debug arap

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

debug arap {**internal** | **memory** | **mnp4** | **v42bis**} [*linenum* [**aux** | **console** | **tty** | **vty**]]

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.42*bis* compression.
linenum	(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 dialin and callback events.

Use the **debug modem** command to help catch problems related to ARAP autodetection (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.

Examples

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

```
Router# debug arap internal

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

Related Commands

Command	Description
debug callback	Displays callback events when the router is using a modem and a chat script to call back on a terminal line.
debug modem	Observes modem line activity on an access server.

debug arp

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

debug arp

no debug arp

Syntax Description

This command has no arguments or keywords.

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 ARP packets and whether it is receiving ARP packets.

Examples

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 a bridge is being run 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

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

debug asp packet

no debug asp packet

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The router uses asynchronous security protocols from companies including 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 pass-through mode using BSTUN encapsulation.

Examples

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 EIA/TIA 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 datascope 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 22 describes the significant fields in the display.

Table 22 debug asp Packet Descriptions

Field	Description
ASP	Asyncronous security protocol packet.
Serial5	Interface receiving and sending the packet.
ADI-Rx	Packet is being received.
ADI-T	Packet is being sent.
Data (*n* bytes)	Type and size of the packet.
F1FF4c42	Alarm panel device address.

debug async async-queue

To display debug messages for asynchronous rotary line queueing, use the **debug async async-queue** command in privileged EXEC mode.

debug async async-queue

Syntax Description

This command has no arguments or keywords.

Defaults

This command has no default settings.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following example starts the asynchronous rotary line queueing debugging display:

```
Router# debug async async-queue

*Mar  2 03:50:28.377: AsyncQ: First connection to be queued - starting the AsyncQ manager
*Mar  2 03:50:28.377: AsyncQ: Enabling the AsyncQ manager
*Mar  2 03:50:28.377: AsyncQ: Started the AsyncQ manager process with pid 98
*Mar  2 03:50:28.381: AsyncQ: Created a Waiting TTY on TTY66 with pid 99
*Mar  2 03:50:30.164: WaitingTTY66: Did Authentication on waiting TTY (VTY)
*Mar  2 03:50:30.168: AsyncQ: Received ASYNCQ_MSG_ADD
*Mar  2 03:50:30.168: AsyncQ: New queue, adding this connection as the first element
*Mar  2 03:50:34.920: AsyncQ: Created a Waiting TTY on TTY67 with pid 100
*Mar  2 03:50:36.783: WaitingTTY67: Did Authentication on waiting TTY (VTY)
*Mar  2 03:50:36.787: AsyncQ: Received ASYNCQ_MSG_ADD
*Mar  2 03:50:36.787: AsyncQ: Queue exists, adding this connection to the end of the queue
```

Related Commands

Command	Description
debug ip tcp transactions	Enables the IP TCP transactions debugging display to observe significant transactions such as state changes, retransmissions, and duplicate packets.
debug modem	Enables the modem debugging display to observe modem line activity on an access server.

debug atm state

To display the states for Asynchronous Transfer Mode (ATM) common connections on the networking device, use the **debug atm state** command in privileged EXEC mode. To disable the display of debugging output, use the **no** form of this command.

debug atm state

no debug atm state

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3	This command was introduced.

Examples

The following example shuts the interface down and displays the debugging messages in regard to the ATM interface on the networking device:

```
Router# debug atm state

ATM VC States debugging is on

Router# show debug

Generic ATM:
  ATM VC States debugging is on

Router# configure terminal
Router(config)# interface atm 2/0.2
Router(config-if)# shutdown

*Aug  8 17:45:38.987: Changing vc 3/100vc-state to ATM_VC_SHUTTING_DOWN
*Aug  8 17:45:38.991: Changing vc 3/100vc-state to ATM_VC_NOT_IN_SERVICE
```

The following example turns the interface back on and displays the debugging messages in regard to the ATM interface on the networking device:

```
Router(config)# interface atm 2/0.2
Router(config-if)# no shutdown

*Aug  8 17:45:44.711: Changing vc 3/100 vc-state to ATM_VC_ESTABLISHING_VC
*Aug  8 17:45:44.711: Changing vc 3/100 vc-state to ATM_VC_NOT_VERIFIED
*Aug  8 17:45:44.711: Changing vc 3/100 vc-state to ATM_VC_UP
```

Related Commands

Command	Description
debug atm ha-state	Displays the ATM HA state on the networking device.

debug backhaul-session-manager set

To trace state changes and receive messages and events for all the available session sets or a specified session set, use the **debug backhaul-session-manager set** privileged EXEC command.

debug backhaul-session-manager set {**all** | **name** *set-name*}

Syntax Description

all	All available session sets.
name *set-name*	Specified session set.

Defaults

Debugging for backhaul session sets is not enabled.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following is output for the **debug backhaul-session-manager set all** command:

```
Router# debug backhaul-session-manager set all
Router# debug_bsm_command:DEBUG_BSM_SET_ALL

 Function set_proc_event() is called
Session-Set :test-set
Old State   :BSM_SET_OOS
New State   :BSM_SET_OOS
   Active-Grp  :NONE
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-None
   Event rcvd  :EVT_GRP_INS

BSM:Event BSM_SET_UP is sent to user
Session-Set :test-set
Old State   :BSM_SET_OOS
New State   :BSM_SET_ACTIVE_IS
   Active-Grp  :g-11
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-Active
   Event rcvd  :BSM_ACTIVE_TYPE
```

The following is output for the **debug backhaul-session-manager set all name test-set** command:

```
Router# debug backhaul-session-manager set name test-set
Router# debug_bsm_command:DEBUG_BSM_SET_NAME

Nomad-B# Function set_proc_event() is called
Session-Set :test-set
Old State   :BSM_SET_OOS
```

```
New State   :BSM_SET_OOS
   Active-Grp  :NONE
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-None
   Event rcvd  :EVT_GRP_INS

Nomad-B#BSM:Event BSM_SET_UP is sent to user
Session-Set :test-set
Old State   :BSM_SET_OOS
New State   :BSM_SET_ACTIVE_IS
   Active-Grp  :g-11
   Session-Grp :g-11
   Old State   :Group-None
   New State   :Group-Active
   Event rcvd  :BSM_ACTIVE_TYPE
```

Related Commands

Command	Description
debug backhaul-session-manager session	Displays debug information for all available sessions or a specific session.

debug backhaul-session-manager session

To debug all the available sessions or a specified session, use the **debug backhaul-session-manager session** privileged EXEC command.

debug backhaul-session-manager session {**show** | **state** | **xport**} {**all** | *session-id*}

Syntax Description

show	Displays session manager states and statistics. **Note** This command only displays information about the specified session once, and does not enable debugging.
state	Shows information about state transitions. Possible states are: • SESS_SET_IDLE: A session-set has been created. • SESS_SET_OOS: Sessions have been added to session groups. No ACTIVE notification has been received from VSC. • SESS_SET_ACTIVE_IS: An ACTIVE notification has been received over one in-service session group. STANDBY notification has not been received on any available session groups. • SESS_SET_STNDBY_IS: A STANDBY notification is received, but there is no in-service active session group available. • SESS_SET_FULL_IS: A session group in-service has ACTIVE notification and at least one session group in-service has STANDBY notification. • SESS_SET_SWITCH_OVER: An ACTIVE notification is received on a session group in-service, which had received STANDBY notification.
xport	Provides traces for all PDUs (packets), application PDUs, and session manager messages. **Note** Use caution while enabling this debug command in a live system.
all	All available session sets.
session-id	Specified session.

Defaults

Debugging for backhaul session-session is not enabled.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

The following is output for the **debug backhaul-session-manager session all** command:

```
Router# debug backhaul-session-manager session show all

Router# debug_bsm_command:DEBUG_BSM_SESSION_SHOW

23:43:34:Session information --
  Group:g-11
Configuration:
     Local:172.18.72.198  , port:5555
    Remote:161.44.2.72    , port:5555
  Id:33,  Priority:1
  RUDP Option:Client, Conn Id:0x80BA14EC
State:
  Status:OPEN, Use-status:IS,
Statistics:
  # of resets:68
  Receive Total pkts:7, failures:0
  Transmit Total pkts:69, failures:0, blocked:0
  group-ptr:0x80B17E18, tmrid:0x8094D658, debug-mask:0x0

23:43:34:Session information --
  Group:g-12
Configuration:
     Local:172.18.72.198  , port:5575
    Remote:161.44.2.72    , port:5575
  Id:34,  Priority:1
  RUDP Option:Client, Conn Id:0x80BA12FC
State:
  Status:OPEN_WAIT, Use-status:OOS,
Statistics:
  # of resets:88
  Receive Total pkts:8, failures:0
  Transmit Total pkts:88, failures:0, blocked:0
  group-ptr:0x80B17ED0, tmrid:0x8094D678, debug-mask:0x0

Router# debug backhaul-session-manager session show 33

Router# debug_bsm_command:DEBUG_BSM_SESSION_SHOW

23:48:32:Session information --
  Group:g-11
Configuration:
     Local:172.18.72.198  , port:5555
    Remote:161.44.2.72    , port:5555
  Id:33,  Priority:1
  RUDP Option:Client, Conn Id:0x80BA14EC
State:
  Status:OPEN, Use-status:IS,
Statistics:
  # of resets:68
  Receive Total pkts:7, failures:0
  Transmit Total pkts:69, failures:0, blocked:0
  group-ptr:0x80B17E18, tmrid:0x8094D658, debug-mask:0x0

Router# debug backhaul-session-manager session all

Router# debug_bsm_command:DEBUG_BSM_SESSION_ALL
```

```
23:49:14:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:14:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:14:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:14:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:14:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:19:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:19:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:19:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:19:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:19:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:24:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:24:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:24:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:24:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:24:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:29:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:29:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:29:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:29:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:29:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:34:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:49:34:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:49:34:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:34:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:49:34:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:49:34:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 1 (CONN-FAILED)

23:49:34:SESSION:STATE:(33) old-state:OPEN, new-state:CLOSE_WAIT


Router# debug backhaul-session-manager session state all

Router# debug_bsm_command:DEBUG_BSM_SESSION_STATE_ALL

23:50:54:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:CLOSE
23:50:54:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS

23:50:54:SESSION:STATE:(34) old-state:OPEN_WAIT, new-state:OPEN_WAIT
23:50:54:SESSION:STATE:(34) state:OPEN_WAIT, use-state:OOS
```

```
Router# debug backhaul-session-manager session xport all

Router# debug_bsm_command:DEBUG_BSM_SESSION_XPORT
23:51:39:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)

23:51:42:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 5 (CONN-RESET)

23:51:44:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
```

Related Commands

Command	Description
debug atm state	Traces state changes and receives messages and events for all available session sets or a specified session set.

debug bert

To display information on the bit error rate testing (BERT) feature, use the **debug bert** privileged EXEC command. The **no** form of this command disables the debugging output.

debug bert

no debug bert

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(2)XD	This command was introduced.

Usage Guidelines

The **debug bert** command output is used primarily by Cisco technical support representatives. The **debug bert** command displays debugging messages for specific areas of executed code.

Examples

The following is output from the **debug bert** command:

```
Router# debug bert

Bit Error Rate Testing debugging is on

Router# no debug bert

Bit Error Rate Testing debugging is off
```

Related Commands

Command	Description
bert abort	Aborts a bit error rate testing session.
bert controller	Starts a bit error rate test for a particular port on a Cisco AS5300 router.
bert profile	Sets up various bit error rate testing profiles.

debug bri-interface

To display debugging information on ISDN BRI routing activity, use the **debug bri-interface** privileged EXEC command. The **no** form of this command disables debugging output.

debug bri-interface

no debug bri-interface

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **debug bri-interface** 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.

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

Examples

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

```
Router# debug bri-interface

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

Command	Description
debug isdn event	Displays ISDN events occurring on the user side (on the router) of the ISDN interface.
debug isdn q921	Displays data link-layer (Layer 2) access procedures that are taking place at the router on the D channel (LSPD).
debug isdn q931	Displays information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network.

debug bsc event

To display all events occurring in the Binary Synchronous Communications (Bisync) feature, use the **debug bsc event** privileged EXEC command. The **no** form of this command disables debugging output.

debug bsc event [*number*]

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.

Examples

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_EOFile
BSC: Serial2         POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
BSC: Serial2         POLLEE-FSM inp:E_LineFail old_st:CU_Down new_st:TCU_EOFile
0:04:32: BSC: Serial2 :SDI-rx: 9 bytes
BSC: Serial2         POLLEE-FSM inp:E_RxEtx old_st:CU_Down new_st:TCU_EOFile
0:04:32: BSC: Serial2 :SDI-rx: 5 bytes
BSC: Serial2         POLLEE-FSM inp:E_RxEnq old_st:CU_Down new_st:TCU_EOFile
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

Command	Description
debug bsc packet	Displays all frames traveling through the Bisync feature.
debug bstun events	Displays BSTUN connection events and status.

debug bsc packet

To display all frames traveling through the Binary Synchronous Communications (Bisync) feature, use the **debug bsc packet** privileged EXEC command. The **no** form of this command disables debugging output.

debug bsc packet [**group** *number*] [**buffer-size** *bytes*]

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).

Defaults

The default number of bytes displayed is 20.

Usage Guidelines

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

Examples

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

Command	Description
debug bsc event	Displays all events occurring in the Bisync feature.
debug bstun events	Displays BSTUN connection events and status.

debug bstun events

To display BSTUN connection events and status, use the **debug bstun events** privileged EXEC command. The **no** form of this command disables debugging output.

debug bstun events [*number*]

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 Bisync protocol level.

Examples

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

Command	Description
debug bsc event	Displays all events occurring in the Bisync feature.
debug bsc packet	Displays all frames traveling through the Bisync feature.
debug bundle errors	Displays packet information on packets traveling through the BSTUN links.

debug bundle errors

To enable the display of information on bundle errors, use the **debug bundle errors** privileged EXEC command.

debug bundle errors

no debug bundle errors

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)T	This command was introduced.

Usage Guidelines

Use this command to enable the display of error information for a bundle, such as reports of inconsistent mapping in the bundle.

Related Commands

Command	Description
bump	Configures the bumping rules for a VC class that can be assigned to a VC bundle.
bundle	Creates a bundle or modifies an existing bundle to enter bundle configuration mode.
debug bundle events	Enables display of bundle events when use occurs.

debug bundle events

To enable display of bundle events when use occurs, use the **debug bundle events** privileged EXEC command in debug mode.

debug bundle events

no debug bundle events

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)T	This command was introduced.

Usage Guidelines

Use this command to enable the display of bundle events, such as occurrences of VC bumping, when bundles were brought up, when they were taken down, and so forth.

Related Commands

Command	Description
debug bundle errors	Enables the display of information on bundle errors.

debug bstun packet

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

debug bstun packet [**group** *number*] [**buffer-size** *bytes*]

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).

Defaults

The default number of bytes displayed is 20.

Examples

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: 02C1C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:00 Serial2         SDI: Addr: 40 Data: 02C1C1C1C1C1C1C1C1C1
BSTUN bsc-local-ack: 0:00:06 Serial2         NDI: Addr: 40 Data: 0227F5C31140C11D60C8
```

Related Commands

Command	Description
debug bstun events	Displays BSTUN connection events and status.

debug cable env

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

debug cable env

no debug cable env

Syntax Description

This command has no arguments or keywords.

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.

Examples

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 23 describes the significant fields in the display.

Table 23 debug cable env Field Descriptions

Field	Description
ps id	Power supply raw voltage reading.
pstype	Power supply type determined from the ps id, v, and r values. The Cisco uBR7246 universal broadband router 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.

debug cable err

To display errors that occur in the cable MAC protocols, use the **debug cable err** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable err

no debug cable err

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

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

Examples

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 freqhop

To display debug messages for frequency hopping, use the **debug cable freqhop** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cable freqhop

no debug cable freqhop

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for frequency hopping is not enabled.

Command History

Release	Modification
12.0(4)XI	This command was introduced.

Examples

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

```
Router# debug cable freqhop

CMTS freqhop debugging is on
```

Related Commands

Command	Description
debug cable hw-spectrum	Displays debug information about spectrum management (frequency agility).
debug cable freqhop	Displays debug information about frequency hopping, which is a facet of spectrum management.

debug cable hw-spectrum

To display debug messages for spectrum management (frequency agility), use the **debug cable hw-spectrum** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cable hw-spectrum

no debug cable hw-spectrum

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for spectrum management is not enabled.

Command History

Release	Modification
12.0	This command was introduced as **debug cable specmgmt**.
12.0(4)XI	This command was renamed as **debug cable hw-spectrum**.

Examples

The following is sample output for the **debug cable hw-spectrum** command:

```
Router# debug cable hw-spectrum

CMTS specmgmt debugging is on
```

debug cable interface

To perform debugging on a specified interface, use the **debug cable interface** privileged EXEC command. To turn off debugging on a specified interface, use the **no** form of this command.

debug cable interface *interface* [**mac-address** *address* | **mask** | **verbose**]

no debug cable interface *interface* **mac-address** *address*

Syntax Description

interface	Specifies the cable interface to be debugged.
mac-address	(Optional) Specifies that debugging is to be done on a specified MAC address.
address	(Optional) Specifies the MAC address of the interface.
mask	(Optional) Specifies the MAC address validation address.
verbose	(Optional) Displays detailed debug information.

Command History

Release	Modification
12.0(6)T	This command was introduced.

Usage Guidelines

You can repeat this debug command for other interfaces. Each time you specify a different cable interface or MAC address, debugging is turned on for this cable interface or MAC address.

If you enter two debug commands with the same interface or MAC address, but with different mask or verbose keywords, the router treats both commands as the same. In this case, the latest debug information supersedes the previous debugging information.

Examples

The following example demonstrates how to enable debugging on interface c3/0:

```
Router# debug cable interface c3/0
```

The following example demonstrates how to enable detailed debugging on interface c3/0:

```
Router# debug cable interface c3/0 verbose
```

The following example demonstrates how to enable debugging on interface c3/0 for all traffic coming from modems with MAC addresses 0010.00xx.xxxx:

```
Router# debug cable interface c3/0 mac-address 0010.0000.0000 ffff.ff00.0000
```

Related Commands

Command	Description
debug cable mac-address	Enables debugging on traffic from modems with the specified MAC address or MAC address range.

debug cable keyman

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

debug cable keyman

no debug cable keyman

Syntax Description

This command has no arguments or keywords.

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.

Examples

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 = %llx, readback = %llx
     Verify iv failed with SID %2x : setvalue = %llx, readback = %llx
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]  : %llx\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]  : %llx\tKey Life[1]: %u sec\n",
          sidkey->key_status[1], sidkey->des_key213
1].iv,
          compute_remain_lifetime(&sidkey->des_key[1]));
```

debug cable mac

To display MAC-layer information for the specified cable modem, use the **debug cable mac** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable mac

no debug cable mac

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Caution Do not use this command if you have a large number of modems on your network. The Cisco uBR7246 universal broadband router will become flooded with console printouts.

Examples

The following example shows the return for the MAC layer:

```
Router# debug cable mac

19:46:27: Ranging Modem with Sid 1 on i/f : Cable6/0/U0

19:46:27: Got a ranging request
19:46:27: SID value is 1 on Interface Cable6/0/U0
19:46:27: CM mac address 00:E0:1E:B2:BB:07
19:46:27: Timing offset is 0
19:46:27: Power value is FE0, or 0 dB
19:46:27: Freq Error = 0, Freq offset is 0
19:46:27: Ranging has been successful for SID 1 on Interface Cable6/0/U0

19:46:29: Ranging Modem with Sid 2 on i/f : Cable6/0/U0
19:46:29: Got a ranging request
19:46:29: SID value is 2 on Interface Cable6/0/U0
19:46:29: CM mac address 00:E0:1E:B2:BB:8F
19:46:29: Timing offset is 1
19:46:29: Power value is 1350, or 0 dB
19:46:29: Freq Error = 0, Freq offset is 0
19:46:29: Ranging has been successful for SID 2 on Interface Cable6/0/U0

19:46:32: Ranging Modem with Sid 3 on i/f : Cable6/0/U0

19:46:32: Got a ranging request
19:46:32: SID value is 3 on Interface Cable6/0/U0
19:46:32: CM mac address 00:E0:1E:B2:BB:B1
19:46:32: Timing offset is FFFFFFFF
19:46:32: Power value is 1890, or -1 dB
19:46:32: Freq Error = 0, Freq offset is 0
19:46:32: Ranging has been successful for SID 3 on Interface Cable6/0/U0

19:46:34: Ranging Modem with Sid 5 on i/f : Cable6/0/U0
```

Table 24 describes the significant fields in the display.

Table 24 debug cable mac Field Descriptions

Field	Description
SID value is....	Reports the service ID of the modem. The range is from 1 through 891. The information on this line should agree with the first line of the return (that is, Ranging Modem with Sid...).
CM mac address....	MAC address of the specified cable modem.
Timing offset is....	Time by which to offset the frame transmission upstream so the frame arrives at the expected minislot time at the CMTS.
Power value is FE0, or 0 dB	Raw value derived from the 3137 Broadcom chip. Alternately, the decibel value specifies the relative change in the transmission power level that the cable modem needs to make so transmissions arrive at the CMTS at the desired power level. This desired power level is usually 0, but you can use the CLI to change it via the **cable power-level** command.
Freq Error = ...	Raw value derived from the 3137 Broadcom chip.
Freq offset is	Specifies the relative change in the transmission frequency that the cable modem will make to match the CMTS.

Related Commands

Command	Description
show controllers cable	Displays interface controller information for the specified slot.

debug cable mac-address

To enable debugging for a specified MAC address, use the **debug cable mac-address** privileged EXEC command. To turn off debugging for the specified MAC address, use the **no** form of this command.

debug cable mac-address *address* [**mask** | **verbose**]

no debug cable mac-address *address*

Syntax Description

address	Specifies the MAC address of the interface.
mask	(Optional) Specifies the MAC address validation address.
verbose	(Optional) Displays detailed debug information.

Command History

Release	Modification
12.0(6)T	This command was introduced.

Usage Guidelines

You can repeat this debug command for other MAC addresses. Each time you specify a different MAC address, debugging is turned on for this MAC address.

If you enter two debug commands with the same MAC address, but with different mask or verbose keywords, the router treats both commands as the same. In this case, the latest debug information supersedes the previous debugging information.

Examples

The following example demonstrates how to enable debugging for all traffic coming from all interfaces of modems with the MAC address 0010.00xx.xxxx:

```
Router# debug cable mac-address 0010.0000 ffff.ff00.000
```

Related Commands

Command	Description
debug cable interface	Enables debugging on the cable interface specified.

debug cable map

To display map debugging messages, use the **debug cable map** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable map

no debug cable map

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Examples

The following example displays all the map messages with and without data grants:

```
Router# debug cable map

19:41:53: On interface Cable6/0, sent 5000 MAPs, 1321 MAPs had grant(s)Long Grants
13256993, Total Short Grants 223
A sample Map without any data grant
------------------ MAP MSG  --------------------
us_ch_id: 1   ucd_count: 5  num_elems: 9  reserved: 0
Alloc Start Time: 33792        Ack Time: 33618
Rng_bkoff_start: 0    Rng_bkoff_end: 2
Data_bkoff_start: 1   Data_bkoff_end: 3:
sid:16383   iuc:1   mslot_offset:0
sid:0   iuc:7   mslot_offset:40
A sample Map with data grant(s)
------------------ MAP MSG  ----------------------
us_ch_id: 1   ucd_count: 5  num_elems: 7  reserved: 0
Alloc Start Time: 33712        Ack Time: 33578
Rng_bkoff_start: 0    Rng_bkoff_end: 2
Data_bkoff_start: 1   Data_bkoff_end: 3
sid:2   iuc:6   mslot_offset:0
sid:16383   iuc:1   mslot_offset:16
sid:0   iuc:7   mslot_offset:40
```

Table 25 shows the significant fields in the display.

Table 25 debug cable map Field Descriptions

Field	Description
sent 5000 MAPs	Total number of maps sent.
MAPs had grant(s) Long Grants	Total number of grants considered long sized by the CMTS.
Total Short Grants	Total number of grants considered short sized by the CMTS.
us_ch_id	Identifies the upstream channel ID for this message.
ucd_count	Number of upstream channel descriptors (UCDs).
num_elems	Number of information elements in the map.
reserved	Reserved for alignment.

Table 25 debug cable map Field Descriptions (continued)

Field	Description
Alloc Start Time	Start time from CMTS initialization (in minislots) for assignments in this map.
Ack Time	Latest time from CMTS initialization (in minislots) processed in upstream. The cable modems use this time for collision detection.
Rng_bkoff_start	Initial backoff window for initial ranging contention, expressed as a power of 2. Valid values are from 0 to 15.
Rng_bkoff_end	Final backoff window for initial ranging contention, expressed as a power of 2. Valid values are from 0 to 15.
Data_bkoff_start	Initial backoff window for contention data and requests, expressed as a power of 2. Valid values are from 0 to 15.
Data_bkoff_end	Final backoff window for contention data and requests, expressed as a power of 2. Valid values are from 0 to 15.
sid	Service ID.
iuc	Interval usage code (IUC) value.
mslot_offset	Minislot offset.

Related Commands

Command	Description
show controllers cable	Displays interface controller information for the specified slot.

debug cable-modem bpkm

To debug baseline privacy information on a Cisco uBR900 series cable access router, use the **debug cable-modem bpkm** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem bpkm {**errors** | **events** | **packets**}

no debug cable-modem bpkm {**errors** | **events** | **packets**}

Syntax Description

errors	Provides debugging information about Cisco uBR900 series privacy errors.
events	Provides debugging information about events related to cable baseline privacy.
packets	Provides debugging information about baseline privacy packets.

Command History

Release	Modification
11.3 NA	This command was introduced.

Usage Guidelines

Baseline privacy key management exchanges take place only when both the Cisco uBR900 series and the CMTS are running code images that support baseline privacy, and the privacy class of service is enabled via the configuration file that is downloaded to the cable modem. Baseline privacy code images for the Cisco uBR900 series contain “k1” in the code image name.

Examples

The following example shows debug output when the headend does not have privacy enabled:

```
Router# debug cable-modem bpkm errors

cm_bpkm_fsm(): machine: KEK, event/state: EVENT_4_TIMEOUT/STATE_B_AUTH_WAIT, new state:
STATE_B_AUTH_WAIT

cm_bpkm_fsm(): machine: KEK, event/state: EVENT_4_TIMEOUT/STATE_B_AUTH_WAIT, new state:
STATE_B_AUTH_WAIT

%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to down
cm_bpkm_fsm(): machine: KEK, event/state: EVENT_1_PROVISIONED/STATE_A_START, new state:
STATE_B_AUTH_WAIT

%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to up
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series cable access router.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.

Command	Description
debug cable-modem mac	Troubleshoots the Cisco uBR900 series cable access router MAC layer.
debug cable-modem map	Displays the timing from map messages to synchronize messages and the timing between map messages.

debug cable-modem bridge

To debug bridge filter processing information on a Cisco uBR900 series cable access router, use the **debug cable-modem bridge** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem bridge

no debug cable-modem bridge

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Usage Guidelines

When the interface is down, all bridge table entries learned on the Ethernet interface are set to discard because traffic is not bridged until the cable interface has completed initialization. After the interface (the line protocol) is completely up, bridge table entries learned on the Ethernet interface program the cable MAC data filters. The cable MAC hardware filters out any received packets whose addresses are not in the filters. In this way, the cable interface only receives packets addressed to its own MAC address or an address it has learned on the Ethernet interface.

Examples

The following example shows sample display output for the **debug cable-modem bridge** command:

```
Router# debug cable-modem bridge

%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to downshut
cm_tbridge_add_entry(): MAC not initialized, discarding entry: 00e0.fe7a.186fno shut
cm_tbridge_add_entry(): MAC not initialized, discarding entry: 00e0.fe7a.186f
%LINEPROTO-5-UPDOWN: Line protocol on Interface cable-modem0, changed state to up
cm_tbridge_add_entry(): Adding entry 00e0.fe7a.186f to filter 2
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.
debug cable-modem map	Displays the timing from MAP messages to synchronize messages and the timing between MAP messages.

debug cable-modem error

To enable debugging messages for the cable interface driver, use the **debug cable-modem error** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem error

no debug cable-modem error

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Usage Guidelines

This command displays detailed output about the sanity checking of received frame formats, the acquisition of downstream QAM/FEC lock, the receipt or nonreceipt of SYNC messages from the CMTS, reception errors, and bandwidth request failures.

Examples

The following example shows sample display output for the **debug cable-modem error** command:

```
Router# debug cable-modem error

*Mar  7 20:16:29: AcquireSync(): Update rate is 100 Hz
*Mar  7 20:16:30: 1st Sync acquired after 1100 ms.
*Mar  7 20:16:30: Recovery loop is locked (7/9)
*Mar  7 20:16:30: 2nd Sync acquired after 100 ms.
*Mar  7 20:16:30: Recovery loop is locked (10/15)
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.
debug cable-modem map	Displays the timing from MAP messages to sync messages and the timing between MAP messages.

debug cable-modem interrupts

To debug Cisco uBR900 series interrupts, use the **debug cable-modem interrupts** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem interrupts

no debug cable-modem interrupts

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Examples

The following example shows sample debug output for Cisco uBR900 series interrupts:

```
Router# debug cable-modem interrupts

*** BCM3300_rx_mac_msg_interrupt ***
*** BCM3300_rx_mac_msg_interrupt ***
### BCM3300_tx_interrupt ###
*** BCM3300_rx_mac_msg_interrupt ***
### BCM3300_tx_interrupt ###
*** BCM3300_rx_mac_msg_interrupt ***
### BCM3300_tx_interrupt ###
### BCM3300_tx_interrupt ###
### BCM3300_tx_interrupt ###
### BCM3300_tx_interrupt ###
```

Related Commands

Command	Description
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for Cisco uBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.
debug cable-modem map	Displays the timing from MAP messages to sync messages and the timing between MAP messages.

debug cable-modem mac

To troubleshoot the Cisco uBR900 series MAC layer, use the **debug cable-modem mac** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem mac {**log** [**verbose**] | **messages**}

no debug cable-modem mac {**log** [**verbose**] | **messages**}

Syntax Description

log	Displays the real-time MAC log.
verbose	(Optional) Displays periodic MAC-layer events, such as ranging.
messages	Displays MAC layer management messages.

Command History

Release	Modification
11.3 NA	This command was introduced.

Usage Guidelines

Of all the available **debug cable-modem** commands, the most useful is **debug cable-modem mac log**.

MAC log messages are written to a circular log file even when debugging is not turned on. These messages include time stamps, events, and information pertinent to these events. Enter the **debug cable-modem mac log** command to view MAC log messages. If you want to view this information without entering debug mode, enter the **show controllers cable-modem** *number* **mac log** command. The same information is displayed by both commands.

If the Cisco uBR900 series interface fails to come up or resets periodically, the MAC log will show what happened. For example, if an address is not obtained from the DHCP server, an error is logged, initialization starts over, and the Cisco uBR900 series cable access server router scans for a downstream frequency. The **debug cable-modem mac log** command displays the log from the oldest to the newest entry.

After initial ranging is successful (dhcp_state has been reached), further `RNG-REQ/RNG-RSP` messages and watchdog timer entries are suppressed from output unless the **verbose** keyword is used. Note that `CMAC_LOG_WATCHDOG_TIMER` entries while in the maintenance_state are normal when the **verbose** keyword is used.

Examples

The following example shows sample display output from the **debug cable-modem mac log** command. The fields of the output are the time since bootup, the log message, and in some cases a parameter that gives more detail about the log entry.

```
Router# debug cable-modem mac log

*Mar  7 01:42:59: 528302.040 CMAC_LOG_LINK_DOWN
*Mar  7 01:42:59: 528302.042 CMAC_LOG_RESET_FROM_DRIVER
*Mar  7 01:42:59: 528302.044 CMAC_LOG_STATE_CHANGE
wait_for_link_up_state
*Mar  7 01:42:59: 528302.046 CMAC_LOG_DRIVER_INIT_IDB_SHUTDOWN          0x08098D02
*Mar  7 01:42:59: 528302.048 CMAC_LOG_LINK_DOWN
*Mar  7 01:43:05: 528308.428 CMAC_LOG_DRIVER_INIT_IDB_RESET             0x08098E5E
*Mar  7 01:43:05: 528308.432 CMAC_LOG_LINK_DOWN
*Mar  7 01:43:05: 528308.434 CMAC_LOG_LINK_UP
```

```
*Mar  7 01:43:05: 528308.436 CMAC_LOG_STATE_CHANGE
ds_channel_scanning_state
*Mar  7 01:43:05: 528308.440 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
88/453000000/855000000/6000000
*Mar  7 01:43:05: 528308.444 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
89/93000000/105000000/6000000
*Mar  7 01:43:05: 528308.448 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
90/111250000/117250000/6000000
*Mar  7 01:43:05: 528308.452 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
91/231012500/327012500/6000000
*Mar  7 01:43:05: 528308.456 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
92/333015000/333015000/6000000
*Mar  7 01:43:05: 528308.460 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
93/339012500/399012500/6000000
*Mar  7 01:43:05: 528308.462 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
94/405000000/447000000/6000000
*Mar  7 01:43:05: 528308.466 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
95/123015000/129015000/6000000
*Mar  7 01:43:05: 528308.470 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
96/135012500/135012500/6000000
*Mar  7 01:43:05: 528308.474 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
97/141000000/171000000/6000000
*Mar  7 01:43:05: 528308.478 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
98/219000000/225000000/6000000
*Mar  7 01:43:05: 528308.482 CMAC_LOG_WILL_SEARCH_DS_FREQUENCY_BAND
99/177000000/213000000/6000000
*Mar  7 01:43:05: 528308.486 CMAC_LOG_WILL_SEARCH_SAVED_DS_FREQUENCY     663000000
*Mar  7 01:43:05: 528308.488 CMAC_LOG_WILL_SEARCH_USER_DS_FREQUENCY      663000000
*Mar  7 01:43:07: 528310.292 CMAC_LOG_DS_64QAM_LOCK_ACQUIRED             663000000
.
528383.992 CMAC_LOG_STATE_CHANGE                        registration_state
528384.044 CMAC_LOG_REG_REQ_MSG_QUEUED
528384.050 CMAC_LOG_REG_REQ_TRANSMITTED
528384.052 CMAC_LOG_REG_RSP_MSG_RCVD
528384.078 CMAC_LOG_COS_ASSIGNED_SID                    1/4
528384.102 CMAC_LOG_RNG_REQ_QUEUED                      4
528384.102 CMAC_LOG_REGISTRATION_OK
528384.102 CMAC_LOG_STATE_CHANGE                        establish_privacy_state
528384.102 CMAC_LOG_STATE_CHANGE                        maintenance_state
528388.444 CMAC_LOG_RNG_REQ_TRANSMITTED
528388.444 CMAC_LOG_RNG_RSP_MSG_RCVD
528398.514 CMAC_LOG_RNG_REQ_TRANSMITTED
528398.516 CMAC_LOG_RNG_RSP_MSG_RCVD
528408.584 CMAC_LOG_RNG_REQ_TRANSMITTED
528408.586 CMAC_LOG_RNG_RSP_MSG_RCVD
528414.102 CMAC_LOG_WATCHDOG_TIMER
528418.654 CMAC_LOG_RNG_REQ_TRANSMITTED
528418.656 CMAC_LOG_RNG_RSP_MSG_RCVD
528428.726 CMAC_LOG_RNG_REQ_TRANSMITTED
528428.728 CMAC_LOG_RNG_RSP_MSG_RCVD
528438.796 CMAC_LOG_RNG_REQ_TRANSMITTED
528438.798 CMAC_LOG_RNG_RSP_MSG_RCVD
528444.102 CMAC_LOG_WATCHDOG_TIMER
528444.492 CMAC_LOG_LINK_DOWN
528444.494 CMAC_LOG_RESET_FROM_DRIVER
528444.494 CMAC_LOG_STATE_CHANGE                        wait_for_link_up_state
528444.494 CMAC_LOG_DRIVER_INIT_IDB_SHUTDOWN            0x08098D02
528444.494 CMAC_LOG_LINK_DOWN
528474.494 CMAC_LOG_WATCHDOG_TIMER
528504.494 CMAC_LOG_WATCHDOG_TIMER
528534.494 CMAC_LOG_WATCHDOG_TIMER

0 events dropped due to lack of a chunk
```

The line "0 events dropped due to lack of a chunk" at the end of a display indicates that no log entries were discarded due to a temporary lack of memory, which means the log is accurate and reliable.

The following example compares the output of the **debug cable-modem mac log** command with the **debug cable-modem mac log verbose** command. The **verbose** keyword displays periodic events such as ranging.

```
Router# debug cable-modem mac log

Cable Modem mac log debugging is on
Router#
Router# debug cable-modem mac log verbose

Cable Modem mac log debugging is on (verbose)
Router#
574623.810 CMAC_LOG_RNG_REQ_TRANSMITTED
574623.812 CMAC_LOG_RNG_RSP_MSG_RCVD
574627.942 CMAC_LOG_WATCHDOG_TIMER
574633.880 CMAC_LOG_RNG_REQ_TRANSMITTED
574633.884 CMAC_LOG_RNG_RSP_MSG_RCVD
574643.950 CMAC_LOG_RNG_REQ_TRANSMITTED
574643.954 CMAC_LOG_RNG_RSP_MSG_RCVD
574654.022 CMAC_LOG_RNG_REQ_TRANSMITTED
574654.024 CMAC_LOG_RNG_RSP_MSG_RCVD
574657.978 CMAC_LOG_WATCHDOG_TIMER
574664.094 CMAC_LOG_RNG_REQ_TRANSMITTED
574664.096 CMAC_LOG_RNG_RSP_MSG_RCVD
574674.164 CMAC_LOG_RNG_REQ_TRANSMITTED
574674.166 CMAC_LOG_RNG_RSP_MSG_RCVD

Router# no debug cable-modem mac log verbose

Cable Modem mac log debugging is off
Router#
574684.234 CMAC_LOG_RNG_REQ_TRANSMITTED
574684.238 CMAC_LOG_RNG_RSP_MSG_RCVD
```

The following example shows display output for the **debug cable-modem mac messages** command. This command causes received cable MAC management messages to be displayed in a verbose format.

```
Router# debug cable-modem mac messages ?

  dynsrv   dynamic service mac messages
  map      map messages received
  reg-req  reg-req  messages transmitted
  reg-rsp  reg-rsp messages received
  rng-req  rng-req  messages transmitted
  rng-rsp  rng-rsp messages received
  sync     Sync messages received
  ucc-req  ucc-req messages received
  ucc-rsp  ucc-rsp messages transmitted
  ucd      UCD messages received
  <cr>
```

The **dynsrv** keyword displays Dynamic Service Add or Dynamic Service Delete messages during the off-hook/on-hook transitions of a phone connected to the Cisco uBR900 series cable access router.

In addition, sent REG-REQ messages are displayed in hexadecimal dump format. The output from this command is very verbose and is usually not needed for normal interface debugging. The command is most useful when attempting to attach a Cisco uBR900 series cable access router to a CMTS that is not DOCSIS-qualified.

For a description of the displayed fields of each message, refer to the DOCSIS Radio Frequency Interface Specification, v1.0 (SP-RFI-I04-980724).

```
Router# debug cable mac messages

*Mar  7 01:44:06:
*Mar  7 01:44:06: UCD MESSAGE
*Mar  7 01:44:06: -----------
*Mar  7 01:44:06:   FRAME HEADER
*Mar  7 01:44:06:     FC                        - 0xC2 == MAC Management
*Mar  7 01:44:06:     MAC_PARM                  - 0x00
*Mar  7 01:44:06:     LEN                       - 0xD3
*Mar  7 01:44:06:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:06:     DA                        - 01E0.2F00.0001
*Mar  7 01:44:06:     SA                        - 00E0.1EA5.BB60
*Mar  7 01:44:06:     msg LEN                   - C1
*Mar  7 01:44:06:     DSAP                      - 0
*Mar  7 01:44:06:     SSAP                      - 0
*Mar  7 01:44:06:     control                   - 03
*Mar  7 01:44:06:     version                   - 01
*Mar  7 01:44:06:     type                      - 02 == UCD
*Mar  7 01:44:06:     RSVD                      - 0
*Mar  7 01:44:06:   US Channel ID               - 1
*Mar  7 01:44:06:   Configuration Change Count  - 4
*Mar  7 01:44:06:   Mini-Slot Size              - 8
*Mar  7 01:44:06:   DS Channel ID               - 1
*Mar  7 01:44:06:   Symbol Rate                 - 8
*Mar  7 01:44:06:   Frequency                   - 20000000
*Mar  7 01:44:06:   Preamble Pattern            - CC CC CC CC CC CC CC CC CC CC CC CC CC
CC 0D 0D
*Mar  7 01:44:06:   Burst Descriptor 0
*Mar  7 01:44:06:     Interval Usage Code       - 1
*Mar  7 01:44:06:     Modulation Type           - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding     - 2 == OFF
*Mar  7 01:44:06:     Preamble Length           - 64
*Mar  7 01:44:06:     Preamble Value Offset     - 56
*Mar  7 01:44:06:     FEC Error Correction      - 0
*Mar  7 01:44:06:     FEC Codeword Info Bytes   - 16
*Mar  7 01:44:06:     Scrambler Seed            - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size        - 1
*Mar  7 01:44:06:     Guard Time Size           - 8
*Mar  7 01:44:06:     Last Codeword Length      - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off          - 1 == ON
*Mar  7 01:44:06:   Burst Descriptor 1
*Mar  7 01:44:06:     Interval Usage Code       - 3
*Mar  7 01:44:06:     Modulation Type           - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding     - 2 == OFF
*Mar  7 01:44:06:     Preamble Length           - 128
*Mar  7 01:44:06:     Preamble Value Offset     - 0
*Mar  7 01:44:06:     FEC Error Correction      - 5
*Mar  7 01:44:06:     FEC Codeword Info Bytes   - 34
*Mar  7 01:44:06:     Scrambler Seed            - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size        - 0
*Mar  7 01:44:06:     Guard Time Size           - 48
*Mar  7 01:44:06:     Last Codeword Length      - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off          - 1 == ON
*Mar  7 01:44:06:   Burst Descriptor 2
*Mar  7 01:44:06:     Interval Usage Code       - 4
*Mar  7 01:44:06:     Modulation Type           - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding     - 2 == OFF
*Mar  7 01:44:06:     Preamble Length           - 128
*Mar  7 01:44:06:     Preamble Value Offset     - 0
*Mar  7 01:44:06:     FEC Error Correction      - 5
*Mar  7 01:44:06:     FEC Codeword Info Bytes   - 34
```

```
*Mar  7 01:44:06:     Scrambler Seed            - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size        - 0
*Mar  7 01:44:06:     Guard Time Size           - 48
*Mar  7 01:44:06:     Last Codeword Length      - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off          - 1 == ON
*Mar  7 01:44:06:   Burst Descriptor 3
*Mar  7 01:44:06:     Interval Usage Code       - 5
*Mar  7 01:44:06:     Modulation Type           - 1 == QPSK
*Mar  7 01:44:06:     Differential Encoding     - 2 == OFF
*Mar  7 01:44:06:     Preamble Length           - 72
*Mar  7 01:44:06:     Preamble Value Offset     - 48
*Mar  7 01:44:06:     FEC Error Correction      - 5
*Mar  7 01:44:06:     FEC Codeword Info Bytes   - 75
*Mar  7 01:44:06:     Scrambler Seed            - 0x0152
*Mar  7 01:44:06:     Maximum Burst Size        - 0
*Mar  7 01:44:06:     Guard Time Size           - 8
*Mar  7 01:44:06:     Last Codeword Length      - 1 == FIXED
*Mar  7 01:44:06:     Scrambler on/off          - 1 == ON
*Mar  7 01:44:06:
*Mar  7 01:44:06:
*Mar  7 01:44:06: MAP MESSAGE
*Mar  7 01:44:06: -----------
*Mar  7 01:44:06:   FRAME HEADER
*Mar  7 01:44:06:     FC                        - 0xC3 == MAC Management with Extended
Header
*Mar  7 01:44:06:     MAC_PARM                  - 0x02
*Mar  7 01:44:06:     LEN                       - 0x42
*Mar  7 01:44:06:     EHDR                      - 0x00 0x00
*Mar  7 01:44:06:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:06:     DA                        - 01E0.2F00.0001
.
*Mar  7 01:44:17: RNG-RSP MESSAGE
*Mar  7 01:44:17: ---------------
*Mar  7 01:44:17:   FRAME HEADER
*Mar  7 01:44:17:     FC                        - 0xC2 == MAC Management
*Mar  7 01:44:17:     MAC_PARM                  - 0x00
*Mar  7 01:44:17:     LEN                       - 0x2B
*Mar  7 01:44:17:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:17:     DA                        - 00F0.1EB2.BB61
.
*Mar  7 01:44:20: REG-REQ MESSAGE
*Mar  7 01:44:20: ---------------
*Mar  7 01:44:20: C20000A5 000000E0  1EA5BB60 00F01EB2
*Mar  7 01:44:20: BB610093 00000301  06000004 03010104
*Mar  7 01:44:20: 1F010101 0204003D  09000304 001E8480
*Mar  7 01:44:20: 04010705 04000186  A0060200 0C070101
*Mar  7 01:44:20: 080300F0 1E112A01  04000000 0A020400
*Mar  7 01:44:20: 00000A03 04000002  58040400 00000105
*Mar  7 01:44:20: 04000000 01060400  00025807 04000000
*Mar  7 01:44:20: 3C2B0563 6973636F  06105E4F C908C655
*Mar  7 01:44:20: 61086FD5 5C9D756F  7B730710 434D5453
*Mar  7 01:44:20: 204D4943 202D2D2D  2D2D2D2D 0C040000
*Mar  7 01:44:20: 00000503 010100
*Mar  7 01:44:20:
*Mar  7 01:44:20:
*Mar  7 01:44:20: REG-RSP MESSAGE
*Mar  7 01:44:20: ---------------
*Mar  7 01:44:20:   FRAME HEADER
*Mar  7 01:44:20:     FC                        - 0xC2 == MAC Management
*Mar  7 01:44:20:     MAC_PARM                  - 0x00
*Mar  7 01:44:20:     LEN                       - 0x29
*Mar  7 01:44:20:   MAC MANAGEMENT MESSAGE HEADER
*Mar  7 01:44:20:     DA                        - 00F0.1EB2.BB61
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays baseline privacy information for a Cisco uBR900 series cable access router.
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for CiscouBR900 series cable access routers.
debug cable-modem map	Displays the timing from MAP messages to synchronize messages and the timing between MAP messages.

debug cable-modem map

To display the timing from MAP messages to synchronized messages and the timing between MAP messages on a Cisco uBR900 series cable access router, use the **debug cable-modem map** privileged EXEC command. To turn off the debugging messages, use the **no** form of this command.

debug cable-modem map

no debug cable-modem map

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3 NA	This command was introduced.

Examples

The following example shows display output for the **debug cable-modem map** command:

```
Router# debug cable-modem map

Cable Modem MAP debugging is on
Router#
*Mar  7 20:12:08: 595322.942: Min MAP to sync=72
*Mar  7 20:12:08: 595322.944: Max map to map time is 40
*Mar  7 20:12:08: 595322.982: Min MAP to sync=63
*Mar  7 20:12:08: 595323.110: Max map to map time is 41
*Mar  7 20:12:08: 595323.262: Min MAP to sync=59
*Mar  7 20:12:08: 595323.440: Max map to map time is 46
*Mar  7 20:12:09: 595323.872: Min MAP to sync=58
```

Related Commands

Command	Description
debug cable-modem bpkm	Displays baseline privacy information for a Cisco uBR900 series cable access router.
debug cable-modem bridge	Displays bridge filter processing information for a Cisco uBR900 series cable access router.
debug cable-modem error	Enables debugging messages for the cable interface driver on a Cisco uBR900 series.
debug cable-modem interrupts	Displays interrupts for CiscouBR900 series cable access routers.
debug cable-modem mac	Troubleshoots the Cisco uBR900 series MAC layer.

debug cable phy

To activate debugging of messages generated in the cable physical layer, use the **debug cable phy** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable phy

no debug cable phy

Syntax Description

This command has no arguments or keywords.

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 router 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.

Examples

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

To activate debugging of baseline privacy, use the **debug cable privacy** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable privacy

no debug cable privacy

Syntax Description

This command has no arguments or keywords.

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.

Examples

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

To activate quality of service (QoS) debugging, use the **debug cable qos** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable qos

no debug cable qos

Syntax Description

This command has no arguments or keywords.

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.

Examples

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

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

debug cable range

no debug cable range

Syntax Description

This command has no arguments or keywords.

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 keepalive monitoring.

Examples

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

```
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 radio frequency (RF) interface of the modem, not its Ethernet interface. The “Timing offset” is a measure of the distance between the modem and the Cisco uBR7246 universal broadband router expressed in 10.24-MHz clocks. This value is adjusted down to zero by the maintenance procedures. The first sixteen 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 universal broadband router 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

To display reset messages from cable interfaces, use the **debug cable reset** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable reset

no debug cable reset

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command activates display of reset messages from cable interfaces.

Examples

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

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

debug cable specmgmt

no debug cable specmgmt

Syntax Description

This command has no arguments or keywords.

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 lever changes controlled by spectrum groups.

Examples

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 the interface list of a frequency:

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

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

```
cmts_frequency_hop_decision(0x60B57FEC)
```

debug cable startalloc

To debug channel allocations on the HFC network, use the **debug cable startalloc** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable startalloc

no debug cable startalloc

Syntax Description

This command has no arguments or keywords.

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.

Examples

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 telco-return

To display debug messages for Telco return events, use the **debug cable telco-return** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cable telco-return

no debug cable telco-return

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for Telco return events is not enabled.

Command History

Release	Modification
12.0(4)XI	This command was introduced.

Examples

```
Router# debug cable telco-return

CMTS telco-return debugging is on
```

Related Commands

Command	Description
debug cable ucc	Displays debug messages for Telco return events.

debug cable ucc

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

debug cable ucc

no debug cable ucc

Syntax Description

This command has no arguments or keywords.

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.

Examples

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

To debug upstream channel descriptor (UCD) messages, use the **debug cable ucd** privileged EXEC command. The **no** form of this command disables debugging output.

debug cable ucd

no debug cable ucd

Syntax Description

This command has no arguments or keywords.

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.

Examples

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       t               - 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 0x0D
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 call fallback detail

To display details of the voice fallback, use the **debug call fallback detail** EXEC command. To disable debugging output, use the **no** form of this command.

debug call fallback detail

no debug call fallback detail

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled.

Command Modes

EXEC

Command History

Release	Modification
12.1(3)T	This command was introduced.

Examples

The following example depicts a call coming in to 1.1.1.4 with codec type g729r8. Because there is no cache entry for this destination, a probe is sent and values are inserted into the cache. A lookup is performed again, entry is found, and a fallback decision is made to admit the call.

```
Router# debug call fallback detail

When cache is empty:
debug call fallback detail:
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_check:no entry exists, enqueueing probe info... 1.1.1.4, codec:g729r8
2d19h:fb_main:Got FB_APP_INQ event
2d19h:fb_main:Dequeued prob info: 1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_cache_insert:insert:1.1.1.4, codec:g729r8
2d19h:fb_cache_insert:returning entry:1.1.1.4, codec:g729r8
2d19h:fb_initiate_probe:Creating probe... 1.1.1.4, codec:g729r8
2d19h:fb_initiate_probe:Created and started on probe #13, 1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
2d19h:fb_check:returned FB_CHECK_TRUE, 1.1.1.4, codec:g729r8
2d19h:fb_main:calling callback function with:TRUE
```

The following example depicts a call coming in to 1.1.1.4 with codec g729r8. A lookup is performed, entry is found, and a fallback decision is made to admit the call.

```
Router# debug call fallback detail

When cache is full:
2d19h:fb_lookup_cache:1.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
```

```
2d19h:fb_check:returned FB_CHECK_TRUE, 1.1.1.4, codec:g729r8
2d19h:fb_main:calling callback function with:TRUE
```

debug call fallback probes

To display details of the voice fallback probes, use the **debug call fallback probes** EXEC command. To disable debugging output, use the **no** form of this command.

debug call fallback probes

no debug call fallback probes

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging is not enabled.

Command Modes

EXEC

Command History

Release	Modification
12.1(3)T	This command was introduced.

Examples

The following example depicts a call coming in to 1.1.1.4 with codec type g729r8. Because there is no cache entry for this IP address, a g729r8 probe is initiated. The probe consists of 20 packet returns with an average delay of 43 milliseconds. The "jitter out" is jitter from source to destination router and "jitter in" is jitter from destination to source router. The delay, loss, and Calculated Planning Impairment Factor (ICPIF) values following g113_calc_icpif are the instantaneous values, whereas those values following "New smoothed values" are the values after applying the smoothing with weight 65.

```
Router# debug call fallback probes

2d19h:fb_initiate_probe:Probe payload is 32
2d19h:fb_main:NumOfRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0->
1.1.1.4, codec:g729r8
2d19h:g113_calc_icpif(delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0

2d19h:fb_main:Probe timer expired, 1.1.1.4, codec:g729r8
2d19h:fb_main:NumOfRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0->
1.1.1.4, codec:g729r8
2d19h:g113_calc_icpif(delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0
2d19h:fb_main:New smoothed values:inst_weight=65, ICPIF=0, Delay=43, Loss=0 -> 1.1.1.4,
codec:g729r8
```

debug call-mgmt

To display debugging information for call accounting, including modem and time slot usage, for active and recent calls, use the **debug call-mgmt** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug call-mgmt

no debug call-mgmt

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
12.1	This command was introduced.

Examples

The following is an example of the debug output that will be received after the **debug call-mgmt** command has been enabled:

```
Router# debug call-mgmt

Call Management debugging is on
Router#
Dec 26 13:57:27.710: msg_to_calls_mgmt: msg type CPM_NEW_CALL_CSM_CONNECT received
Dec 26 13:57:27.714: In actv_c_proc_message,
   access type CPM_INSERT_NEW_CALL,
   call type CPM_ISDN_ANALOG:
     CSM completed connecting a new modem call
.
.
.
Dec 26 13:57:45.906: msg_to_calls_mgmt: msg type CPM_NEW_CALL_ISDN_CONNECT received
Dec 26 13:57:45.906: In actv_c_proc_message,
   access type CPM_INSERT_NEW_CALL,
   call type CPM_ISDN_ANALOG:
     Added a new ISDN analog call to the active-calls list
     CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
     Mdm-Slot#1, Mdm-Port#3, TTY#219
.
.
.
```

```
Dec 26 13:58:25.682: Call mgmt per minute statistics:
     active list length: 1
     history list length: 3
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 1
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 2
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 3
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 4
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 5
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 6
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 7
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 8
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 9
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 10
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 11
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 12
Dec 26 13:58:25.682:      0 timeslots active at slot 7, ctrlr 13
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 14
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 15
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 16
Dec 26 13:58:25.686:      1 timeslots active at slot 7, ctrlr 17
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 18
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 19
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 20
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 21
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 22
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 23
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 24
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 25
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 26
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 27
Dec 26 13:58:25.686:      0 timeslots active at slot 7, ctrlr 28

Router# clear int as1/03

Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received
Dec 26 13:58:26.538: In actv_c_proc_message,
   access type CPM_REMOVE_DISC_CALL,
   call type CPM_ISDN_ANALOG:
     Removed a disconnected ISDN analog call
     CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
Dec 26 13:58:26.538:      Mdm-Slot#1, Mdm-Port#3, TTY#219
```

Table 26 describes the significant fields shown in the display.

Table 26 debug call-mgmt Command Field Descriptions

Field	Description
CPM_NEW_CALL_CSM_CONNECT	Indicates the arrival of a new call.
access type CPM_INSERT_NEW_CALL, call type CPM_ISDN_ANALOG:	Indicates that the new call is an analog ISDN B-channel call (either a voice call or a call over an analog modem), rather than a digital (V.110) call.
CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Mdm-Slot#1, Mdm-Port#3, TTY#219	Indicates that the call is connected via the B-channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).

Table 26 debug call-mgmt Command Field Descriptions (continued)

Field	Description
Dec 26 13:58:25.682: Call mgmt per minute statistics: active list length: 1 history list length: 3	Displays periodic statistics that give the allocation state of each DSX1 interface present in the system, as well as the number of current (active) and recent (history) calls.
Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_ AVAIL received	Indicates that the analog ISDN B-channel call has been disassociated from a modem.
access type CPM_REMOVE_DISC_CALL, call type CPM_ISDN_ANALOG: Removed a disconnected ISDN analog call	Indicates that the analog ISDN B-channel call has been disconnected.
CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Dec 26 13:58:26.538: Mdm-Slot#1, Mdm-Port#3, TTY#219	Indicates that the call has been disconnected via the B-channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).

debug call rsvp-sync events

To display events that occur during RSVP setup, use the **debug call rsvp-sync events** privileged EXEC command. To restore the default condition, use the **no** form of this command.

debug call rsvp-sync events

no debug call rsvp-sync events

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI1	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

It is highly recommended that you log the output from the **debug call rsvp-sync events** command to a buffer, rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows a portion of sample output for a call initiating RSVP when using the **debug call rsvp-sync events** command:

```
00:03:25:  Parameters: localip: 10.19.101.117 :localport: 16660

00:03:25:  Parameters: remoteip: 10.19.101.116 :remoteport: 17568

00:03:25:  QoS Primitive Event for Call id 0x1 : QoS Listen
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498

00:03:25: Hashed entry 0x1 in call table 0x61FC2498

00:03:25: Entry Not found

00:03:25:  Parameters: localip: 10.19.101.117

00:03:25:  remoteip: 10.19.101.116

00:03:25:  QoSpcb : 0x61FC34D8

00:03:25:  Response Status : 0
Starting timer for call with CallId 0x1 for 10000 secs

00:03:25:  Handling QoS Primitive QoS Listen

00:03:25:  Establishing RSVP RESV state : rsvp_request_reservation()

00:03:25:  For streams from 10.19.101.116:17568 to 10.19.101.117:16660
```

```
00:03:25:  RSVP Confirmation required

00:03:25:  QoS Primitive Event for Call id 0x1 : QoS Resv
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498

00:03:25: Hashed entry 0x1 in call table 0x61FC2498

00:03:25:  Initiating RVSP PATH messages to be Sent : reg_invoke_rsvp_advertise_sender()

00:03:25:  Advertizing for streams to 10.19.101.116:17568 from 10.19.101.117:16660

00:03:25:  RESV notification event received is : 2

00:03:25: Received RESVCONFIRM

00:03:25:  RESV CONFIRM message received from 10.19.101.116 for RESV setup from
10.19.101.117

00:03:25:  RESV event received is : 0

00:03:25:  RESV message received from 10.19.101.116:17568 for streams from
10.19.101.117:16660

00:03:25:  RESERVATIONS ESTABLISHED : CallId: 1Stop timer and notify Session Protocol of
Success (ie. if notification requested)

00:03:25:  Invoking spQoSresvCallback with Success
```

Related Commands

Command	Description
call rsvp-sync	Enables synchronization between RSVP and the H.323 voice signalling protocol.
call rsvp-sync resv-timer	Sets the timer for RSVP reservation setup.
debug call rsvp-sync func-trace	Displays messages about the software functions called by RSVP synchronization.
show call rsvp-sync conf	Displays the RSVP synchronization configuration.
show call rsvp-sync stats	Displays statistics for calls that attempted RSVP reservation.

debug call rsvp-sync func-trace

To display messages about software functions called by RSVP, use the **debug call rsvp-sync func-trace** privileged EXEC command. To restore the default condition, use the **no** form of this command.

debug call rsvp-sync func-trace

no debug call rsvp-sync func-trace

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command History

Release	Modification
12.1(3)XI1	This command was introduced.
12.1(5)T	This command was integrated into Cisco IOS Release 12.1(5)T.

Usage Guidelines

It is highly recommended that you log the output from the **debug call rsvp-sync func-trace** command to a buffer, rather than sending the output to the console; otherwise, the size of the output could severely impact the performance of the gateway.

Examples

The following example shows a portion of sample output for a call initiating RSVP when using the **debug call rsvp-sync func-trace** command in conjunction with the **debug call rsvp-sync events** command:

```
00:03:41: Entering Function QoS_Listen

00:03:41: Parameters:localip:10.10.101.116 :localport:17568

00:03:41:remoteip:10.10.101.117 :remoteport:0

00:03:41: Entering Function qos_dequeue_event

00:03:41: Entering Function process_queue_event

00:03:41: QoS Primitive Event for Call id 0x2 :QoS Listen
00:03:41: Entering Function get_pcb

00:03:41: Entering Function hash_tbl_lookup

00:03:41:Lookup to be done on hashkey 0x2 in hash table 0x61FAECD8

00:03:41: Entering Function hash_func

00:03:41:Hashed entry 0x2 in call table 0x61FAECD8

00:03:41:Entry Not found

00:03:41: Entering Function qos_dequeue_pcb
```

```
00:03:41: Entering Function qos_initialize_pcb

00:03:41: Parameters:localip:10.10.101.116

00:03:41:remoteip:10.10.101.117

00:03:41: QoSpcb :0x61FAFD18

00:03:41: Response Status :0

00:03:41: Entering Function hash_tbl_insert_entry

00:03:41: Entering Function hash_func

00:03:41: Handling QoS Primitive QoS Listen

00:03:41: Entering Function qos_dequeue_hash_port_entry

00:03:41: Entering Function qos_port_tbl_insert_entry

00:03:41: Entering Function hash_func

00:03:41: Doing RSVP Listen :rsvp_add_ip_listen_api()
```

Related Commands

Command	Description
call rsvp-sync	Enables synchronization between RSVP and the H.323 voice signaling protocol.
call rsvp-sync resv-timer	Sets the timer for RSVP reservation setup.
debug call rsvp-sync events	Displays the events that occur during RSVP synchronization.
show call rsvp-sync conf	Displays the RSVP synchronization configuration.
show call rsvp-sync stats	Displays statistics for calls that attempted RSVP reservation.

debug callback

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

debug callback

no debug callback

Syntax Description

This command has no arguments or keywords.

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.

Examples

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

Command	Description
debug arap	Displays ARAP events.
debug ppp	Displays information on traffic and exchanges in an internetwork implementing the PPP.

debug ccaal2 session

To display the ccaal2 function calls during call setup and teardown, use the **debug ccaal2 session** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug ccaal2 session

no debug ccaal2 session

Syntax Description

This command has no arguments or keywords.

Defaults

Debugging for AAL2 sessions is not enabled.

Command History

Release	Modification
12.1(1)XA	This command was introduced on the Cisco MC380 series.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command when troubleshooting an AAL2 trunk setup or teardown problem.

Examples

The following example shows sample output from the **debug ccaal2 session** command for a forced shutdown of a voice port:

```
Router# debug ccaal2 session

Router(config)# voice-port 1/1

Router(config-voiceport)# shutdown

Router(config-voiceport)#
3d21h:%Voice port in use.  Force shutdown.
3d21h:%Voice-port 1/1 is down.
3d21h:ccaal2_call_disconnect:peer tag 0
3d21h:ccaal2_evhandle_call_disconnect:Entered
3d21h:ccaal2_call_cleanup:freeccb 1, call_disconnected 1ccaal2_receive:xmitFunc is NULL
ccaal2_receive:xmitFunc is NULL

3d21h:starting incoming timer:Setting accept_incoming to FALSE and
3d21h:timer 2:(0x126AD48)starts - delay (70000)
3d21h:ccaal2_call_cleanup:Generating Call record
3d21h:cause=81 tcause=81    cause_text=unspecified
3d21h:ccaal2_call_cleanup:ccb 0x1506A84, vdbPtr 0x15ACFD0
        freeccb_flag=1, call_disconnected_flag=1
3d21h:%LINK-3-UPDOWN:Interface FXS 1/1, changed state to Administrative Shutdown
```

The following example shows sample output from the **debug ccaal2 session** command for a trunk setup on a voice port:

```
router(config-voiceport)# no shutdown
router(config-voiceport)#
3d21h:%Voice-port 1/1 is up.
3d21h:%LINK-3-UPDOWN:Interface FXS 1/1, changed state to up
```

```
3d21h:ccaal2_call_setup_request:Entered
3d21h:ccaal2_evhandle_call_setup_request:Entered
3d21h:ccaal2_initialize_ccb:preferred_codec set(-1)(0)
3d21h:ccaal2_evhandle_call_setup_request:preferred_codec set(5)(40). VAD is 0
3d21h:ccaal2_call_setup_trunk:subchannel linking successful

3d21h:ccaal2_caps_ind:PeerTag = 2007
3d21h:     codec(preferred) = 1, fax_rate = 2, vad = 1
3d21h:     cid = 25, config_bitmask = 0, codec_bytes = 40, signal_type=8
3d21h:encap VOAAL2
3d21h:%HTSP-5-UPDOWN:Trunk port(channel) [1/1] is up
```

Related Commands

Command	Description
show debug	Displays which debug commands are enabled.

debug ccfrf11 session

To display the ccfrf11 function calls during call setup and teardown, use the **debug ccfrf11 session** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccfrf11 session

no debug ccfrf11 session

Syntax Description

This command has no keywords or arguments.

Command History

Release	Modification
12.0(3)XG	This command was introduced on the Cisco 2600 and Cisco 3600 series routers.
12.0(4)T	This command was integrated into Cisco IOS Release 12.0(4)T.
12.0(7)XK	This command was first supported on the Cisco MC3810 series.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command to display debug information about the various FRF.11 VoFR service provider interface (SPI) functions. Note that this debug command does not display any information regarding the proprietary Cisco switched-VoFR SPI.

This debug is useful only when the session protocol is “frf11-trunk.”

Examples

The following example shows sample output from the **debug ccfr11 session** command:

```
Router# debug ccfrf11 session

INCOMING CALL SETUP (port setup for answer-mode):
*Mar  6 18:04:07.693:ccfrf11_process_timers:scb (0x60EB6040) timer (0x60EB6098) expired
*Mar  6 18:04:07.693:Setting accept_incoming to TRUE
*Mar  6 18:04:11.213:ccfrf11_incoming_request:peer tag 800:callingNumber=+2602100,
       calledNumber=+3622110
*Mar  6 18:04:11.213:ccfrf11_initialize_ccb:preffered_codec set(-1)(0)
*Mar  6 18:04:11.213:ccfrf11_evhandle_incoming_call_setup_request:calling +2602100,
       called +3622110 Incoming Tag 800
*Mar  6 18:04:11.217:ccfrf11_caps_ind:PeerTag = 800
*Mar  6 18:04:11.217:     codec(preferred) = 4, fax_rate = 2, vad = 2
*Mar  6 18:04:11.217:     cid = 30, config_bitmask = 0, codec_bytes = 20, signal_type=2
*Mar  6 18:04:11.217:     required_bandwidth 8192
*Mar  6 18:04:11.217:ccfrf11_caps_ind:Bandwidth reservation of 8192 bytes succeeded.
*Mar  6 18:04:11.221:ccfrf11_evhandle_call_connect:Entered

CALL SETUP (MASTER):
5d22h:ccfrf11_call_setup_request:Entered
5d22h:ccfrf11_evhandle_call_setup_request:Entered
5d22h:ccfrf11_initialize_ccb:preffered_codec set(-1)(0)
5d22h:ccfrf11_evhandle_call_setup_request:preffered_codec set(9)(24)
5d22h:ccfrf11_call_setup_trunk:subchannel linking successful
5d22h:ccfrf11_caps_ind:PeerTag = 810
5d22h:     codec(preferred) = 512, fax_rate = 2, vad = 2
```

```
5d22h:     cid = 30, config_bitmask = 1, codec_bytes = 24, signal_type=2
5d22h:     required_bandwidth 6500
5d22h:ccfrf11_caps_ind:Bandwidth reservation of 6500 bytes succeeded.

CALL TEARDOWN:
*Mar  6 18:09:14.805:ccfrf11_call_disconnect:peer tag 0
*Mar  6 18:09:14.805:ccfrf11_evhandle_call_disconnect:Entered
*Mar  6 18:09:14.805:ccfrf11_call_cleanup:freeccb 1, call_disconnected 1
*Mar  6 18:09:14.805:ccfrf11_call_cleanup:Setting accept_incoming to FALSE and starting
        incoming timer
*Mar  6 18:09:14.809:timer 2:(0x60EB6098)starts - delay (70000)
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:Alive timer stopped
*Mar  6 18:09:14.809:timer 1:(0x60F64104) stops
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:Generating Call record
*Mar  6 18:09:14.809:cause=10 tcause=10    cause_text="normal call clearing."
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:Releasing 8192 bytes of reserved bandwidth
*Mar  6 18:09:14.809:ccfrf11_call_cleanup:ccb 0x60F6404C, vdbPtr 0x610DB7A4
        freeccb_flag=1, call_disconnected_flag=1
```

Related Commands

Command	Description
debug call-mgmt	Displays the ccswvoice function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug vtsp session	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug cch323 h225

To trace the state transition of the H.225 state machine based on the processed event, use the **debug cch323 h225** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cch323 h225

no debug cch323 h225

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.

Usage Guidelines

State Descriptions

The state definitions of the different states of the H.225 state machine are as follows:

- H225_IDLE—This is the initial state of the H.225 state machine. The H.225 state machine is in this state before issuing a call setup request (for the outbound IP call case) or when ready to receive an incoming IP call.
- H225_SETUP—This is the call setup state. The state machine changes to this state after sending out a call setup request or after the reception of an incoming call indication.
- H225_ALERT—This is the call alerting state. The state machine changes to this state after sending the alerting message or after the reception of an alerting message from the peer.
- H225_CALLPROC—This is the call proceeding state.
- H225_ACTIVE—This is the call connected state. In this state, the call is active. The state machine changes to this state after sending the connect message to the peer or after the reception of the connect message from the peer.
- H225_WAIT_FOR_ARQ—This is the state where the H.225 state machine is waiting for the completion of the ARQ process from the RAS state machine.
- H225_WAIT_FOR_DRQ—This is the state where the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.
- H225_WAIT_FOR_H245—This is the state where the H.225 state machine is waiting for the success or failure from the H.245 state machine.

Events Description

The event definitions of the different events of the H.225 state machine are as follows:

- H225_EVENT_NONE— No event.
- H225_EVENT_ALERT—This event indicates to the H.225 state machine to send an alert message to the peer.
- H225_EVENT_ALERT_IND—This event indicates to the H.225 state machine that an alert message arrived from the peer.

- H225_EVENT_CALLPROC—This event indicates to the H.225 state machine to send a call proceeding message to the peer.
- H225_EVENT_CALLPROC_IND—This event indicates to the H.225 state machine that a call proceeding message is received from the peer.
- H225_EVENT_REJECT—This event indicates to the H.225 state machine to reject the call setup request from the peer.
- H225_EVENT_REJECT_IND—This event indicates to the H.225 state machine that a call setup request to the peer is rejected.
- H225_EVENT_RELEASE—This event indicates to the H.225 state machine to send a release complete message to the peer.
- H225_EVENT_RELEASE_IND—This event indicates to the H.225 state machine that a release complete message is received from the peer.
- H225_EVENT_SETUP—This event indicates to the H.225 state machine to send a setup message to the peer.
- H225_EVENT_SETUP_IND—This event indicates to the H.225 state machine that a setup message is received from the peer.
- H225_EVENT_SETUP_CFM—This event indicates to the H.225 state machine to send a connect message to the peer.
- H225_EVENT_SETUP_CFM_IND—This event indicates to the H.225 state machine that a connect message arrived from the peer.
- H225_EVENT_RAS_SUCCESS—This event indicates to the H.225 state machine that the pending RAS operation is successful.
- H225_EVENT_RAS_FAILED—This event indicates to the H.225 state machine that the pending RAS operation failed.
- H225_EVENT_H245_SUCCESS—This event indicates to the H.225 state machine that the pending H.245 operation is successful.
- H225_EVENT_H245_FAILED—This event indicates to the H.225 state machine that the pending H.245 operation failed.

Examples

The following is example output from the **debug cch323 h225** command.

```
Router# debug cch323 h225

20:59:17:Set new event H225_EVENT_SETUP
20:59:17:H225 FSM:received event H225_EVENT_SETUP while at state H225_IDLE
20:59:17:Changing from H225_IDLE state to H225_SETUP state
20:59:17:cch323_h225_receiver:received msg of type SETUPCFM_CHOSEN
20:59:17:H225 FSM:received event H225_EVENT_SETUP_CFM_IND while at state
H225_SETUP
20:59:17:Changing from H225_SETUP state to H225_ACTIVE state
20:59:17:Set new event H225_EVENT_H245_SUCCESS
20:59:17:H225 FSM:received event H225_EVENT_H245_SUCCESS while at state
H225_ACTIVE
20:59:20:Set new event H225_EVENT_RELEASE
20:59:20:H225 FSM:received event H225_EVENT_RELEASE while at state
H225_ACTIVE
20:59:20:Changing from H225_ACTIVE state to H225_WAIT_FOR_DRQ state
20:59:20:Set new event H225_EVENT_RAS_SUCCESS
20:59:20:H225 FSM:received event H225_EVENT_RAS_SUCCESS while at state
H225_WAIT_FOR_DRQ
```

```
20:59:20:Changing from H225_WAIT_FOR_DRQ state to H225_IDLE state
```

debug cch323 h245

To trace the state transition of the H.245 state machine based on the processed events, use the **debug cch323 h245** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cch323 h245

no debug cch323 h245

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.

Usage Guidelines

The H.245 state machines include the following three state machines:

- Master slave determination (MSD) state machine
- Capability exchange (CAP) state machine
- Open logical channel (OLC) state machine

State Definitions

The state definitions are as follows:

- H245_MS_NONE—This is the initial state of the master slave determination state machine.
- H245_MS_WAIT—In this state, a Master Slave Determination message is sent, and the device is waiting for the reply.
- H245_MS_DONE— The result is in.
- H245_CAP_NONE—This is the initial state of the capability exchange state machine.
- H245_CAP_WAIT—In this state, a capability exchange message is sent, and the device is waiting for reply.
- H245_CAP_DONE—The result is in.
- H245_OLC_NONE—This is the initial state of the open logical channel state machine.
- H245_OLC_WAIT: OLC message sent, and the device is waiting for reply.
- H245_OLC_DONE: OLC done.

Event Definitions

The event definitions are as follows:

- H245_EVENT_MSD—Send MSD message
- H245_EVENT_MS_CFM—Send MSD acknowledge message
- H245_EVENT_MS_REJ—Send MSD reject message
- H245_EVENT_MS_IND—Received MSD message
- H245_EVENT_CAP—Send CAP message

- H245_EVENT_CAP_CFM—Send CAP acknowledge message
- H245_EVENT_CAP_REJ—Send CAP reject message
- H245_EVENT_CAP_IND—Received CAP message
- H245_EVENT_OLC—Send OLC message
- H245_EVENT_OLC_CFM—Send OLC acknowledge message
- H245_EVENT_OLC_REJ—Send OLC reject message
- H245_EVENT_OLC_IND—Received OLC message

Examples

The following is sample output for the **debug cch323 h245** command.

```
Router# debug cch323 h245

20:58:23:Changing to new event H245_EVENT_MSD
20:58:23:H245 MS FSM:received event H245_EVENT_MSD while at state
H245_MS_NONE
20:58:23:changing from H245_MS_NONE state to H245_MS_WAIT state
20:58:23:Changing to new event H245_EVENT_CAP
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP while at state
H245_CAP_NONE
20:58:23:changing from H245_CAP_NONE state to H245_CAP_WAIT state
20:58:23:cch323_h245_receiver:received msg of type
M_H245_MS_DETERMINE_INDICATION
20:58:23:Changing to new event H245_EVENT_MS_IND
20:58:23:H245 MS FSM:received event H245_EVENT_MS_IND while at state
H245_MS_WAIT
20:58:23:cch323_h245_receiver:received msg of type
M_H245_CAP_TRANSFER_INDICATION
20:58:23:Changing to new event H245_EVENT_CAP_IND
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_IND while at state
H245_CAP_WAIT
20:58:23:cch323_h245_receiver:received msg of type
M_H245_MS_DETERMINE_CONFIRM
20:58:23:Changing to new event H245_EVENT_MS_CFM
20:58:23:H245 MS FSM:received event H245_EVENT_MS_CFM while at state
H245_MS_WAIT
20:58:23:changing from H245_MS_WAIT state to H245_MS_DONE state
0:58:23:cch323_h245_receiver:received msg of type M_H245_CAP_TRANSFER_CONFIRM
20:58:23:Changing to new event H245_EVENT_CAP_CFM
20:58:23:H245 CAP FSM:received event H245_EVENT_CAP_CFM while at state
H245_CAP_WAIT
20:58:23:changing from H245_CAP_WAIT state to H245_CAP_DONE state
20:58:23:Changing to new event H245_EVENT_OLC
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC while at state
H245_OLC_NONE
20:58:23:changing from H245_OLC_NONE state to H245_OLC_WAIT state
20:58:23:cch323_h245_receiver:received msg of type
M_H245_UCHAN_ESTABLISH_INDICATION
20:58:23:Changing to new event H245_EVENT_OLC_IND
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_IND while at state
H245_OLC_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_UCHAN_ESTAB_ACK
20:58:23:Changing to new event H245_EVENT_OLC_CFM
20:58:23:H245 OLC FSM:received event H245_EVENT_OLC_CFM while at state
H245_OLC_WAIT
20:58:23:changing from H245_OLC_WAIT state to H245_OLC_DONE state
```

debug cch323 ras

To trace the state transition of the RAS state machine based on the processed events, use the **debug cch323 ras** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug cch323 ras

no debug cch323 ras

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.3(6)NA2	This command was introduced.

Usage Guidelines

RAS operates in two state machines. One global state machine controls the overall RAS operation of the gateway. The other state machine is a per-call state machine that controls the active calls.

State Definitions

The state definitions of the different states of the RAS state machine are as follows:

- CCH323_RAS_STATE_NONE—This is the initial state of the RAS state machine.
- CCH323_RAS_STATE_GRQ—The state machine is in the GRQ state. In this state, the gateway is discovering a gatekeeper.
- CCH323_RAS_STATE_RRQ—The state machine is in the RRQ state. In this state, the gateway is registering with a gatekeeper.
- CCH323_RAS_STATE_IDLE—The global state machine is in the idle state.
- CCH323_RAS_STATE_URQ—The state machine is in the URQ state. In this state, the gateway is in the process of unregistering with a gatekeeper.
- CCH323_RAS_STATE_ARQ—The per-call state machine is in the process of admitting a new call.
- CCH323_RAS_STATE_ACTIVE—The per-call state machine is in the call active state.
- CCH323_RAS_STATE_DRQ—The per-call state machine is in the process of disengaging an active call.

Event Definitions

The event definitions of the different states of the RAS state machine are as follows:

- CCH323_RAS_EVENT_NONE—Nothing
- CCH323_RAS_EVENT_GWUP—Gateway is coming up
- CCH323_RAS_EVENT_GWDWN—Gateway is going down
- CCH323_RAS_EVENT_NEWCALL:—New call
- CCH323_RAS_EVENT_CALLDISC—Call disconnect
- CCH323_RAS_EVENT_GCF—Received GCF
- CCH323_RAS_EVENT_GRJ—Received GRJ

- CCH323_RAS_EVENT_ACF—Received ACF
- CCH323_RAS_EVENT_ARJ—Received ARJ
- CCH323_RAS_EVENT_SEND_RRQ—Send RRQ
- CCH323_RAS_EVENT_RCF—Received RCF
- CCH323_RAS_EVENT_RRJ—Received RRJ
- CCH323_RAS_EVENT_SEND_URQ—Send URQ
- CCH323_RAS_EVENT_URQ—Received URQ
- CCH323_RAS_EVENT_UCF—Received UCF
- CCH323_RAS_EVENT_SEND_UCF—Send UCF
- CCH323_RAS_EVENT_URJ—Received URJ
- CCH323_RAS_EVENT_BCF—Received BCF
- CCH323_RAS_EVENT_BRJ—Received BRJ
- CCH323_RAS_EVENT_DRQ—Received DRQ
- CCH323_RAS_EVENT_DCF—Received DCF
- CCH323_RAS_EVENT_SEND_DCF—Send DCF
- CCH323_RAS_EVENT_DRJ—Received DRJ
- CCH323_RAS_EVENT_IRQ—Received IRQ
- CCH323_RAS_EVENT_IRR—Send IRR
- CCH323_RAS_EVENT_TIMEOUT—Message timeout

Examples

The following is sample output from the **debug cch323 ras** command.

```
Router# debug cch323 ras

20:58:49:Changing to new event CCH323_RAS_EVENT_SEND_RRQ
cch323_run_ras_sm:received event CCH323_RAS_EVENT_SEND_RRQ while at CCH323_RAS_STATE_IDLE
state
cch323_run_ras_sm:changing to CCH323_RAS_STATE_RRQ state
cch323_ras_receiver:received msg of type RCF_CHOSEN
cch323_run_ras_sm:received event CCH323_RAS_EVENT_RCF while at CCH323_RAS_STATE_RRQ state
cch323_run_ras_sm:changing to CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_NEWCALL while at
CCH323_RAS_STATE_IDLE state
20:58:59:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_ARQ
cch323_ras_receiver:received msg of type ACF_CHOSEN
20:58:59:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_ACF while at
CCH323_RAS_STATE_ARQ state
20:58:59:cch323_percall_ras_sm:changing to new state
CCH323_RAS_STATE_ACTIVE
20:59:02:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_CALLDISC while
at CCH323_RAS_STATE_ACTIVE state
20:59:02:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_DRQ
cch323_ras_receiver:received msg of type DCF_CHOSEN
20:59:02:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_DCF while at
CCH323_RAS_STATE_DRQ state
20:59:02:cch323_percall_ras_sm:changing to new state CCH323_RAS_STATE_IDLE
20:59:04:cch323_percall_ras_sm:received event CCH323_RAS_EVENT_IRR while at
CCH323_RAS_STATE_ACTIVE state
20:59:04:cch323_percall_ras_sm:changing to new state
```

```
CCH323_RAS_STATE_ACTIVE
```

debug ccsip all

To enable all SIP-related debugging, use the **debug ccsip all** EXEC command. To disable all debugging output, use the **no** form of this command.

debug ccsip all

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.1.(3)T	The output of the command was changed.

Usage Guidelines

The **debug ccsip all** command enables the following debug SIP commands:

- **debug ccsip calls**
- **debug ccsip error**
- **debug ccsip events**
- **debug ccsip messages**
- **debug ccsip states**

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip all

All SIP call tracing enabled
Router#
*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_NONE, SUBSTATE_NONE)  to
(STATE_IDLE, SUBSTATE_NONE)
*Mar  6 14:10:42:  Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_idle_call_setup
*Mar  6 14:10:42:  act_idle_call_setup:Not using Voice Class Codec

*Mar  6 14:10:42: act_idle_call_setup: preferred_codec set[0] type :g711ulaw bytes: 160
*Mar  6 14:10:42:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_NONE)  to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Mar  6 14:10:42: REQUEST CONNECTION TO IP:166.34.245.231 PORT:5060

*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_CONNECTING)  to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_idle_connection_created
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_idle_connection_created: Connid(1) created to
166.34.245.231:5060, local_port 54113
```

```
*Mar  6 14:10:42: sipSPIAddLocalContact
*Mar  6 14:10:42:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_IDLE, SUBSTATE_CONNECTING)  to
(STATE_SENT_INVITE, SUBSTATE_NONE)
*Mar  6 14:10:42: Sent:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Cisco-Guid: 2881152943-2184249548-0-483039712
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427042
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  6 14:10:42: Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  6 14:10:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_sentinvite_new_message
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:42:  Roundtrip delay 4 milliseconds for method INVITE

*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_SENT_INVITE, SUBSTATE_NONE)  to
(STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
*Mar  6 14:10:42: Received:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137
```

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  6 14:10:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:10:42: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:42:  Roundtrip delay 8 milliseconds for method INVITE

*Mar  6 14:10:42: HandleSIP1xxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec      : g711ulaw , bytes :160
Inband Alerting       : 0

*Mar  6 14:10:42: 0x624CFEF8 : State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_PROCEEDING)  to (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
*Mar  6 14:10:46: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  6 14:10:46: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:46:  Roundtrip delay 3536 milliseconds for method INVITE

*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  act_recdproc_new_message: SDP MediaTypes negotiation
successful!
Negotiated Codec      : g711ulaw , bytes :160

*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sipSPIReconnectConnection
*Mar  6 14:10:46:  Queued event from SIP SPI : SIPSPI_EV_RECONNECT_CONNECTION
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  recv_200_OK_for_invite
*Mar  6 14:10:46:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:10:46: 0x624CFEF8 : State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_ALERTING)  to (STATE_ACTIVE, SUBSTATE_NONE)
*Mar  6 14:10:46: The Call Setup Information is :
```

```
        Call Control Block (CCB) : 0x624CFEF8
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20208
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20038
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:10:46: HandleUdpReconnection: Udp socket connected for fd: 1 with
166.34.245.231:5060
*Mar  6 14:10:46: Sent:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 137
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  6 14:10:46: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  6 14:10:46: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
*Mar  6 14:10:46: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  6 14:10:50: Received:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:36:44 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0



*Mar  6 14:10:50: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:54835
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  act_active_new_message
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sact_active_new_message_request
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:10:50:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sipSPIInitiateCallDisconnect : Initiate call
disconnect(16) for outgoing call
```

```
*Mar  6 14:10:50: 0x624CFEF8 : State change from (STATE_ACTIVE, SUBSTATE_NONE)  to
(STATE_DISCONNECTING, SUBSTATE_NONE)
*Mar  6 14:10:50: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612207
Content-Length: 0
CSeq: 101 BYE



*Mar  6 14:10:50:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  act_disconnecting_disconnect
*Mar  6 14:10:50: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  6 14:10:50:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
*Mar  6 14:10:50: CLOSE CONNECTION TO CONNID:1

*Mar  6 14:10:50: sipSPIIcpifUpdate :CallState: 4 Playout: 1755 DiscTime:48305031 ConnTime
48304651

*Mar  6 14:10:50: 0x624CFEF8 : State change from (STATE_DISCONNECTING, SUBSTATE_NONE)  to
(STATE_DEAD, SUBSTATE_NONE)
*Mar  6 14:10:50: The Call Setup Information is :

        Call Control Block (CCB) : 0x624CFEF8
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20208
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20038
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:10:50:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200

*Mar  6 14:10:50: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
Router#
```

From the other side of the call, the debug output is as follows:

```
3660-2#debug ccsip all
All SIP call tracing enabled
3660-2#
*Mar  8 17:36:40: Received:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:10:42 GMT
```

```
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Cisco-Guid: 2881152943-2184249548-0-483039712
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427042
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  8 17:36:40: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:54113
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sipSPISipIncomingCall
*Mar  8 17:36:40: 0x624D8CCC : State change from (STATE_NONE, SUBSTATE_NONE)  to
(STATE_IDLE, SUBSTATE_NONE)
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  act_idle_new_message
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sact_idle_new_message_invite
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:36:40:  sact_idle_new_message_invite:Not Using Voice Class Codec

*Mar  8 17:36:40: sact_idle_new_message_invite: Preferred codec[0] type: g711ulaw Bytes
:160
*Mar  8 17:36:40: sact_idle_new_message_invite: Media Negotiation successful for an
incoming call

*Mar  8 17:36:40: sact_idle_new_message_invite: Negotiated Codec      : g711ulaw, bytes
:160
Preferred Codec       : g711ulaw, bytes :160

*Mar  8 17:36:40:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:40: Num of Contact Locations 1 3660110 166.34.245.230 5060

*Mar  8 17:36:40: 0x624D8CCC : State change from (STATE_IDLE, SUBSTATE_NONE)  to
(STATE_RECD_INVITE, SUBSTATE_RECD_INVITE_CALL_SETUP)
*Mar  8 17:36:40: Sent:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  8 17:36:40:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_PROCEEDING
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  act_recdinvite_proceeding
*Mar  8 17:36:40:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_ALERTING
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  8 17:36:40: ccsip_caps_ind: codec(negotiated) = 5(Bytes 160)
*Mar  8 17:36:40: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  8 17:36:40: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
```

```
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  act_recdinvite_alerting
*Mar  8 17:36:40:  180 Ringing with SDP - not likely

*Mar  8 17:36:40:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:40: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:40: 0x624D8CCC : State change from (STATE_RECD_INVITE,
SUBSTATE_RECD_INVITE_CALL_SETUP)  to (STATE_SENT_ALERTING, SUBSTATE_NONE)
*Mar  8 17:36:40: Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  8 17:36:44:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_CONNECT
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  act_sentalert_connect
*Mar  8 17:36:44: sipSPIAddLocalContact
*Mar  8 17:36:44:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:44: 0x624D8CCC : State change from (STATE_SENT_ALERTING, SUBSTATE_NONE)  to
(STATE_SENT_SUCCESS, SUBSTATE_NONE)
*Mar  8 17:36:44: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137

v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20038 RTP/AVP 0

*Mar  8 17:36:44: Received:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
Date: Sat, 06 Mar 1993 19:10:42 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Max-Forwards: 6
```

```
Content-Type: application/sdp
Content-Length: 137
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 1212 283 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 RTP/AVP 0

*Mar  8 17:36:44: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:54113
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  act_sentsucc_new_message
*Mar  8 17:36:44: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:36:44: 0x624D8CCC : State change from (STATE_SENT_SUCCESS, SUBSTATE_NONE)  to
(STATE_ACTIVE, SUBSTATE_NONE)
*Mar  8 17:36:44: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D8CCC
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20038
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20208
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:36:47:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  act_active_disconnect
*Mar  8 17:36:47:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_NONE)  to
(STATE_ACTIVE, SUBSTATE_CONNECTING)
*Mar  8 17:36:47: REQUEST CONNECTION TO IP:166.34.245.230 PORT:5060

*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_CONNECTING)  to
(STATE_ACTIVE, SUBSTATE_CONNECTING)
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  act_active_connection_created
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection: Connid(1) created to
166.34.245.230:5060, local_port 54835
*Mar  8 17:36:47:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_ACTIVE, SUBSTATE_CONNECTING)  to
(STATE_DISCONNECTING, SUBSTATE_NONE)
*Mar  8 17:36:47: Sent:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:36:44 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0
```

```
*Mar  8 17:36:47: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:54835
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27D3FCA8-C7F
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:10:50 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612207
Content-Length: 0
CSeq: 101 BYE



*Mar  8 17:36:47: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:54113
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  act_disconnecting_new_message
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sact_disconnecting_new_message_response
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:36:47:  Roundtrip delay 4 milliseconds for method BYE

*Mar  8 17:36:47: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  8 17:36:47:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
*Mar  8 17:36:47: CLOSE CONNECTION TO CONNID:1

*Mar  8 17:36:47: sipSPIIcpifUpdate :CallState: 4 Playout: 1265 DiscTime:66820800 ConnTime
66820420

*Mar  8 17:36:47: 0x624D8CCC : State change from (STATE_DISCONNECTING, SUBSTATE_NONE)  to
(STATE_DEAD, SUBSTATE_NONE)
*Mar  8 17:36:47: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D8CCC
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20038
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20208
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:36:47:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200

*Mar  8 17:36:47: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
```

Related Commands

Command	Description
debug ccsip calls	Displays all SIP SPI call tracing and traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Displays SIP SPI errors and traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UAC and the access server.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip calls

To show all SIP Service Provider Interface (SPI) call tracing, use the **debug ccsip calls** command.

debug ccsip calls

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.1.(3)T	The output of the command was changed.

Usage Guidelines

This command traces the SIP call details as they are updated in the SIP call control block.

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip calls

SIP Call statistics tracing is enabled
Router#
*Mar  6 14:12:33: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D078C
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20644
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20500
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:12:40: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D078C
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.230
         Source IP Port    (Media): 20644
```

```
         Destn  IP Address (Media): 166.34.245.231
         Destn  IP Port    (Media): 20500
         Destn SIP Addr (Control) : 166.34.245.231
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.231

*Mar  6 14:12:40:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200

Router#
```

From the other side of the call, the debug output is as follows:

```
Router#debug ccsip calls
SIP Call statistics tracing is enabled
Router#
*Mar  8 17:38:31: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D9560
         State of The Call        : STATE_ACTIVE
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20500
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20644
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:38:38: The Call Setup Information is :

        Call Control Block (CCB) : 0x624D9560
         State of The Call        : STATE_DEAD
         TCP Sockets Used         : NO
         Calling Number           : 3660110
         Called Number            : 3660210
         Negotiated Codec         : g711ulaw
         Source IP Address (Media): 166.34.245.231
         Source IP Port    (Media): 20500
         Destn  IP Address (Media): 166.34.245.230
         Destn  IP Port    (Media): 20644
         Destn SIP Addr (Control) : 166.34.245.230
         Destn SIP Port (Control) : 5060
         Destination Name         : 166.34.245.230

*Mar  8 17:38:38:

        Disconnect Cause (CC)    : 16
        Disconnect Cause (SIP)   : 200
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip error	Displays SIP SPI errors. This command traces all error messages generated from errors encountered by the SIP subsystem.

Command	Description
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip error

To show SIP SPI errors, use the **debug ccsip error** EXEC command.

debug ccsip error

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.1.(3)T	The output of the command was changed.

Usage Guidelines

This command traces all error messages generated from errors encountered by the SIP subsystem.

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip error

SIP Call error tracing is enabled
Router#

*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_idle_call_setup
*Mar  6 14:16:41:  act_idle_call_setup:Not using Voice Class Codec

*Mar  6 14:16:41: act_idle_call_setup: preferred_codec set[0] type :g711ulaw bytes: 160
*Mar  6 14:16:41: REQUEST CONNECTION TO IP:166.34.245.231 PORT:5060

*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_idle_connection_created
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_idle_connection_created: Connid(1) created to
166.34.245.231:5060, local_port 55674
*Mar  6 14:16:41: sipSPIAddLocalContact
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:16:41: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_sentinvite_new_message
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:41:  Roundtrip delay 4 milliseconds for method INVITE

*Mar  6 14:16:41: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:16:41: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:41:  Roundtrip delay 8 milliseconds for method INVITE
```

```
*Mar  6 14:16:41: HandleSIP1xxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec      : g711ulaw , bytes :160
Inband Alerting       : 0

*Mar  6 14:16:45: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:5060
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  act_recdproc_new_message
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sipSPICheckResponse : Updating session description
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:45:  Roundtrip delay 3844 milliseconds for method INVITE

*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  act_recdproc_new_message: SDP MediaTypes negotiation
successful!
Negotiated Codec      : g711ulaw , bytes :160

*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sipSPIReconnectConnection
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  recv_200_OK_for_invite
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:16:45: HandleUdpReconnection: Udp socket connected for fd: 1 with
166.34.245.231:5060
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  6 14:16:45: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  6 14:16:45: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
*Mar  6 14:16:45: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  6 14:16:49: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.231:56101
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  act_active_new_message
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sact_active_new_message_request
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sipSPIInitiateCallDisconnect : Initiate call
disconnect(16) for outgoing call
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  act_disconnecting_disconnect
*Mar  6 14:16:49: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  6 14:16:49: CLOSE CONNECTION TO CONNID:1

*Mar  6 14:16:49: sipSPIIcpifUpdate :CallState: 4 Playout: 2945 DiscTime:48340988 ConnTime
48340525

*Mar  6 14:16:49: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
Router#
```

From the other side of the call, the debug output is as follows:

```
Router#debug ccsip error
SIP Call error tracing is enabled
Router#
*Mar  8 17:42:39: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:55674
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sipSPISipIncomingCall
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  act_idle_new_message
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sact_idle_new_message_invite
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:42:39:  sact_idle_new_message_invite:Not Using Voice Class Codec

*Mar  8 17:42:39: sact_idle_new_message_invite: Preferred codec[0] type: g711ulaw Bytes
:160
*Mar  8 17:42:39: sact_idle_new_message_invite: Media Negotiation successful for an
incoming call
```

```
*Mar  8 17:42:39: sact_idle_new_message_invite: Negotiated Codec      : g711ulaw, bytes
:160
Preferred Codec       : g711ulaw, bytes :160

*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:39: Num of Contact Locations 1 3660110 166.34.245.230 5060

*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  act_recdinvite_proceeding
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  ccsip_caps_ind
*Mar  8 17:42:39: ccsip_caps_ind: codec(negotiated) = 5(Bytes 160)
*Mar  8 17:42:39: ccsip_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*Mar  8 17:42:39: ccsip_caps_ind: set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  ccsip_caps_ack
*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  act_recdinvite_alerting
*Mar  8 17:42:39:  180 Ringing with SDP - not likely

*Mar  8 17:42:39: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  act_sentalert_connect
*Mar  8 17:42:42: sipSPIAddLocalContact
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:42: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:55674
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  act_sentsucc_new_message
*Mar  8 17:42:42: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  act_active_disconnect
*Mar  8 17:42:47: REQUEST CONNECTION TO IP:166.34.245.230 PORT:5060

*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  act_active_connection_created
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICheckSocketConnection: Connid(1) created to
166.34.245.230:5060, local_port 56101
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sip_stats_method
*Mar  8 17:42:47: HandleUdpSocketReads :Msg enqueued for SPI with IPaddr:
166.34.245.230:55674
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  act_disconnecting_new_message
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sact_disconnecting_new_message_response
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICheckResponse
*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sip_stats_status_code
*Mar  8 17:42:47:  Roundtrip delay 0 milliseconds for method BYE

*Mar  8 17:42:47: CCSIP-SPI-CONTROL:  sipSPICallCleanup
*Mar  8 17:42:47: CLOSE CONNECTION TO CONNID:1

*Mar  8 17:42:47: sipSPIIcpifUpdate :CallState: 4 Playout: 1255 DiscTime:66856757 ConnTime
66856294

*Mar  8 17:42:47: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote
port: 5060
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Displays all SIP Service Provider Interface (SPI) call tracing and traces the SIP call details as they are updated in the SIP call control block.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.

Command	Description
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip events

To show all SIP SPI events tracing, use the **debug ccsip events** command.

debug ccsip events

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Usage Guidelines

This command traces the events posted to SIP SPI from all interfaces.

Command History

Release	Modification
12.1(1)T	This command was introduced.

Examples

From one side of the call, the debug output is as follows:

```
Router# debug ccsip events

SIP Call events tracing is enabled
Router#

*Mar  6 14:17:57:  Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
*Mar  6 14:17:57:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  6 14:17:57:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:18:00:  Queued event from SIP SPI : SIPSPI_EV_RECONNECT_CONNECTION
*Mar  6 14:18:00:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:18:04:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  6 14:18:04:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  6 14:18:04:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
Router#
```

From the other side of the call, the debug output is as follows:

```
Router# deb ccsip events

SIP Call events tracing is enabled
Router#

*Mar  8 17:43:55:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:43:55:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_PROCEEDING
*Mar  8 17:43:55:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_ALERTING
*Mar  8 17:43:55:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:43:58:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_CONNECT
*Mar  8 17:43:58:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:44:01:  Queued event From SIP SPI to CCAPI/DNS : SIPSPI_EV_CC_CALL_DISCONNECT
*Mar  8 17:44:01:  Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
*Mar  8 17:44:01:  Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar  8 17:44:01:  Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Shows all SIP Service Provider Interface (SPI) call tracing. This command traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Shows SIP SPI errors. This command traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip messages	Shows all SIP SPI message tracing. This command traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.
debug ccsip states	Shows all SIP SPI state tracing. This command traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip messages

To show all SIP SPI message tracing, use the **debug ccsip message**s command.

debug ccsip messages

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

This command traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.

Examples

From one side of the call, the debug output is as follows:

```
Router#debug ccsip message

SIP Call messages tracing is enabled
Router#

*Mar  6 14:19:14: Sent:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Cisco-Guid: 2881152943-2184249568-0-483551624
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427554
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  6 14:19:14: Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:55820
```

```
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  6 14:19:14: Received:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  6 14:19:16: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  6 14:19:16: Sent:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 138
CSeq: 101 ACK
```

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  6 14:19:19: Received:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:45:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612717
CSeq: 101 BYE
Content-Length: 0



*Mar  6 14:19:19: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:19:19 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612717
Content-Length: 0
CSeq: 101 BYE

Router#
```

From the other side of the call, the debug output is as follows:

```
Router#debug ccsip message

SIP Call messages tracing is enabled
Router#

*Mar  8 17:45:12: Received:
INVITE sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Cisco-Guid: 2881152943-2184249568-0-483551624
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427554
Contact: <sip:3660110@166.34.245.230:5060;user=phone>
Expires: 180
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
```

```
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  8 17:45:12: Sent:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0



*Mar  8 17:45:12: Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  8 17:45:14: Sent:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Contact: <sip:3660210@166.34.245.231:5060;user=phone>
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 138

v=0
o=CiscoSystemsSIP-GW-UserAgent 1193 7927 IN IP4 166.34.245.231
s=SIP Call
t=0 0
c=IN IP4 166.34.245.231
m=audio 20224 RTP/AVP 0

*Mar  8 17:45:14: Received:
ACK sip:3660210@166.34.245.231:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.230:55820
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
```

```
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Max-Forwards: 6
Content-Type: application/sdp
Content-Length: 138
CSeq: 101 ACK

v=0
o=CiscoSystemsSIP-GW-UserAgent 5596 7982 IN IP4 166.34.245.230
s=SIP Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0

*Mar  8 17:45:17: Sent:
BYE sip:3660110@166.34.245.230:5060;user=phone SIP/2.0
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Mon, 08 Mar 1993 22:45:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612717
CSeq: 101 BYE
Content-Length: 0



*Mar  8 17:45:17: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  166.34.245.231:53600
From: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>;tag=27DBC6D8-1357
To: "3660110" <sip:3660110@166.34.245.230>
Date: Sat, 06 Mar 1993 19:19:19 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC@172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731612717
Content-Length: 0
CSeq: 101 BYE
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Displays all SIP Service Provider Interface (SPI) call tracingand traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Displays SIP SPI errors and traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip states	Displays all SIP SPI state tracing and traces the state machine changes of SIP SPI and displays the state transitions.

debug ccsip states

To show all SIP SPI state tracing, use the **debug ccsip states** EXEC command.

debug ccsip states

Syntax Description

This command has no arguments or keywords.

Command Modes

EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.

Usage Guidelines

This command traces the state machine changes of SIP SPI and displays the state transitions.

Examples

The following is sample output for the **debug ccsip states** command.

```
Router# debug ccsip states

SIP Call states tracing is enabled
Router#

*Jan 2 18:34:37.793:0x6220C634 :State change from (STATE_NONE, SUBSTATE_NONE) to
(STATE_IDLE, SUBSTATE_NONE)
*Jan 2 18:34:37.797:0x6220C634 :State change from (STATE_IDLE, SUBSTATE_NONE) to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Jan 2 18:34:37.797:0x6220C634 :State change from (STATE_IDLE, SUBSTATE_CONNECTING) to
(STATE_IDLE, SUBSTATE_CONNECTING)
*Jan 2 18:34:37.801:0x6220C634 :State change from (STATE_IDLE, SUBSTATE_CONNECTING) to
(STATE_SENT_INVITE, SUBSTATE_NONE)
*Jan 2 18:34:37.809:0x6220C634 :State change from (STATE_SENT_INVITE, SUBSTATE_NONE) to
(STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
*Jan 2 18:34:37.853:0x6220C634 :State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_PROCEEDING) to (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
*Jan 2 18:34:38.261:0x6220C634 :State change from (STATE_RECD_PROCEEDING,
SUBSTATE_PROCEEDING_ALERTING) to (STATE_ACTIVE, SUBSTATE_NONE)
*Jan 2 18:35:09.860:0x6220C634 :State change from (STATE_ACTIVE, SUBSTATE_NONE) to
(STATE_DISCONNECTING, SUBSTATE_NONE)
*Jan 2 18:35:09.868:0x6220C634 :State change from (STATE_DISCONNECTING, SUBSTATE_NONE) to
(STATE_DEAD, SUBSTATE_NONE)
*Jan 2 18:28:38.404: Queued event from SIP SPI :SIPSPI_EV_CLOSE_CONNECTION
```

PSTN Cause Code and SIP Event Mappings

Table 27 lists the PSTN cause codes that can be sent as an ISDN cause information element (IE) and the corresponding SIP event for each.

Table 27 PSTN Cause Code to SIP Event Mappings

PSTN Cause Code	Description	SIP Event
1	Unallocated number	410 Gone
3	No route to destination	404 Not found
16	Normal call clearing	BYE
17	User busy	486 Busy here
18	No user responding	480 Temporarily unavailable
19	No answer from the user	
21	Call rejected	603 Decline
22	Number changed	301Moved temporarily
27	Destination out of order	404 Not found
28	Address incomplete	484 Address incomplete
29	Facility rejected	501 Not implemented
31	Normal unspecified	404 Not found
34	No circuit available	503 Service unavailable
38	Network out of order	
41	Temporary failure	
42	Switching equipment congestion	
44	Requested channel not available	
47	Resource unavailable	
55	Incoming class barred within CUG	603 Decline
57	Bearer capability not authorized	501 Not implemented
58	Bearer capability not available	
63	Service or option unavailable	503 Service unavailable
65	Bearer cap not implemented	501 Not implemented
79	Service or option not implemented	
87	User not a member of CUG	603 Decline
88	Incompatible destination	400 Bad request
95	Invalid message	
102	Recover on timer expiry	408 Request timeout
111	Protocol error	400 Bad request
127	Interworking unspecified	500 Internal server error
Any code other than those listed		500 Internal server error

Table 28 lists the SIP events and the corresponding PSTN cause codes for each.

Table 28 SIP Event to PSTN Cause Code Mapping

SIP Event	PSTN Cause Code	Description
400 Bad request	127	Interworking
401 Unauthorized	57	Bearer cap not authorized
402 Payment required	21	Call rejected
403 Forbidden	57	Bearer cap not authorized
404 Not found	1	Unallocated number
405 Method not allowed	127	Interworking
406 Not acceptable		
407 Proxy authentication required	21	Call rejected
408 Request timeout	102	Recover on timer expiry
409 Conflict	41	Temporary failure
410 Gone	1	Unallocated number
411 Length required	127	Interworking
413 Request entity too long		
414 Request URI too long		
415 Unsupported media type	79	Service or option not available
420 Bad extension	127	Interworking
480 Temporarily unavailable	18	No user response
481 Call leg does not exist	127	Interworking
482 Loop detected		
483 Too many hops		
484 Address incomplete	28	Address incomplete
485 Address ambiguous	1	Unallocated number
486 Busy here	17	User busy
500 Internal server error	41	Temporary failure
501 Not implemented	79	Service or option not implemented
502 Bad gateway	38	Network out of order
503 Service unavailable	63	Service or option not available
504 Gateway timeout	102	Recover on timer expiry
505 Version not implemented	127	Interworking
600 Busy everywhere	17	User busy
603 Decline	21	Call rejected
604 Does not exist anywhere	1	Unallocated number
606 Not acceptable	58	Bearer cap not available

debug ccswvoice vofr-debug

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-debug** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-debug

no debug ccswvoice vofr-debug

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced.

Usage Guidelines

This command does not apply to the Cisco MC3810 networking device.

This command should be used when attempting to troubleshoot a Voice over Frame Relay (VoFR) call that uses the "cisco-switched" session protocol. It provides the same information as the **debug ccswvoice vofr-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswvoice vofr-debug** command:

```
Router# debug ccswvoice vofr-debug

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 03:02:08.719:ccswvofr_bridge_drop:dropping bridge calls src 17 dst 16 dlci 100
      cid 9 state ACTIVE
*Mar  1 03:02:08.727:ccswvofr:callID 17 dlci 100 cid 9 state ACTIVE event O/G REL
*Mar  1 03:02:08.735:ccswvofr:callID 17 dlci 100 cid 9 state RELEASE event I/C RELCOMP
*Mar  1 03:02:08.735:ccswvofr_store_call_history_entry:cause=22 tcause=22
      cause_text=no circuit.
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 03:03:22.651:ccswvofr:callID 23 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 03:03:22.651:ccswvofr_out_callinit_setup:callID 23 using dlci 100 cid 10
*Mar  1 03:03:22.659:ccswvofr:callID 23 dlci 100 cid 10 state O/G INIT event I/C PROC
*Mar  1 03:03:22.667:ccswvofr:callID 23 dlci 100 cid 10 state O/G PROC event I/C CONN
ccfrf11_caps_ind:codec(preferred) = 0
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug frame-relay fragment	Displays information related to Frame Relay fragmentation on a PVC.

Command	Description
debug voice vofr	Displays Cisco trunk and FRF.11 trunk call setup attempts and displays which dial peer is used in the call setup.
debug vpm error	Displays the behavior of the Holst state machine.
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Displays all SIP Service Provider Interface (SPI) call tracing and traces the SIP call details as they are updated in the SIP call control block.
debug ccsip error	Displays SIP SPI errors and traces all error messages generated from errors encountered by the SIP subsystem.
debug ccsip events	Displays all SIP SPI events tracing and traces the events posted to SIP SPI from all interfaces.
debug ccsip messages	Displays all SIP SPI message tracing and traces the SIP messages exchanged between the SIP UA client (UAC) and the access server.

debug ccswvoice vofr-session

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-session** privileged EXEC command. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-session

no debug ccswvoice vofr-session

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced.

Usage Guidelines

This command does not apply to the Cisco MC3810 networking device.

This command can be used to show the state transitions of the cisco-switched-vofr state machine as a call is processed. It should be used when attempting to troubleshoot a Voice over Frame Relay (VoFR) call that uses the “cisco-switched” session protocol.

Examples

The following example shows sample output from the **debug ccswvoice vofr-session** command:

```
Router# debug ccswvoice vofr-session

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 02:58:13.203:ccswvofr:callID 14 dlci 100 cid 8 state ACTIVE event O/G REL
*Mar  1 02:58:13.215:ccswvofr:callID 14 dlci 100 cid 8 state RELEASE event I/C RELCOMP
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 02:59:46.551:ccswvofr:callID 17 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 02:59:46.559:ccswvofr:callID 17 dlci 100 cid 9 state O/G INIT event I/C PROC
*Mar  1 02:59:46.567:ccswvofr:callID 17 dlci 100 cid 9 state O/G PROC event I/C CONN
3640_vofr(config-voiceport)#
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccsip all	Displays the ccswvoice function calls during call setup and teardown.
debug voice vofr	Displays Cisco trunk and FRF.11 trunk call setup attempts and displays which dial peer is used in the call setup.
debug vpm error	Displays the behavior of the Holst state machine.
debug vtsp port	Displays the behavior of the VTSP state machine.

debug ccswvoice vo-debug

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice voo-debug** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice voatm-debug

no debug ccswvoice voatm-debug

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(1)MA	This command was introduced on the Cisco MC3810 networking device.
12.0(7)XK	This command was first supported on the Cisco 3600 series router.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command when attempting to troubleshoot a Vo call that uses the "cisco-switched" session protocol. This command provides the same information as the **debug ccswvoice vo-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswvoice vo-debug** command:

```
Router# debug ccswvoice vo-debug

2w2d: ccswvoice: callID 529927 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccswvoice_out_callinit_setup: callID 529927 using pvcid 1 cid 15
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G INIT event I/C PROC
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G PROC event I/C
ALERTccfrf11_caps_ind: codec(preferred) = 1

2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state O/G ALERT event I/C CONN
2w2d: ccswvoice_bridge_drop: dropping bridge calls src 529927 dst 529926 pvcid 1 cid 15
state ACTIVE
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state ACTIVE event O/G REL
2w2d: ccswvoice: callID 529927 pvcid 1 cid 15 state RELEASE event I/C RELCOMP
2w2d: ccswvo_store_call_history_entry: cause=10 tcause=10    cause_text=normal call
clearing.
```

Related Commands

Command	Description
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.

debug ccswvoice vo-session

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vo-session** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vo-session

no debug ccswvoice vo-session

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(1)MA	This command was introduced on the Cisco MC3810 networking device.
12.0(7)XK	This command was first supported on the Cisco 3600 series router.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command to show the state transitions of the cisco-switched-vo state machine as a call is processed. This command should be used when attempting to troubleshoot a Vo call that uses the "cisco-switched" session protocol.

Examples

The following example shows sample output from the **debug ccswvoice vo-session** command:

```
Router# debug ccswvoice vo-session

2w2d: ccswvoice: callID 529919 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G INIT event I/C PROC
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G PROC event I/C ALERT
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state O/G ALERT event I/C CONN
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state ACTIVE event O/G REL
2w2d: ccswvoice: callID 529919 pvcid 1 cid 11 state RELEASE event I/C RELCOMP
```

Related Commands

Command	Description
debug call-mgmt	Displays the ccswvoice function calls during call setup and teardown.

debug ccswvoice vofr-debug

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-debug** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-debug

no debug ccswvoice vofr-debug

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced on the Cisco 2600 and Cisco 3600 series routers.
12.0(4)T	This command was integrated into Cisco IOS Release 12.0(4)T.
12.0(7)XK	This command was first supported on the Cisco MC3810 networking device.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command when troubleshooting a VoFR call that uses the "cisco-switched" session protocol. This command provides the same information as the **debug ccswvoice vofr-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswvoice vofr-debug** command:

```
Router# debug ccswvoice vofr-debug

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 03:02:08.719:ccswvofr_bridge_drop:dropping bridge calls src 17 dst 16 dlci 100
      cid 9 state ACTIVE
*Mar  1 03:02:08.727:ccswvofr:callID 17 dlci 100 cid 9 state ACTIVE event O/G REL
*Mar  1 03:02:08.735:ccswvofr:callID 17 dlci 100 cid 9 state RELEASE event I/C RELCOMP
*Mar  1 03:02:08.735:ccswvofr_store_call_history_entry:cause=22 tcause=22
      cause_text=no circuit.
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 03:03:22.651:ccswvofr:callID 23 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 03:03:22.651:ccswvofr_out_callinit_setup:callID 23 using dlci 100 cid 10
*Mar  1 03:03:22.659:ccswvofr:callID 23 dlci 100 cid 10 state O/G INIT event I/C PROC
*Mar  1 03:03:22.667:ccswvofr:callID 23 dlci 100 cid 10 state O/G PROC event I/C CONN
ccfrf11_caps_ind:codec(preferred) = 0
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug ccswvoice vofr-session	Displays the ccswvoice function calls during call setup and teardown.
debug vtsp session	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug ccswvoice vofr-session

To display the ccswvoice function calls during call setup and teardown, use the **debug ccswvoice vofr-session** command in privileged EXEC mode. Use the **no** form of this command to turn off the debug function.

debug ccswvoice vofr-session

no debug ccswvoice vofr-session

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)XG	This command was introduced on the Cisco 2600 and Cisco 3600 series routers.
12.0(4)T	This command was integrated into Cisco IOS Release 12.0(4)T.
12.0(7)XK	This command was first supported on the Cisco MC3810 networking device.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use this command to show the state transitions of the cisco-switched-vofr state machine as a call is processed, and when attempting to troubleshoot a VoFR call that uses the “cisco-switched” session protocol.

Examples

The following example shows sample output from the **debug ccswvoice vofr-session** command:

```
Router# debug ccswvoice vofr-session

CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar  1 02:58:13.203:ccswvofr:callID 14 dlci 100 cid 8 state ACTIVE event O/G REL
*Mar  1 02:58:13.215:ccswvofr:callID 14 dlci 100 cid 8 state RELEASE event I/C RELCOMP
3640_vofr(config-voiceport)#

CALL SETUP (outgoing):
*Mar  1 02:59:46.551:ccswvofr:callID 17 dlci -1 cid -1 state NULL event O/G SETUP
*Mar  1 02:59:46.559:ccswvofr:callID 17 dlci 100 cid 9 state O/G INIT event I/C PROC
*Mar  1 02:59:46.567:ccswvofr:callID 17 dlci 100 cid 9 state O/G PROC event I/C CONN
3640_vofr(config-voiceport)#
```

Related Commands

Command	Description
debug ccfrf11 session	Displays the ccfrf11 function calls during call setup and teardown.
debug call-mgmt	Displays the ccswvoice function calls during call setup and teardown.
debug vtsp session	Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.

debug cdapi

To display information about the call distributor application programming interface (CDAPI), use the **debug cdapi** privileged EXEC command.

debug cdapi {*detail* | *events*}

Syntax Description

detail	Displays when applications register or unregister with CDAPI, when calls are added or deleted from the CDAPI routing table, and when CDAPI messages are created and freed. It is useful for determining if messages are being lost (or not freed) and the size of the raw messages passed between CDAPI and applications so that you can check that the correct number of bytes is being passed.
events	Displays the events passing between CDAPI and an application or signalling stack. This debug is useful for determining if certain ISDN messages are not being received by an application and if calls are not being directed to an application.

Defaults

Disabled

Command History

Release	Modification
12.0(6)T	This command was introduced.

Examples

The following example shows output for the **debug cdapi** command:

```
003909 ISDN Se123 RX <-  SETUP pd = 8  callref = 0x06BB
003909         Bearer Capability i = 0x9090A2
003909         Channel ID i = 0xA18381
003909         Facility i =
0x9FAA068001008201008B0100A1180202274C020100800F534341524C415454492D3530303733
003909         Progress Ind i = 0x8183 - Origination address is non-ISDN
003909         Calling Party Number i = 0xA1, '50073'
003909         Called Party Number i = 0xC1, '3450070'
003909 CDAPI Se123 TX -> CDAPI_MSG_CONNECT_IND to TSP CDAPI Application call = 0x24
003909       From Appl/Stack = ISDN
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
003909       Calling Party Number = 50073
003909       Called Party Number = 3450070
003909 CDAPI Se123 TX -> CDAPI_MSG_CONNECT_RESP to ISDN call = 0x24
003909       From Appl/Stack = TSP CDAPI Application
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
003909 CDAPI-ISDN Se123 RX <- CDAPI_MSG_CONNECT_RESP from TSP CDAPI Application call =
0x24
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
```

```
003909 CDAPI Se123 TX -> CDAPI_MSG_SUBTYPE_CALL_PROC_REQ to ISDN call = 0x24
003909       From Appl/Stack = TSP CDAPI Application
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
003909 CDAPI-ISDN Se123 RX <- CDAPI_MSG_SUBTYPE_CALL_PROC_REQ from TSP CDAPI Application
call = 0x24
003909       Call Type = VOICE
003909       B Channel = 0
003909       Cause     = 0
003909 ISDN Se123 TX ->  CALL_PROC pd = 8  callref = 0x86BB
003909         Channel ID i = 0xA98381
```

Related Commands

Command	Description
debug cdapi	Displays information about the CDAPI.
debug voip rawmsg	Displays the raw message owner, length, and pointer.

debug cdp

To enable debugging of the Cisco Discovery Protocol (CDP), use the **debug cdp** privileged EXEC command. The **no** form of this command disables debugging output.

debug cdp {**packets** | **adjacency** | **events**}

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.

Examples

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

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

debug cdp ip

no debug cdp ip

Syntax Description

This command has no arguments or keywords.

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.

Examples

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 sent, 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 172.16.1.0/24 into the IP routing table:

  ```
  CDP-IP: Updating prefix 172.16.1.0/24 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 172.16.1.0/24 from a CDP neighbor connected via Ethernet interface 0/0. The neighbor IP address is 10.0.01.

  ```
  CDP-IP: Reading prefix 172.16.1.0/24 source 10.0.0.1 via Ethernet0/0
  ```

Related Commands

Command	Description
debug ip routing	Displays information on RIP routing table updates and route cache updates.

debug channel events

To display processing events on Cisco 7000 series routers that occur on the channel adapter interfaces of all installed adapters, use the **debug channel events** privileged EXEC command. Use the **no** form of this command to disable debugging output.

debug channel events

no debug channel events

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
12.0(3)T	This command was introduced.

Usage Guidelines

This command displays CMCC adapter events that occur on the CIP or CPA 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, the messages can indicate the possible source of the problems. To observe the statistic message (cip_love_letter) sent every 10 seconds, use the **debug channel love** command.

When configuring or making changes to a router or interface that supports IBM channel attach, enable the **debug channel events** command. 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.

Examples

The following sample output is 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 s**how extended channel** *slot*/*port* **subchannels** command. The CIP needs to send this information 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 also shows up when you use the **debug channel packets** command.

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

The following is a sample output that is generated by the **debug channel events** command when a CMPC+ IP TG connection is activated with the host:

```
1d05h:Channel4/2:Received route UP  for tg (768)
1d05h:Adding STATIC ROUTE for vc:768
```

The following is a sample output from the **debug channel events** command when a CMPC+ IP TG connection is deactivated:

```
1d05h:Channel4/2:Received route DOWN  for tg (768)
1d05h:Deleting STATIC ROUTE for vc:768
```

Related Commands

Command	Description
debug channel ilan	Displays CIP love letter events.
debug channel packets	Displays per-packet debugging output.

debug channel ilan

To display messages relating to configuration and bridging using CMCC internal LANs and to help debug source-route bridging (SRB) problems related to CMCC internal LANs, use the **debug channel ilan** privileged EXEC command. The **no** form of this command disables debugging output.

debug channel ilan

no debug channel ilan

Syntax Description

This command has no arguments or keywords.

Command History

Release	Modification
11.0(3)	This command was introduced.

Usage Guidelines

The **debug channel ilan** command displays events related to CMCC internal LANs. This command is useful for debugging problems associated with CMCC internal LAN configuration. It is also useful for debugging problems related to SRB packet flows through internal LANs.

Examples

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

```
Router# debug channel ilan

Channel internal LANs debugging is on
```

The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the LLC end station in Cisco IOS software did not exist:

```
CIP ILAN(Channel3/2-Token): Packet dropped - NULL LLC
```

The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the CMCC had not yet acknowledged the internal MAC adapter configuration command:

```
Channel3/2: ILAN Token-Ring 3 - CIP internal MAC adapter not acknowledged
DMAC(4000.7000.0001) SMAC(0c00.8123.0023)
```

Related Commands

Command	Description
debug source bridge	Displays information about packets and frames transferred across a source-route bridge.
debug channel events	Displays processing that occurs on the channel adapter interfaces of all installed adapters.

debug channel love

To display Channel Interface Processor (CIP) love letter events, use the **debug channel love** privileged EXEC command. The **no** form of this command disables debugging output.

debug channel love

no debug channel love

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

This command displays 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 sent every 10 seconds. This command is valid for the Cisco 7000 series routers only.

Examples

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 ll, 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 ll, but no cip_info
```

Related Commands

Command	Description
debug channel events	Displays processing that occur on the channel adapter interfaces of all installed adapters.
debug channel packets	Displays per-packet debugging output.

debug channel packets

To display per-packet debugging output, use the **debug channel packets** privileged EXEC command. The output reports information when a packet is received or a transmission is attempted. The **no** form of this command disables debugging output.

debug channel packets

no debug channel packets

Syntax Description

This command has no arguments or keywords.

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 sent correctly.

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

Examples

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 29 describes the significant fields in the display.

Table 29 debug channel packets Field Descriptions

Field	Description
(Channel3/0)	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 =	Number of bytes in the packet, including internal overhead.
vc =	Value from 0 to 511 that maps to the **claw** interface configuration command. This information is from the MAC layer.
type =	Encapsulation type in the MAC layer. The value 0800 indicates an IP datagram.
src	Origin, or source, of the packet, as opposed to the previous hop address.
dst	Destination of the packet, as opposed to the next hop address.

debug clns esis events

To display uncommon End System-to-Intermediate System (ES-IS) events, including previously unknown neighbors, neighbors that have aged out, and neighbors that have changed roles (ES-IS, for example), use the **debug clns esis events** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns esis events

no debug clns esis events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns esis events

ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```

The following line indicates that the router received a hello packet (ISH) from the IS at MAC address aa00.0400.2c05 on Ethernet interface 1. The hold time (or number of seconds to consider this packet valid before deleting it) for this packet is 30 seconds.

```
ES-IS: ISH from aa00.0400.2c05 (Ethernet1), HT 30
```

The following line indicates that the router received a hello packet (ESH) from the ES at MAC address aa00.0400.9105 on the Ethernet interface 1. The hold time is 150 seconds.

```
ES-IS: ESH from aa00.0400.9105 (Ethernet1), HT 150
```

The following line indicates that the router sent an IS hello packet on the Ethernet interface 0 to all ESs on the network. The network entity title (NET) address of the router is 49.0001.0400.AA00.6904.00; the hold time for this packet is 299 seconds; and the header length of this packet is 20 bytes.

```
ES-IS: ISH sent to All ESs (Ethernet1): NET 49.0001.AA00.0400.6904.00, HT 299, HLEN 20
```

debug clns esis packets

To enable display information on End System-to-Intermediate System (ES-IS) packets that the router has received and sent, use the **debug clns esis packets** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns esis packets

no debug clns esis packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns esis packets

ES-IS: ISH sent to All ESs (Ethernet0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
ES-IS: ISH sent to All ESs (Ethernet1): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
ES-IS: ISH from aa00.0400.6408 (Ethernet0), HT 299
ES-IS: ISH sent to All ESs (Tunnel0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.O906.4023.00, HT 299, HLEN 34
IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
```

The following line indicates that the router has sent an IS hello packet on Ethernet interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Ethernet0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 33
```

The following line indicates that the router has sent an IS hello packet on Ethernet interface 1 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Ethernet1): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```

The following line indicates that the router received a hello packet on Ethernet interface 0 from an intermediate system, aa00.0400.6408. The hold time for this packet is 299 seconds.

```
ES-IS: ISH from aa00.0400.6408 (Ethernet0), HT 299
```

The following line indicates that the router has sent an IS hello packet on Tunnel interface 0 to all ESs on the network. This hello packet indicates that the NET of the router is 47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00. The hold time for this packet is 299 seconds. The packet header is 33 bytes in length.

```
ES-IS: ISH sent to All ESs (Tunnel0): NET
47.0005.80ff.ef00.0000.0001.5940.1600.8906.4023.00, HT 299, HLEN 34
```

The following line indicates that on Ethernet interface 0, the router received a hello packet from an end system with an SNPA of 0000.0c00.bda8. The hold time for this packet is 300 seconds.

```
IS-IS: ESH from 0000.0c00.bda8 (Ethernet0), HT 300
```

debug clns events

To display CLNS events that are occurring at the router, use the **debug clns events** privileged EXEC command. The **no** form of this command disables debugging output.

debug clns events

no debug clns events

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns events

CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00
         via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
CLNS: Forwarding packet size 117
      from 39.0001.2222.2222.2222.00
      to 49.0002.0001.AAAA.AAAA.AAAA.00
      via 49.0002 (Ethernet3 0000.0c00.b5a3)
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18,
      redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```

The following line indicates that the router received an echo PDU on Ethernet interface 3 from source network service access point (NSAP) 39.0001.2222.2222.2222.00. The exclamation point at the end of the line has no significance.

```
CLNS: Echo PDU received on Ethernet3 from 39.0001.2222.2222.2222.00!
```

The following lines indicate that the router at source NSAP 39.0001.3333.3333.3333.00 is sending a CLNS echo packet to destination NSAP 39.0001.2222.2222.2222.00 via an IS with system ID 2222.2222.2222. The packet is being sent on Ethernet interface 3, with a MAC address of 0000.0c00.3a18.

```
CLNS: Sending from 39.0001.3333.3333.3333.00 to 39.0001.2222.2222.2222.00
         via 2222.2222.2222 (Ethernet3 0000.0c00.3a18)
```

The following lines indicate that a CLNS echo packet 117 bytes in size is being sent from source NSAP 39.0001.2222.2222.2222.00 to destination NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 via the router at NSAP 49.0002. The packet is being forwarded on the Ethernet interface 3, with a MAC address of 0000.0c00.b5a3.

```
CLNS: Forwarding packet size 117
      from 39.0001.2222.2222.2222.00
      to 49.0002.0001.AAAA.AAAA.AAAA.00
      via 49.0002 (Ethernet3 0000.0c00.b5a3)
```

The following lines indicate that the router sent a redirect packet on the Ethernet interface 3 to the NSAP 39.0001.2222.2222.2222.00 at MAC address 0000.0c00.3a18 to indicate that NSAP 49.0002.0001.AAAA.AAAA.AAAA.00 can be reached at MAC address 0000.0c00.b5a3.

```
CLNS: RD Sent on Ethernet3 to 39.0001.2222.2222.2222.00 @ 0000.0c00.3a18,
      redirecting 49.0002.0001.AAAA.AAAA.AAAA.00 to 0000.0c00.b5a3
```