network attempted to send the router an ARP reply for its own address. The router ignores meaningless replies. Usually, meaningless replies happen if someone is running a bridge in parallel with the router and is allowing ARP to be bridged. This condition indicates a network misconfiguration.

```
IP ARP: rep filtered src 172.16.22.7 aa92.1b36.a456, dst 255.255.255.255 ffff.ffff.ffff
```

The fifth line indicates that another host on the network attempted to inform the router that it is on network 172.16.9.7, but the router does not know that the network is attached to a different router interface. The remote host (probably a PC or an X terminal) is misconfigured. If the router were to install this entry, it would deny service to the real machine on the proper cable.

```
IP ARP: rep filtered src 172.16.9.7 0000.0c00.6b31, dst 172.16.22.7 0800.2010.b908
```

debug asp packet

Use the **debug asp packet** EXEC command to display information on all asynchronous security protocols operating on the router. The **no** form of this command disables debugging output.

[no] debug asp packet

Usage Guidelines

The router uses asynchronous security protocols such as ADT Security Systems, Inc., Adplex, and Diebold to transport alarm blocks between two devices (such as a security alarm system console and an alarm panel). The alarm blocks are transported in passthrough mode using BSTUN encapsulation.

Sample Display

The following is partial sample output from the **debug asp packet** command for asynchronous security protocols when packet debugging is enabled on an asynchronous line carrying Diebold alarm traffic. In this example, two polls are sent from the Diebold alarm console to two alarm panels that are multidropped from a single RS-232 interface. The alarm panels have device addresses F0 and F1. The example trace indicates that F1 is responding and F0 is not responding. At this point, you need to examine the physical link and possibly use a datascopes to determine why the device is not responding.

```
Router# debug asp packet

12:19:48: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FF4C42
12:19:49: ASP: Serial5: ADI-Tx: Data (1 bytes): 88
12:19:49: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FF9B94
12:20:47: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FF757B
12:20:48: ASP: Serial5: ADI-Tx: Data (1 bytes): F3
12:20:48: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FFB1BE
12:21:46: ASP: Serial5: ADI-Rx: Data (4 bytes): F1FFE6E8
12:21:46: ASP: Serial5: ADI-Tx: Data (1 bytes): 6F
12:21:46: ASP: Serial5: ADI-Rx: Data (4 bytes): F0FFC1CE
```

Table 20 describes the fields and messages.

Table 20 Debug ASP Packet Field Descriptions

Field	Description
ASP	Async security protocol packet.
Serial 5	Interface receiving and transmitting the packet.
ADI-Rx	Packet is being received.
ADI-TX	Packet is being transmitted.
Data (<i>n</i> bytes)	Type and size of the packet.
F1FF4c42	Alarm panel device address.

debug atm errors

Use the **debug atm errors** EXEC command to display Asynchronous Transfer Mode (ATM) errors. The **no** form of this command disables debugging output.

[no] debug atm errors

Sample Display

The following is sample output from the **debug atm errors** command:

```
Router# debug atm errors  
  
ATM(ATM2/0): Encapsulation error, link=7, host=836CA86D.  
ATM(ATM4/0): VCD#7 failed to echo OAM. 4 tries
```

The first line of output indicates that a packet was routed to the ATM interface, but no static map was set up to route that packet to the proper virtual circuit.

The second line of output shows that an OAM F5 (virtual circuit) cell error occurred.

debug atm events

Use the **debug atm events** EXEC command to display ATM events. The **no** form of this command disables debugging output.

[no] debug atm events

Usage Guidelines

This command displays ATM events that occur on the ATM interface processor and is useful for diagnosing problems in an ATM network. It provides an overall picture of the stability of the network. In a stable network, the **debug atm events** command does not return any information. If the command generates numerous messages, the messages can indicate the possible source of problems.

When configuring or making changes to a router or interface for ATM, enable **debug atm events**. Doing so alerts you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

Sample Display

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

```
Router# debug atm events

RESET(ATM4/0): PLIM type is 1, Rate is 100Mbps
aip_disable(ATM4/0): state=1
config(ATM4/0)
aip_love_note(ATM4/0): asr=0x201
aip_enable(ATM4/0)
aip_love_note(ATM4/0): asr=0x4000
aip_enable(ATM4/0): restarting VCs: 7
aip_setup_vc(ATM4/0): vc:1 vpi:1 vci:1
aip_love_note(ATM4/0): asr=0x200
aip_setup_vc(ATM4/0): vc:2 vpi:2 vci:2
aip_love_note(ATM4/0): asr=0x200
aip_setup_vc(ATM4/0): vc:3 vpi:3 vci:3
aip_love_note(ATM4/0): asr=0x200
aip_setup_vc(ATM4/0): vc:4 vpi:4 vci:4
aip_love_note(ATM4/0): asr=0x200
aip_setup_vc(ATM4/0): vc:6 vpi:6 vci:6
aip_love_note(ATM4/0): asr=0x200
aip_setup_vc(ATM4/0): vc:7 vpi:7 vci:7
aip_love_note(ATM4/0): asr=0x200
aip_setup_vc(ATM4/0): vc:11 vpi:11 vci:11
aip_love_note(ATM4/0): asr=0x200
```

Table 21 describes significant fields in the output.

Table 21 Debug ATM Events Field Descriptions

Field	Description
PLIM type	Indicates the interface rate in Mbps. Possible values are <ul style="list-style-type: none"> • 1 = TAXI(4B5B) 100 Mbps • 2 = SONET 155 Mbps • 3 = E3 34 Mbps

Table 21 Debug ATM Events Field Descriptions (Continued)

Field	Description
state	Indicates current state of the AIP. Possible values are <ul style="list-style-type: none"> • 1 = An ENABLE will be issued soon • 0 = The AIP will remain shut down
asr	Defines a bitmask, which indicates actions or completions to commands. Valid bitmask values are <ul style="list-style-type: none"> • 0x0800 = AIP crashed, reload may be required. • 0x0400 = AIP detected a carrier state change. • 0x0n00 = Command completion status. Command completion status codes are <ul style="list-style-type: none"> — n = 8 Invalid PLIM detected — n = 4 Command failed — n = 2 Command completed successfully — n = 1 CONFIG request failed — n = 0 Invalid value

The following line indicates that the ATM Interface Processor (AIP) was reset. The PLIM TYPE detected was 1, so the maximum rate is set to 100 Mbps.

```
RESET(ATM4/0): PLIM type is 1, Rate is 100Mbps
```

The following line indicates that the ATM Interface Processor (AIP) was given a **shutdown** command, but the current configuration indicates that the AIP should be up:

```
aip_disable(ATM4/0): state=1
```

The following line indicates that a configuration command has been completed by the AIP:

```
aip_love_note(ATM4/0): asr=0x201
```

The following line indicates that the AIP was given a **no shutdown** command to take it out of shutdown:

```
aip_enable(ATM4/0)
```

The following line indicates that the AIP detected a carrier state change. It does not indicate that the carrier is down or up, only that it has changed.

```
aip_love_note(ATM4/0): asr=0x4000
```

The following line of output indicates that the AIP enable function is restarting all PVCs automatically:

```
aip_enable(ATM4/0): restarting VCs: 7
```

The following lines of output indicate that PVC 1 was set up and a successful completion code was returned:

```
aip_setup_vc(ATM4/0): vc:1 vpi:1 vci:1
aip_love_note(ATM4/0): asr=0x200
```

debug atm oam

Use the **debug atm oam** EXEC command to display ATM operation and maintenance (OAM) events. The **no** form of this command disables debugging output.

[no] debug atm oam

Sample Display

The following is sample output from the **debug atm oam** command:

```
Router# debug atm oam

ATM4/0(O): VCD:0x0 DM:0x300 *OAM Cell* Length:0x39
0000 0300 0070 007A 0018 0100 0000 05FF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
FFFF FFFF FFFF FFFF FF6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A6A 6A00 0000
```

Table 22 describes the output fields.

Table 22 Debug ATM OAM Field Descriptions

Field	Description
0000	VCD Special OAM indicator
0300	Descriptor MODE bits for the AIP
0	GFC (4 bits)
07	VPI (8 bits)
0007	VCI (16 bits)
A	Payload type field(PTI)(4 bits)
00	Header Error Correction(8 bits)
1	OAM Fault mgmt. cell(4 bits)
8	OAM LOOPBACK indicator (4 bits)
01	Loopback indicator value, always 1(8 bits)
00000005	Loopback unique ID, sequence number (32 bits)
FF6A	F's and 6A required in the remaining ATM cell, per UNI3.0

debug atm packet

Use the **debug atm packet** EXEC command to display per-packet debugging output. The output reports information online when a packet is received or a transmit is attempted. The **no** form of this command disables debugging output.

```
[no] debug atm packet [interface atm number [vcd vcd-number] | vc vpi/vci | vc-name]
```

Syntax Description

interface atm <i>number</i>	(Optional) ATM interface or subinterface number.
vcd <i>vcd-number</i>	(Optional) Number of the virtual circuit designator (VCD).
vc <i>vpi/vci</i>	(Optional) VPI and VCI numbers of the VC.
<i>vc-name</i>	(Optional) Name of the PVC or SVC.

Usage Guidelines

The **debug atm packet** command displays all process-level ATM packets for both outbound and inbound packets. This command is useful for determining whether packets are being received and transmitted correctly.

For transmitted packets, the information is displayed only after the protocol data unit (PDU) is entirely encapsulated and a next hop virtual circuit (VC) is found. If information is not displayed, the address translation probably failed during encapsulation. When a next hop VC is found, the packet is displayed exactly as it will be presented on the wire. Having a display indicates the packets are properly encapsulated for transmission.

For received packets, information is displayed for all incoming frames. The display can show whether the transmitting station properly encapsulates the frames. Because all incoming frames are displayed, this information is useful when performing back-to-back testing and corrupted frames cannot be dropped by an intermediary ATM switch.

The **debug atm packet** command also displays the initial bytes of the actual PDU in hexadecimal. This information can be decoded only by qualified support or engineering personnel.

Note Because the **debug atm packet** command generates a significant amount of output for every packet processed, use it only when traffic on the network is low, so other activity on the system is not adversely affected.

Sample Display

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

```
Router# debug atm packet

ATM2/0.5(I): VCD:0x9 VCI:0x23 Type:0x0 SAP:AAAA CTL:03 OUI:000000 TYPE:0800 Length0x70
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFF 0105
```

Table 23 describes significant fields.

Table 23 Debug ATM Packet Field Descriptions

Field	Description
ATM2/0.5	Indicates the subinterface that generated this packet.
(I)	Indicates a receive packet. (O) indicates an output packet.
VCD: 0xn	Indicates the virtual circuit associated with this packet, where <i>n</i> is some value.
DM: 0xnmm	Indicates the descriptor mode bits on output only, where <i>mmmm</i> is a hexadecimal value.
TYPE: <i>n</i>	Shows the encapsulation type for this packet.
Length: <i>n</i>	Shows the total length of the packet including the ATM header(s).

The following two lines of output are the binary data, which are the contents of the protocol PDU before encapsulation at the ATM:

```
4500 002E 0000 0000 0209 92ED 836C A26E FFFF FFFF 1108 006D 0001 0000 0000
A5CC 6CA2 0000 000A 0000 6411 76FF 0100 6C08 00FF FFFF 0003 E805 DCFF 0105
```

debug atm pvcd

Use the **debug atm pvcd** EXEC command to display the PVC Discovery events and ILMI MIB traffic used when discovering PVCs. The **no** form of this command disables debugging output.

[no] debug atm pvcd

Usage Guidelines

This command is primarily used by your technical support representative.

Sample Display

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

```
Router# debug atm pvcd

PVCD: PVCD enabled w/ Subif
PVCD(ATM2/0): clearing event queue
PVCD: ATM2/0 Forgetting discovered PVCs...
PVCD: Removing all dynamic PVCs on ATM2/0
PVCD: Restoring MIXED PVCs w/ default parms on ATM2/0
PVCD: Marking static PVCs as UNKNWN on ATM2/0
PVCD: Marking static PVC 0/50 as UNKNWN on ATM2/0 ...
PVCD: Trying to discover PVCs on ATM2/0...
PVCD: pvcd_discoverPVCs
PVCD: pvcd_ping
PVCD: atmPortEntry.5.0 = 2
PVCD: pvcd_getPeerVccTableSize
PVCD: atmAtmLayerEntry.5.0 = 13
PVCD:end allocating VccTable size 13
PVCD: pvcd_getPeerVccTable
PVCD:***** ATM2/0: getNext on atmVccEntry = NULL TYPE/VALUE numFiledS = 19 numVccs =
13
PVCD: Creating Dynamic PVC 0/33 on ATM2/0
PVCD(ATM2/0): Before atm_update_inheritance() and atm_create_pvc() VC 0/33: DYNAMIC
PVCD: After atm_create_pvc() VC 0/33: DYNAMIC0/33 on ATM2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/34 on ATM2/0
PVCD(ATM2/0): Before atm_update_inheritance() and atm_create_pvc() VC 0/34: DYNAMIC
PVCD: After atm_create_pvc() VC 0/34: DYNAMIC0/34 on ATM2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/44 on ATM2/0
PVCD(ATM2/0): Before atm_update_inheritance() and atm_create_pvc() VC 0/44: DYNAMIC
PVCD: After atm_create_pvc() VC 0/44: DYNAMIC0/44 on ATM2/0 : UBR PCR = -1
PVCD: PVC 0/50 with INHERITED_QOSTYPE
PVCD: atm_oi_state_change ( 0/50, 1 = ILMI_VC_UP )
PVCD: Creating Dynamic PVC 0/60 on ATM2/0
PVCD(ATM2/0): Before atm_update_inheritance() and atm_create_pvc() VC 0/60: DYNAMIC
PVCD: After atm_create_pvc() VC 0/60: DYNAMIC0/60 on ATM2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/80 on ATM2/0
PVCD(ATM2/0): Before atm_update_inheritance() and atm_create_pvc() VC 0/80: DYNAMIC
PVCD: After atm_create_pvc() VC 0/80: DYNAMIC0/80 on ATM2/0 : UBR PCR = -1
PVCD: Creating Dynamic PVC 0/99 on ATM2/0
```

debug bri

Use the **debug bri** EXEC command to display debugging information on Integrated Services Digital Networks (ISDN) Basic Rate Interface (BRI) routing activity. The **no** form of this command disables debugging output.

[no] debug bri

Usage Guidelines

The **debug bri** command indicates whether the ISDN code is enabling and disabling the B-channels when attempting an outgoing call. This command is available for the low-end router products that have a multi-BRI network interface module installed.

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

Sample Display

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

```
Router# debug bri

BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: write_sid: wrote 15 for subunit 0, slot 1.
BRI: write_sid: wrote 17 for subunit 0, slot 1.
BRI: write_sid: wrote 6 for subunit 0, slot 1.
BRI: write_sid: wrote 8 for subunit 0, slot 1.
BRI: write_sid: wrote 11 for subunit 0, slot 1.
BRI: write_sid: wrote 13 for subunit 0, slot 1.
BRI: write_sid: wrote 29 for subunit 0, slot 1.
BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: write_sid: wrote 15 for subunit 0, slot 1.
BRI: write_sid: wrote 17 for subunit 0, slot 1.
BRI: write_sid: wrote 20 for subunit 0, slot 1.
BRI: Starting Power Up timer for unit = 0.
BRI: write_sid: wrote 3 for subunit 0, slot 1.
BRI: Starting T3 timer after expiry of PUP timeout for unit = 0, current state is F4.
BRI: write_sid: wrote FF for subunit 0, slot 1.
BRI: Activation for unit = 0, current state is F7.
BRI: enable channel B1
BRI: write_sid: wrote 14 for subunit 0, slot 1.

%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to up!!!
BRI: disable channel B1
BRI: write_sid: wrote 15 for subunit 0, slot 1.

%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to down
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to
down
```

The following line indicates that an internal command was written to the interface controller. The subunit identifies the first interface in the slot.

```
BRI: write_sid: wrote 1B for subunit 0, slot 1.
```

The following line indicates that the power-up timer was started for the named unit:

```
BRI: Starting Power Up timer for unit = 0.
```

The following lines indicate that the channel or the protocol on the interface changed state:

```
%LINK-3-UPDOWN: Interface BRI0: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRI0: B-Channel 1, changed state to up.!!!
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRI0: B-Channel 1, changed state to
down
```

The following line indicates that the channel was disabled:

```
BRI: disable channel B1
```

Lines of output not described are for use by support staff only.

Related Commands

```
debug isdn event
debug isdn q921
debug isdn q931
```

debug bsc event

Use the **debug bsc event EXEC** command to display all events occurring in the Binary Synchronous Communications (BSC) feature. The **no** form of this command disables debugging output.

[no] debug bsc event [*number*]

Syntax Description

number (Optional) Group number.

Usage Guidelines

This command traces all interfaces configured with a **bsc protocol-group** *number* command.

Sample Display

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

```
Router# debug bsc event

BSC: Serial2          POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFfile
BSC: Serial2          POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFfile
BSC: Serial2          POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFfile
0:04:32: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2          POLLEE-FSM inp:E_RxEtx old_st:CU_Down new_st:TCU_EOFfile
0:04:32: BSC: Serial2 :SDI-rx: 5 bytes
BSC: Serial2          POLLEE-FSM inp:E_RxEnq old_st:CU_Down new_st:TCU_EOFfile
BSC: Serial2          POLLEE-FSM inp:E_Timeout old_st:CU_Down new_st:TCU_InFile
BSC: Serial2          POLLEE-FSM inp:E_Timeout old_st:CU_Idle new_st:TCU_InFile
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2, changed state to up
%LINK-3-UPDOWN: Interface Serial2, changed state to up
BSC: Serial2          POLLEE-FSM inp:E_Timeout old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2          POLLEE-FSM inp:E_RxEtx old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :SDI-rx: 5 bytes
BSC: Serial2          POLLEE-FSM inp:E_RxEnq old_st:CU_Idle new_st:TCU_InFile
0:04:35: BSC: Serial2 :NDI-rx: 3 bytes
```

Related Commands

debug bsc packet
debug bstun events

debug bsc packet

Use the **debug bsc packet** EXEC command to display all frames traveling through the Binary Synchronous Communications (BSC) feature. The **no** form of this command disables debugging output.

```
[no] debug bsc packet [group number] [buffer-size bytes]
```

Syntax Description

group number	(Optional) Group number.
buffer-size bytes	(Optional) Number of bytes displayed per packet (defaults to 20).

Usage Guidelines

This command traces all interfaces configured with a **bsc protocol-group number** command.

Sample Display

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

```
Router# debug bsc packet
0:23:33: BSC: Serial2      :NDI-rx : 27 bytes 401A400227F5C31140C11D60C8C5D3D3D51D4013
0:23:33: BSC: Serial2      :SDI-tx : 12 bytes 00323237FF3232606040402D
0:23:33: BSC: Serial2      :SDI-rx : 2 bytes 1070
0:23:33: BSC: Serial2      :SDI-tx : 27 bytes 401A400227F5C31140C11D60C8C5D3D3D51D4013
0:23:33: BSC: Serial2      :SDI-rx : 2 bytes 1061
0:23:33: BSC: Serial2      :SDI-tx : 5 bytes 00323237FF
```

Related Commands

- debug bsc event**
- debug bstun events**

debug bstun events

Use the **debug bstun events** EXEC command to display BSTUN connection events and status. The **no** form of this command disables debugging output.

[no] debug bstun events *number*

Syntax Description

number (Optional) Group number.

Usage Guidelines

When you enable the **debug bstun events** command, messages showing connection establishment and other overall status messages are displayed.

You can use the **debug bstun events** command to assist you in determining whether the BSTUN peers are configured correctly and are communicating. For example, if you enable the **debug bstun packet** command and you do not see any packets, you may want to enable event debugging.

Note Also refer to the **debug bsc packet** and **debug bsc event** commands. Currently, these two commands support the only protocol working through the BSTUN tunnel. Sometimes frames do not go through the tunnel because they have been discarded at the BSC protocol level.

Sample Displays

The following is sample output from the **debug bstun events** command of keepalive messages working correctly. If the routers are configured correctly, at least one router will show reply messages.

```
Router# debug bstun packet

BSTUN: Received Version Reply opcode from (all[2])_172.16.12.2/1976 at 1360
BSTUN: Received Version Request opcode from (all[2])_172.16.12.2/1976 at 1379
BSTUN: Received Version Reply opcode from (all[2])_172.16.12.2/1976 at 1390
```

Note In a scenario where there is constantly loaded bi-directional traffic, you might not see keepalive messages because they are sent only when the remote end has been silent for the keepalive period.

The following is sample output from the **debug bstun events** output of an event trace in which the wrong TCP address has been specified for the remote peer. These are non-keepalive related messages.

```
Router# debug bstun packet

BSTUN: Change state for peer (C1[1])172.16.12.22/1976 (closed->opening)
BSTUN: Change state for peer (C1[1])172.16.12.22/1976 (opening->open wait)
%BSTUN-6-OPENING: CONN: opening peer (C1[1])172.16.12.22/1976, 3
BSTUN: tcpd sender in wrong state, dropping packet
BSTUN: tcpd sender in wrong state, dropping packet
BSTUN: tcpd sender in wrong state, dropping packet
```

Related Commands

- debug bsc event**
- debug bsc packet**
- debug bstun packet**

debug bstun packet

Use the **debug bstun packet** EXEC command to display packet information on packets traveling through the BSTUN links. The **no** form of this command disables debugging output.

[no] debug bstun packet [group number] [buffer-size bytes]

Syntax Description

group number	(Optional) BSTUN group number.
buffer-size bytes	(Optional) Number of bytes displayed per packet (defaults to 20).

Sample Display

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

```
Router# debug bstun packet
BSTUN bsc-local-ack: 0:00:00 Serial2      SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:00 Serial2      SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:06 Serial2      NDI: Addr: 40 Data: 0227F5C31140C11D60C8
```

Related Command

debug bstun events

debug cable env

Use the **debug cable env** EXEC command to display information about the Cisco uBR7246 physical environment, including internal temperature, midplane voltages, fan performance, and power supply voltages. The **no** form of this command disables debugging output.

[no] debug cable env

Usage Guidelines

This command is used to debug the sensor circuitry used to measure internal temperature, midplane voltages, fan performance, and power supply voltages on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable env** command:

```
Router: debug cable env
ENVM: ps id=0xFF0, v=0x2050, r=0xC0AB, pstype=1
ENVM: ps id=0x2FD0, v=0x2050, r=0x24201, pstype=27
ENVM: Sensor 0: a2dref=131, a2dact=31, vref=12219, vact=1552
      Alpha=8990, temp=27
```

Table 24 describes significant fields in the output.

Table 24 Debug Cable ENV Field Descriptions

Field	Description
ps id	Power supply raw voltage reading.
pstype	Power supply type determined from ps id, v, and r. The Cisco uBR7246 contains dual power supplies so id information for two types is usually printed.
Sensor	Sensor number.
a2dref	Analog to digital converter reference reading.
a2dact	Analog to digital converter actual (measured reading).
vref	Reference voltage.
vact	Actual voltage.
Alpha	Raw temperature reading.
temp	Temperature corresponding to Alpha.

Related Commands

show environment all
show environment last
show environment table

debug cable err

Use the **debug cable err** EXEC command to display errors that occur in the cable MAC protocols. The **no** form of this command disables debugging output.

[no] debug cable err

Usage Guidelines

This command is used to display unexpected DOCSIS MAC protocol messages. When the Cisco uBR7246 does not expect to receive a specific MAC message, an error message and hex dump are printed. Other miscellaneous error conditions may result in output.

Sample Display

The following is sample output from the **debug cable err** command:

```
Router: debug cable err
This is a UCD Message
This is a MAP Message
This is a RNG_RSP Message
This is a REG_RSP Message
This is a UCC_REQ Message
This is a BPKM_RSP Message
This is a TRI_TCD Message
This is a TRI_TSI Message
This is a unrecognized MCNS message

ERROR:#####TICKS PER MSLOT NOT POWER OF 2####
```

debug cable keyman

Use the **debug cable keyman** EXEC command to activate debugging of TEK and KEK baseline privacy key activity. The **no** form of this command disables debugging output.

[no] debug cable keyman

Usage Guidelines

This command activates debugging of the TEK and KEK baseline privacy key activity. When this command is activated, all activity related to KEK and TEK keys will be displayed on the Cisco uBR7246 console. This command is used to display encryption key management debugging output.

Sample Display

The following is sample output from the **debug cable keyman** command:

```
Router: debug cable keyman
Read Verify DES failed with SID %2x
    Verify key failed with SID %2x : setvalue = %11x, readback = %11x
    Verify iv failed with SID %2x : setvalue = %11x, readback = %11x
Next TEK lifetime check is set to %u seconds.
    Next Multicast TEK lifetime check is set to 1 seconds

[UCAST_TEK] :", idbp->hw_namestring);
    show_sid_key_chain(ds, &ds->mcast_sid_key_list_hdr);

[MCAST_TEK] :", idbp->hw_namestring);
    buginf("\nSID : %4x\t", sidkey->sid);
    buginf("seq : %2x\t current : %2x\n", sidkey->key_seq_num,
        sidkey->current_key_num);
    buginf(" Status[0] : %x\tDES IV[0] : %11x\tKey Life[0]: %u sec\n",
        sidkey->key_status[0], sidkey->des_key[0].iv,
        compute_remain_lifetime(&sidkey->des_key[0]));

    buginf(" Status[1] : %x\tDES IV[1] : %11x\tKey Life[1]: %u sec\n",
        sidkey->key_status[1], sidkey->des_key[1].iv,
        compute_remain_lifetime(&sidkey->des_key[1]));
```

debug cable phy

Use the **debug cable phy** EXEC command to activate debugging of messages generated in the cable physical layer. The **no** form of this command disables debugging output.

[no] debug cable phy

Usage Guidelines

This command activates debugging of messages generated in the cable phy, which is the physical layer where upstream and downstream activity between the Cisco uBR7246 and the HFC network is controlled. When this command is activated, any messages generated in the cable phy will be displayed on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable phy** command:

```
Router: debug cable phy
cmts_phy_init: mac_version == BCM3210_FPGA
    bcm3033_set_tx_sym_rate(5056941)
    stintctl = 0x54484800
    bcm3033_set_tx_if_freq(44000000)
    stfreqctl = 0x5BAAAAAA
cmts_phy_init_us: U0 part_id = 0x3136, revid = 0x05, rev_id2 = 0x64
cmts_phy_init: mac_version == BCM3210_FPGA
Media access controller chip version.
    bcm3033_set_tx_sym_rate(5056941)
    stintctl = 0x54484800
Physical layer symbol rate register value.
00:51:49: bcm3033_set_tx_if_freq(44000000)
00:51:49: stfreqctl = 0x5BAAAAAA
Physical layer intermediate frequency (IF) register value.
00:51:49: cmts_phy_init_us: U0 part_id = 0x3136, revid = 0x05, rev_id2 = 0x64
Physical layer receiver chip part version.
```

debug cable privacy

Use the **debug cable privacy** EXEC command to activate debugging of baseline privacy. The **no** form of this command disables debugging output.

[no] debug cable privacy

Usage Guidelines

This command activates debugging of baseline privacy. When this command is activated, any messages generated by the spectrum manager will be displayed on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable privacy** command:

```
Router: debug cable privacy  
Removing both odd and even keys for sid %x.  
  
Invalid Len for TLV_SERIAL_NUM_TYPE : %d.  
  
Invalid Len for TLV_MANUF_ID_TYPE : %d.  
  
Invalid Len for TLV_MANUF_ID_TYPE : %d.
```

debug cable qos

Use the **debug cable qos** EXEC command to activate quality of service (QoS) debugging. The **no** form of this command disables debugging output.

[no] debug cable qos

Usage Guidelines

This command activates debugging of QoS. When this command is activated, any messages related to QoS parameters will be displayed on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable qos** command:

```
Router: debug cable qos
      CMTS_QOS_LOG_NO_MORE_QOS_INDEX
Modems cannot add more entries to the class of service table.
      CMTS_QOS_LOG_NOMORE_QOSPRF_MEM
Memory allocation error when creating class of service table entry.
      CMTS_QOS_LOG_NO_CREATION_ALLOWED
Class of service entry cannot be created by modem. Use CLI or SNMP
interface instead of the modem's TFTP configuration file.
      CMTS_QOS_LOG_CANNOT_REGISTER_COS_SID
A service identifier (SID) could not be assigned to the registering modem.
      CMTS_QOS_LOG_CANNOT_DEREGISTER_COS_SID
The modem's service identifier (SID) was already removed.
      CMTS_QOS_LOG_MSLOT_TIMEBASE_WRAPPED
The 160 KHz timebase clock drives a 26-bit counter which wraps around
approximately every 7 minutes. This message is generated every time it
wraps around.
```

debug cable range

Use the **debug cable range** EXEC command to display ranging messages from cable modems on the HFC network. The **no** form of this command disables debugging output.

[no] debug cable range

Usage Guidelines

This command activates debugging of ranging messages from cable modems on the HFC network. When this command is activated, any ranging messages generated when cable modems request or change their upstream frequencies will be displayed on the Cisco uBR7246 console. Use this command to display the details of the initial and station maintenance procedures. The initial maintenance procedure is used for link establishment. The station maintenance procedure is used for link keep-alive monitoring.

Sample Display

The following is sample output from the **debug cable range** command when a modem first seeks to establish a link to the Cisco uBR7246:

```
Router: debug cable range
Got a ranging request
SID value is 0 on Interface Cable3/0/U0
CM mac address 00:10:7B:43:AA:21 Timing offset is 3312
3E 1E 3F FF 00 00 59 BF 01 15 F8 01 A7 00 0C F0
```

The SID value of 0 indicates that the modem has no assigned service identifier. The “CM mac address” is the MAC address of the modem's radio frequency (RF) interface, not its Ethernet interface. The “Timing offset” is a measure of the distance between the modem and the Cisco uBR7246 expressed in 10.24 MHz clocks. This value is adjusted down to zero by the maintenance procedures. The first 16 bytes of the prepended header of the message are dumped in hexadecimal.

The following is sample output when the modem is first assigned a SID during initial maintenance:

```
CM mac address 0010.7b43.aa21
found..Assigned SID #2 on Interface Cable3/0/U0
Timing offset is CF0
Power value is 15F8, or -1 dB
Freq Error = 423, Freq offset is 1692
Ranging Modem with Sid 2 on i/f : Cable3/0/U0
```

The following is sample output when the modem is reassigned the same SID during initial maintenance:

```
Initial Range Message Received on Interface Cable3/0/U0
CMTS reusing old sid : 2 for modem : 0010.7b43.aa21
Timing offset is CF0
Power value is 15F8, or -1 dB
Freq Error = 423, Freq offset is 1692
Ranging Modem with Sid 2 on i/f : Cable3/0/U0
```

The following is sample output when the modem is polled by the uBR7246 during station maintenance. Polling happens at a minimum rate of once every 10 seconds:

```
Ranging Modem with Sid 2 on i/f : Cable3/0/U0

Got a ranging request
SID value is 2 on Interface Cable3/0/U0
CM mac address 00:10:7B:43:AA:21
Timing offset is 0
Power value is 1823, or -1 dB
Freq Error = 13, Freq offset is 0
Ranging has been successful for SID 2 on Interface Cable3/0/U0
```

debug cable reset

Use the **debug cable reset** EXEC command to display reset messages from cable interfaces. The **no** form of this command disables debugging output.

[no] debug cable reset

Usage Guidelines

This command activates display of reset messages from cable interfaces.

Sample Display

The following is sample output from the **debug cable reset** command when the interface is reset due to complete loss of receive packets:

```
Router: debug cable reset  
Resetting CMTS interface.
```

debug cable specmgmt

Use the **debug cable specmgmt** EXEC command to debug spectrum management (frequency agility) on the HFC network. The **no** form of this command disables debugging output.

[no] debug cable specmgmt

Usage Guidelines

This command activates debugging of spectrum management (frequency agility) on the HFC network. When this command is activated, any messages generated due to spectrum group activity will be displayed on the Cisco uBR7246 console. Spectrum group activity can be additions or changes to spectrum groups, or frequency and power level changes controlled by spectrum groups.

Sample Display

The following is sample output from the **debug cable specmgmt** command:

```
Router: debug cable specmgmt  
cmts_next_frequency(0x60A979AC, 1, 1)
```

The following is sample output when the frequency hop was commanded:

```
add_interface_to_freq(0x60BD3734, 0x60C44F68)
```

The following is sample output when the interface was added to a frequency's interface list:

```
set_upstream(0x60A979AC, 1, 21000000, -5)
```

The following is sample output when the spectrum management has set an upstream port's frequency and power level:

```
cmts_frequency_hop_decision(0x60B57FEC)
```

debug cable startalloc

Use the **debug cable startalloc** EXEC command to debug channel allocations on the HFC network. The **no** form of this command disables debugging output.

[no] debug cable startalloc

Usage Guidelines

This command activates debugging of any channel allocations on the HFC network. When this command is activated, any messages generated when channels are allocated to cable modems on the HFC network will be displayed on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable startalloc** command:

```
Router: debug cable startalloc  
MAP startalloc adjusted by <n> mslots
```

This output indicates time-slot MAP processing is active.

debug cable ucc

Use the **debug cable ucc** EXEC command to debug upstream channel change (UCC) messages generated when cable modems request or are assigned a new channel. The **no** form of this command disables debugging output.

[no] debug cable ucc

Usage Guidelines

This command activates debugging of any UCC messages generated when cable modems request or are assigned a new channel. When this command is activated, any messages related to upstream channel changes will be displayed on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable ucc** command when moving a modem from one upstream channel to another:

```
Router: debug cable ucc
SID 2 has been registered

Mac Address of CM for UCC
    00:0E:1D:D8:52:16

UCC Message Sent to CM

Changing SID 2 from upstream channel 1 to upstream channel 2
```

debug cable ucd

Use the **debug cable ucd** EXEC command to debug upstream channel descriptor (UCD) messages. The **no** form of this command disables debugging output.

[no] debug cable ucd

Usage Guidelines

This command activates debugging of any UCD messages. UCD messages contain information about upstream channel characteristics and are sent to the cable modems on the HFC network. Cable modems that are configured to use enhanced upstream channels use these UCD messages to identify and select an enhanced upstream channel to use. When this command is activated, any messages related to upstream channel descriptors will be displayed on the Cisco uBR7246 console.

Sample Display

The following is sample output from the **debug cable ucd** command:

```
Router: debug cable ucd
UCD MESSAGE
-----
FRAME HEADER
  FC                - 0xC2 ==
  MAC_PARM          - 0x00
  LEN               - 0xD3
MAC MANAGEMENT MESSAGE HEADER
  DA                - 01E0.2F00.0001
  SA                - 0009.0CEF.3730
  msg LEN           - C1
  DSAP              - 0
  SSAP              - 0
  control           - 03
  version           - 01
  type              - 02 ==
US Channel ID      - 1
Configuration Change Count - 5
Mini-Slot Size     - 4
DS Channel ID      - 1
Symbol Rate        - 8
Frequency          - 10000000
Preamble Pattern   - CC CC CC CC CC CC CC CC CC CC CC CC CC CC
CC 0D 0D
Burst Descriptor 0
  Interval Usage Code - 1
  Modulation Type     - 1 == QPSK
  Differential Encoding - 2 == OFF
  Preamble Length     - 64
  Preamble Value Offset - 56
  FEC Error Correction - 0
  FEC Codeword Length - 16
  Scrambler Seed      - 0x0152
  Maximum Burst Size  - 2
  Guard Time Size     - 8
  Last Codeword Length - 1 == FIXED
  Scrambler on/off    - 1 == ON
Burst Descriptor 1
  Interval Usage Code - 3
  Modulation Type     - 1 == QPSK
  Differential Encoding - 2 == OFF
  Preamble Length     - 128
  Preamble Value Offset - 0
```

```

FEC Error Correction      - 5
FEC Codeword Length      - 34
Scrambler Seed           - 0x0152
Maximum Burst Size       - 0
Guard Time Size          - 48
Last Codeword Length     - 1 == FIXED
Scrambler on/off        - 1 == ON
Burst Descriptor 2
Interval Usage Code      - 4
Modulation Type          - 1 == QPSK
Differential Encoding     - 2 == OFF
Preamble Length          - 128
Preamble Value Offset    - 0
FEC Error Correction      - 5
FEC Codeword Length      - 34
Scrambler Seed           - 0x0152
Maximum Burst Size       - 0
Guard Time Size          - 48
Last Codeword Length     - 1 == FIXED
Scrambler on/off        - 1 == ON
Burst Descriptor 3
Interval Usage Code      - 5
Modulation Type          - 1 == QPSK
Differential Encoding     - 2 == OFF
Preamble Length          - 72
Preamble Value Offset    - 48
FEC Error Correction      - 5
FEC Codeword Length      - 75
Scrambler Seed           - 0x0152
Maximum Burst Size       - 0
Guard Time Size          - 8
Last Codeword Length     - 1 == FIXED
Scrambler on/off        - 1 == ON

```

```

The UCD MESSAGE is :
0xC2 0x00 0x00 0xD3 0x00 0x00 0x01 0xE0
0x2F 0x00 0x00 0x01 0x00 0x09 0x0C 0xEF
0x37 0x30 0x00 0xC1 0x00 0x00 0x03 0x01
0x02 0x00 0x01 0x05 0x04 0x01 0x01 0x01
0x08 0x02 0x04 0x00 0x98 0x96 0x80 0x03
0x10 0xCC 0xCC 0xCC 0xCC 0xCC 0xCC 0xCC
0xCC 0xCC 0xCC 0xCC 0xCC 0xCC 0xCC 0xD
0x0D 0x04 0x25 0x01 0x01 0x01 0x01 0x02
0x01 0x02 0x03 0x02 0x00 0x40 0x04 0x02
0x00 0x38 0x05 0x01 0x00 0x06 0x01 0x10
0x07 0x02 0x01 0x52 0x08 0x01 0x02 0x09
0x01 0x08 0x0A 0x01 0x01 0x0B 0x01 0x01
0x04 0x25 0x03 0x01 0x01 0x01 0x02 0x01
0x02 0x03 0x02 0x00 0x80 0x04 0x02 0x00
0x00 0x05 0x01 0x05 0x06 0x01 0x22 0x07
0x02 0x01 0x52 0x08 0x01 0x00 0x09 0x01
0x30 0x0A 0x01 0x01 0x0B 0x01 0x01 0x04
0x25 0x04 0x01 0x01 0x01 0x02 0x01 0x02
0x03 0x02 0x00 0x80 0x04 0x02 0x00 0x00
0x05 0x01 0x05 0x06 0x01 0x22 0x07 0x02
0x01 0x52 0x08 0x01 0x00 0x09 0x01 0x30
0x0A 0x01 0x01 0x0B 0x01 0x01 0x04 0x25
0x05 0x01 0x01 0x01 0x02 0x01 0x02 0x03
0x02 0x00 0x48 0x04 0x02 0x00 0x30 0x05
0x01 0x05 0x06 0x01 0x4B 0x07 0x02 0x01
0x52 0x08 0x01 0x00 0x09 0x01 0x08 0x0A
0x01 0x01 0x0B 0x01 0x01

```

debug callback

Use the **debug callback EXEC** command to display callback events when the router is using a modem and a chat script to call back on a terminal line. The **no** form of this command disables debugging output.

[no] debug callback

Usage Guidelines

This command is useful for debugging chat scripts on PPP and ARAP lines that use callback mechanisms. The output provided by the **debug callback** command shows you how the call is progressing when used with the **debug ppp** or **debug arap** commands.

Sample Display

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

```
Router# debug callback

TTY7 Callback process initiated, user: exec_test dialstring 123456
TTY7 Callback forced wait = 4 seconds
TTY7 Exec Callback Successful - await exec/autoselect pickup
TTY7: Callback in effect
```

Related Commands

debug arap
debug ppp

debug cdp

Use the **debug cdp** EXEC command to enable debugging of Cisco Discovery Protocol (CDP). The **no** form of this command disables debugging output.

[no] debug cdp {packets | adjacency | events}

Syntax Description

packets	Enables packet-related debugging output.
adjacency	Enables adjacency-related debugging output.
events	Enables output related to error messages, such as detecting a bad checksum.

Usage Guidelines

Use **debug cdp** commands to display information about CDP packet activity, activity between CDP neighbors, and various CDP events.

Sample Display

The following is sample output from **debug cdp packets**, **debug cdp adjacency**, and **debug cdp events** commands:

```
Router# debug cdp packets
CDP packet info debugging is on
Router# debug cdp adjacency
CDP neighbor info debugging is on
Router# debug cdp events
CDP events debugging is on

CDP-PA: Packet sent out on Ethernet0
CDP-PA: Packet received from gray.cisco.com on interface Ethernet0

CDP-AD: Deleted table entry for violet.cisco.com, interface Ethernet0
CDP-AD: Interface Ethernet2 coming up

CDP-EV: Encapsulation on interface Serial2 failed
```

debug cdp ip

Use the **debug cdp ip** EXEC command to enable debug output for the IP routing information that is carried and processed by the Cisco Discovery Protocol (CDP). The **no** form of this command disables debugging output.

[no] debug cdp ip

Usage Guidelines

CDP is a media- and protocol-independent device-discovery protocol that runs on all Cisco routers.

You can use the **debug cdp ip** command to determine the IP network prefixes CDP is advertising and whether CDP is correctly receiving this information from neighboring routers.

Use the **debug cdp ip** command with the **debug ip routing** command to debug problems that occur when on-demand routing (ODR) routes are not installed in the routing table at a hub router. You can also use the **debug cdp ip** command with the **debug cdp packet** and **debug cdp adjacency** commands along with encapsulation-specific debug commands to debug problems that occur in the receipt of CDP IP information.

Sample Display

The following is sample output from the **debug cdp ip** command. This example shows the transmission of IP-specific information in a CDP update. In this case, three network prefixes are being transmitted, each with a different network mask.

```
Router# debug cdp ip

CDP-IP: Writing prefix 172.1.69.232.112/28
CDP-IP: Writing prefix 172.19.89.0/24
CDP-IP: Writing prefix 11.0.0.0/8
```

In addition to these messages, you might see the following messages:

- This message indicates that CDP is attempting to install the prefix 172.1.1.0/24 into the IP routing table:

```
CDP-IP: Updating prefix 172.1.1.0/24 in routing table
```
- This message indicates a protocol error occurred during an attempt to decode an incoming CDP packet:

```
CDP-IP: IP TLV length (3) invalid
```
- This message indicates the receipt of the IP prefix 172.1.1.0/24 from a CDP neighbor connected via the Ethernet interface 0/0. The neighbor's IP address is 10.0.0.1.

```
CDP-IP: Reading prefix 172.1.1.0/24 source 10.0.0.1 via Ethernet0/0
```

Related Commands

debug cdp adjacency
debug cdp packet
debug ip routing

debug channel events

The **debug channel events** EXEC command displays processing events that occur on the channel adapter interfaces of all installed adapters. This command is valid for the Cisco 7000 series routers only. The **no** form of this command disables debugging output.

[no] debug channel events

Usage Guidelines

This command displays Channel Interface Processor (CIP) events that occur on the CIP interface processor and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the **debug channel events** command does not return any information. If the command generates numerous messages, they can indicate the possible source of the problems. To observe the statistic message (cip_love_letter) transmitted every ten seconds, use the **debug channel love** command.

When configuring or making changes to a router or interface that supports IBM channel attach, enable **debug channel events**. Doing so alerts you to the progress of the changes or to any errors that might result. Also use this command periodically when you suspect network problems.

Sample Display

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

```
Router# debug channel events

Channel3/0: cip_reset(), state administratively down
Channel3/0: cip_reset(), state up
Channel3/0: sending nodeid
Channel3/0: sending command for vc 0, CLAW path C700, device C0
```

The following line indicates that the CIP is being reset to an administrative down state:

```
Channel3/0: cip_reset(), state administratively down
```

The following line indicates that the CIP is being reset to an administrative up state:

```
Channel3/0: cip_reset(), state up
```

The following line indicates that the node id is being sent to the CIP. This information is the same as the “Local Node” information under the **show extended channel slot/port subchannels** command. The CIP needs this information to send to the host mainframe.

```
Channel3/0: sending nodeid
```

The following line indicates that a CLAW subchannel command is being sent from the RP to the CIP. The value vc 0 indicates that the CIP will use virtual circuit number 0 with this device. The virtual circuit number will also show up when you use the **debug channel packets** command.

```
Channel3/0: sending command for vc 0, CLAW path C700, device C0
```

Related Commands

debug channel love
debug channel packets

debug channel love

Use the **debug channel love** EXEC command to display Channel Interface Processor (CIP) love letter events. This command is valid for the Cisco 7000 series routers only. The **no** form of this command disables debugging output.

[no] debug channel love

Usage Guidelines

This command displays Channel Interface Processor (CIP) events that occur on the CIP interface processor and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the **debug channel love** command returns a statistic message (cip_love_letter) that is transmitted every ten seconds.

Sample Display

The following is sample output from the **debug channel love** command:

```
Router# debug channel love

Channel3/1: love letter received, bytes 3308
Channel3/0: love letter received, bytes 3336
cip_love_letter: received 11, but no cip_info
```

The following line indicates that data was received on the CIP:

```
Channel3/1: love letter received, bytes 3308
```

The following line indicates that the interface is enabled, but there is no configuration for it. It does not normally indicate a problem, just that the route processor (RP) got statistics from the CIP but has no place to store them.

```
cip_love_letter: received 11, but no cip_info
```

Related Commands

debug channel events
debug channel packets

debug channel packets

Use the **debug channel packets** EXEC command to display per-packet debugging output. The output reports information when a packet is received or a transmit is attempted. The **no** form of this command disables debugging output.

[no] debug channel packets

Usage Guidelines

The **debug channel packets** command displays all process-level Channel Interface Processor (CIP) packets for both outbound and inbound packets. You will need to disable fast switching and autonomous switching to obtain debugging output. This command is useful for determining whether packets are received or transmitted correctly.

This command is valid for the Cisco 7000 series routers only.

Sample Display

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

```
Router# debug channel packets

(Channel3/0)-out size = 104, vc = 0000, type = 0800, src 172.24.0.11, dst 172.24.1.58
(Channel3/0)-in size = 48, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
(Channel3/0)-in size = 48, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
(Channel3/0)-out size = 71, vc = 0000, type = 0800, src 172.24.15.197, dst 172.24.1.58
(Channel3/0)-in size = 44, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
```

Table 25 describes the fields in the output.

Table 25 Debug Channel Packets Field Descriptions

Field	Description
(Channel3/0)	The interface slot and port.
in / out	In is a packet from the mainframe to the router. Out is a packet from the router to the mainframe.
size =	The number of bytes in the packet, including internal overhead.
vc =	A value from 0–511 that maps to the claw interface configuration command. This information is from the MAC layer.
type =	The encapsulation type in the MAC layer. The value 0800 indicates an IP datagram.
src	The origin, or source, of the packet, as opposed to the previous hop address.
dst	The destination of the packet, as opposed to the next hop address.

Related Commands

debug channel events

debug channel love