

debug cch323

To provide debug output for various components within the H.323 subsystem, use the **debug cch323** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug cch323 {all | error | h225 | h245 | nxe | ras | rawmsg | session}
```

```
no debug cch323
```

Syntax Description

all	Enables all debug cch323 commands.
error	Traces errors encountered in the H.323 subsystem and can be used to help troubleshoot problems with H.323 calls.
h225	Traces the state transition of the H.225 state machine on the basis of the processed event.
h245	Traces the state transition of the H.245 state machine on the basis of the processed events.
nxe	Displays Annex E events that have been transmitted and received.
ras	Traces the state transition of the Registration, Admission, and Status (RAS) state machine on the basis of the processed events.
rawmsg	Troubleshoots raw message buffer problems.
session	Traces general H.323 events and can be used to troubleshoot H.323 problems.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(6)NA2	The debug cch323 command and the following keywords were introduced: h225 , h245 , and ras .
12.2(2)XA	The nxe keyword was added.
12.2(4)T	The following keywords were introduced: all , error , rawmsg , and session . The nxe keyword was integrated into Cisco IOS Release 12.2(4)T on all Cisco H.323 platforms. This command does not support the Cisco access server platforms in this release.
12.2(2)XB1	This command was implemented on the Cisco AS5850.
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

Usage Guidelines**The debug cch323 Command with the all Keyword**

When used with the **debug cch323** command, the **all** keyword provides debug output for various components within the H.323 subsystem.

The **debug cch323** command used with the **all** keyword enables the following **debug cch323** commands:

error	Enables a CCH323 Service Provider Interface (SPI) trace.
h225	Enables an H225 state machine debugging trace.
h245	Enables an H245 state machine debugging trace.
nxe	Enables an Annex E debugging trace.
ras	Enables a RAS state machine debugging trace.
rawmsg	Enables a CCH323 RAWMSG debugging trace.
session	Enables a Session debugging trace.

**Caution**

Using the **debug cch323 all** command could slow your system and flood the TTY if there is significant call traffic.

The debug cch323 Command with the error Keyword

When used with the **debug cch323** command, the **error** keyword allows you to trace errors encountered in the H.323 subsystem.

**Note**

There is little or no output from this command when there is a stable H.323 network.

The debug cch323 Command with the h225 Keyword

When used with the **debug cch323** command, the **h225** keyword allows you to trace the state transition of the H.225 state machine on the basis of the processed event.

The definitions of the different states of the H.225 state machine follow:

- **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 receiving 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 receiving 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 receiving the connect message from the peer.
- **H225_WAIT_FOR_ARQ**—This is the state in which the H.225 state machine is waiting for the completion of the Admission Request (ARQ) process from the RAS state machine.

- H225_WAIT_FOR_DRQ—This is the state in which the H.225 state machine is waiting for the completion of the Disengage Request (DRQ) process from the RAS state machine.
- H225_WAIT_FOR_H245—This is the state in which the H.225 state machine is waiting for the success or failure from the H.245 state machine.

The definitions of the different events of the H.225 state machine follow:

- H225_EVENT_NONE—There is no event.
- H225_EVENT_ALERT—This event instructs 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 instructs 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 has been received from the peer.
- H225_EVENT_REJECT—This event instructs 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 has been rejected.
- H225_EVENT_RELEASE—This event instructs 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 has been received from the peer.
- H225_EVENT_SETUP—This event instructs 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 has been received from the peer.
- H225_EVENT_SETUP_CFM—This event instructs 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 succeeded.
- 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 succeeded.
- H225_EVENT_H245_FAILED—This event indicates to the H.225 state machine that the pending H.245 operation failed.

The debug cch323 Command with the h245 Keyword

When used with the **debug cch323** command, the **h245** keyword allows you to trace the state transition of the H.245 state machine on the basis of the processed event.

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

The state definitions follow:

- H245_MS_NONE—This is the initial state of the MSD state machine.
- H245_MS_WAIT—In this state, an MSD 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 CAP state machine.
- H245_CAP_WAIT—In this state, a CAP message is sent, and the device is waiting for the reply.
- H245_CAP_DONE—The result is in.
- H245_OLC_NONE—This is the initial state of the OLC state machine.
- H245_OLC_WAIT—In this state, an OLC message is sent, and the device is waiting for the reply.
- H245_OLC_DONE—The result is in.

The event definitions follow:

- 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.

The debug cch323 Command with the nxe Keyword

When used with the **debug cch323** command, the **nxe** keyword allows you to display the Annex E events that have been transmitted and received.

The debug cch323 Command with the ras Keyword

When used with the **debug cch323** command, the **ras** keyword allows you to trace the state transition of the RAS state machine based on the processed events.

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.

The definitions of the different states of the RAS state machine follow:

- CCH323_RAS_STATE_NONE—This is the initial state of the RAS state machine.
- CCH323_RAS_STATE_GRQ—The state machine is in the Gatekeeper Request (GRQ) state. In this state, the gateway is discovering a gatekeeper.
- CCH323_RAS_STATE_RRQ—The state machine is in the Registration Request (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 Unregistration Request (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.

The definitions of the different events of the RAS state machine follow:

- 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 Gatekeeper Confirmation (GCF).
- CCH323_RAS_EVENT_GRJ—Received Gatekeeper Rejection (GRJ).
- CCH323_RAS_EVENT_ACF—Received Admission Confirmation (ACF).
- CCH323_RAS_EVENT_ARJ—Received Admission Reject (ARJ).
- CCH323_RAS_EVENT_SEND_RRQ—Send Registration Request (RRQ).
- CCH323_RAS_EVENT_RCF—Received Registration Confirmation (RCF).
- CCH323_RAS_EVENT_RRJ—Received Registration Rejection (RRJ).
- CCH323_RAS_EVENT_SEND_URQ—Send Unregistration Request (URQ).
- CCH323_RAS_EVENT_URQ—Received URQ.
- CCH323_RAS_EVENT_UCF—Received Unregister Confirmation (UCF).
- CCH323_RAS_EVENT_SEND_UCF—Send UCF.
- CCH323_RAS_EVENT_URJ—Received Unregister Reject (URJ).
- CCH323_RAS_EVENT_BCF—Received Bandwidth Confirm (BCF).
- CCH323_RAS_EVENT_BRJ—Received Bandwidth Rejection (BRJ).
- CCH323_RAS_EVENT_DRQ—Received Disengage Request (DRQ).

- CCH323_RAS_EVENT_DCF—Received Disengage Confirm (DCF).
- CCH323_RAS_EVENT_SEND_DCF—Send DCF.
- CCH323_RAS_EVENT_DRJ—Received Disengage Reject (DRJ).
- CCH323_RAS_EVENT_IRQ—Received Interrupt Request (IRQ).
- CCH323_RAS_EVENT_IRR—Send Information Request (IRR).
- CCH323_RAS_EVENT_TIMEOUT—Message timeout.

The debug cch323 Command with the rawmsg Keyword

When used with the **debug cch323** command, the **rawmsg** keyword allows you to troubleshoot raw message buffer problems.



Caution

Using the **debug cch323** command with the **rawmsg** keyword could slow your system and flood the TTY if there is significant call traffic.

The debug cch323 Command with the session Keyword

Used with the **debug cch323** command, the **session** keyword allows you to trace general H.323 events.



Caution

Using the **debug cch323 session** command could slow your system and flood the TTY if there is significant call traffic.

Examples

The debug cch323 Command with the all Keyword Example

The **debug cch323 all** command and keyword combination provides output for the following keywords: **error**, **h225**, **h245**, **nxe**, **ras**, **rawmsg**, and **session**. Examples of output for each keyword follow.

The debug cch323 Command with the error Keyword Example

The following is sample output from a typical **debug cch323 error** request on a Cisco 3640 router:

```
Router# debug cch323 error
cch323_h225_receiver:received msg of unknown type 5
```

The debug cch323 Command with the h225 Keyword Example

The following is sample output from a typical **debug cch323 h225** request on a Cisco 3640 router:

```
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
```

Table 31 describes the significant fields shown in the display.

Table 31 *debug cch323 h225 Command Field Descriptions*

Field	Description
H225_EVENT_SETUP	This event instructs the H.225 state machine to send a setup message to the peer.
H225_IDLE	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	The call setup state. The state machine changes to this state after sending out a call setup request or after receiving an incoming call indication.
SETUPCFM_CHOSEN	The H225 connect message that has been received from a remote H323 endpoint.
H225_EVENT_SETUP_CFM_IND	This event indicates to the H.225 state machine that a connect message arrived from the peer.
H225_ACTIVE	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 receiving the connect message from the peer.
H225_EVENT_H425_SUCCESS	This event indicates to the H.225 state machine that the pending H.245 operation succeeded.
H225_EVENT_RELEASE	This event instructs the H.225 state machine to send a release complete message to the peer.
H225_WAIT_FOR_DRQ	The state in which the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.
H225_EVENT_RAS_SUCCESS	This event indicates to the H.225 state machine that the pending RAS operation succeeded.
H225 FSM	The finite state machine.

The debug cch323 Command with the h245 Keyword Example

The following is sample output from a typical **debug cch323 h245** request on a Cisco 3640 router:

```
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

```

Table 32 describes the significant fields shown in the display.

Table 32 *debug cch323 h245 Command Field Descriptions*

Field	Description
H245_EVENT_MSD	Send MSD event message to the state machine.
H245 MS FSM	An H225 master slave determination finite state machine.
H245_MS_NONE	The initial state of the MSD state machine.
H245_MS_WAIT	In this state, a MSD message is sent, and the device is waiting for the reply.
H245_EVENT_CAP	Send CAP event message.
H245 CAP FSM	This is the H245 terminal CAP finite state machine.
H245_CAP_NONE	The initial state of the CAP state machine.
H245_CAP_WAIT	In this state, a CAP message is sent, and the device is waiting for the reply.
M_H245_MS_DETERMINE_INDICATION	The MSD message that has been received by an H245 terminal from a remote H323 endpoint.
H245_EVENT_MS_IND	Received MSD event message.
M_H245_CAP_TRANSFER_INDICATION	A CAP message that has been received by the H245 terminal from an H323 remote endpoint.

Table 32 *debug cch323 h245 Command Field Descriptions (continued)*

Field	Description
H245_EVENT_CAP_IND	Received CAP event message.
M_H245_MS_DETERMINE_CONFIRM	A confirmation message that the H245 master slave termination message was sent.
H245_EVENT_MS_CFM	Send MSD acknowledge event message.
H245_MS_DONE	The result is in.
M_H245_CAP_TRANSFER_CONFIRM	An indication to the H245 terminal that the CAP message was sent.
H245_EVENT_CAP_CFM	Send CAP acknowledge event message.
H245_CAP_DONE	The result is in.
H245_EVENT_OLC	Send OLC event message.
H245_OLC_NONE	The initial state of the OLC state machine.
H245_OLC_WAIT	In this state, an OLC message is sent, and the device is waiting for the reply.
M_H245_UCHAN_ESTABLISH_INDICATION	The OLC message received by an H245 terminal from a remote H323 endpoint.
H245_EVENT_OLC_IND	Received OLC event message.
M_H245_UCHAN_ESTAB_ACK	The OLC message acknowledgement received by an H245 terminal from a remote H323 endpoint.
H245_EVENT_OLC_CFM	Send OLC acknowledge event message.
H245_OLC_FSM	The OLC finite state machine of the H245 terminal.
H245_EVENT_OLC_CFM	Send OLC acknowledge event message.
H245_OLC_DONE	The result is in.

The debug cch323 Command with the nxe Keyword Example

The following is sample output from a **debug cch323 nxe** request:

```
Router# debug cch323 nxe

00:15:54:nxe_handle_usrmsg_to_remote:User Message size is 227
00:15:54:nxe_msg_send_possible:Msg put in the active Q for CRV [3, direction flag 0]
00:15:54:nxe_send_msg:H323chan returns bytes sent=241, the actual len=241, to IPAddr
[0xA4D4A02], Port [2517]
00:15:54:nxe_handle_usrmsg_to_remote:Usr Msg sent for IPAddr [0xA4D4A02], Port [2517], CRV
[3, direction flag 0]
00:15:54:nxe_parse_msg_from_remote:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Value of PDU flags = 0x2
00:15:54:nxe_parse_payload:Transport msg type, Payload flag = 0x0
00:15:54:nxe_receive_ack:Ack received for 1 pdu
00:15:54:nxe_receive_ack:Ack received for seqnum=13 from IPAddr [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Value of PDU flags = 0x3
00:15:54:nxe_parse_payload:Static msg type, Payload flag = 0xA0
00:15:54:nxe_parse_x_static:Rx H225 msg from IPAddr [0xA4D4A02], Port [2517], CRV [3,
direction flag 0]
00:15:54:nxe_make_ackmsg:NXE ACK Msg made to ack seqnum=14
00:15:54:nxe_send_msg:H323chan returns bytes sent=16, the actual len=16, to IPAddr
```

```
[0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Ack sent for Destination IPAddr [0xA4D4A02], Port
[2517]
00:15:54:nxe_parse_msg_from_remote:Msg received from IP [0xA4D4A02], Port [2517]
00:15:54:nxe_parse_msg_from_remote:Value of PDU flags = 0x3
00:15:54:nxe_parse_payload:Static msg type, Payload flag = 0xA0
00:15:54:nxe_parse_x_static:Rx H225 msg from IPAddr [0xA4D4A02], Port [2517], CRV [3,
direction flag 0]
```

The debug cch323 Command with the ras Keyword Example

The following is sample output from a typical **debug cch323 ras** request on a Cisco 3640 router:

```
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
```

Table 33 describes the significant fields shown in the display.

Table 33 *debug cch323 ras Command Field Descriptions*

Field	Description
CCH323_RAS_EVENT_SEND_RRQ	Send RRQ event message.
CCH323_RAS_STATE_IDLE	The global state machine is in the idle state.
CCH323_RAS_STATE_RRQ	The state machine is in the RRQ state. In this state, the gateway is registering with a gatekeeper.
RCF_CHOSEN	A registration confirm message that has been received from a gatekeeper.
CCH323_RAS_EVENT_RCF	Received RCF event message.
CCH323_RAS_EVENT_NEWCALL	New call event.
CCH323_RAS_STATE_ARQ	The per-call state machine is in the process of admitting a new call.


```

00:33:49:timer (0x81D130D4)starts - delay (15000)
00:33:49:cch323_ct_main:SOCK 1 Event 0x1
00:33:49:timer(0x81D130D4) stops
00:33:49:Near-end Pref Codecs = G.729 IETF
00:33:49: generic_open_logical_channel:codec is g729

00:33:49:cch323_generic_open_logical_channel:Filling in qosCapability field to 0

00:33:49:timer (0x81D130D4)starts - delay (15000)
00:33:49:cch323_ct_main:SOCK 1 Event 0x1
00:33:49:cch323_ct_main:SOCK 1 Event 0x1
00:33:49:      [1]townner_data=0x81D13C88, len=105, msgPtr=0x81D07608

00:33:49:cch323_gw_process_read_socket:received msg for H.225

00:33:49:timer(0x81D130D4) stops
00:33:49:timer (0x81D130D4)starts - delay (180000)
00:33:49:Codec:loc(16), rem(16),
Bytes:loc(20), Fwd(20), Rev(20)
00:33:49:cch323_rtp_open_notify:
00:33:50:cch323_ct_main:SOCK 1 Event 0x1
00:33:50:      [1]townner_data=0x81D13C88, len=71, msgPtr=0x81F1F2E0

00:33:50:cch323_gw_process_read_socket:received msg for H.225

00:33:50:cch323_caps_ind:cap_modem_proto:0, cap_modem_codec:0, cap_modem_redundancy:0
payload 100
00:33:50:cch323_caps_ind:Load DSP with Negotiated codec(16) g729r8, Bytes=20
00:33:50:cch323_caps_ind:set DSP for dtmf-relay = CC_CAP_DTMF_RELAY_INBAND_VOICE

```

The following is sample output from a typical **debug cch323** session request for a call setup on a terminating gateway:

Router# **debug cch323 session**

```

00:23:27:cch323_ct_main:SOCK 0 Event 0x1
00:23:27:cch323_ct_main:SOCK 1 Event 0x1
00:23:27:      [1]townner_data=0x81F9CA9C, len=179, msgPtr=0x81D15C6C

00:23:27:cch323_gw_process_read_socket:received msg for H.225

00:23:27:cch323_h225_receiver CCB not existing already

00:23:27:cch323_get_new_ccb:ccb (0x81F90184) is in use
00:23:27:cch323_h225_receiver Got a new CCB for call id -2115467564

00:23:27:cch323_h225_setup_ind
00:23:27:Not using Voice Class Codec

00:23:27:cch323_set_peer:peer:81FB3228, peer->voice_peer_tag:12C, ccb:81F90184
00:23:27:Near-end Pref Codecs = G.729 IETF
00:23:27:Codec:loc(16), rem(16),
Bytes:loc(20), Fwd(20), Rev(20)

00:23:27:cch323_build_fastStart_cap_response:Retrieved qosCapability of 0

00:23:27:cch323_build_fastStart_cap_response:In Response Filling in qosCapability field
to 0

00:23:27:Not using Voice Class Codec

```

Related Commands

Command	Description
clear h323 gateway	Clears the H.323 gateway counters.
debug h323-annexg	Displays all pertinent AnnexG messages that have been transmitted and received.
debug voip rawmsg	Displays the raw message owner, length, and pointer.
show h323 gateway	Displays statistics for H.323 gateway messages that have been sent and received and displays the reasons for which H.323 calls have been disconnected.

debug cch323 capacity

To track the call capacity of the gatekeeper, use the **debug cch323 capacity** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cch323 capacity

no debug cch323 capacity

Syntax Description

This command has no keywords or arguments.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was introduced.

Usage Guidelines

Use the **debug cch323 capacity** command to track the maximum and current call capacity values in the RAS messages and to debug capacity-related problems while sending RAS messages. This command is entered on the gateway to monitor the call capacity of the gatekeeper.

The command lists the values for current and maximum call capacity provided by the trunk group capacity resource manager if and when the H.323 Service Provider Interface (SPI) requests the information for all or specific groups of circuits.

Examples

The following example illustrates the debug output. Descriptions of the output follow the example.

```
Router# debug cch323 capacity
Call Capacity Information tracing is enabled

5d00h: cch323_process_carrier_update: Registered = 1,Event = 1,Reason = 1
5d00h: cch323_process_carrier_update: CarrierId = CARRIERA_NEWENGLAND
5d00h: cch323_fill_crm_CallCapacities: Reason = 1, GroupID = CARRIERA_NEWENGLAND
5d00h: Capacity Details:                Maximum Channels in Group:    23
      Max. Voice Calls(In) :           23,      Max. Voice Calls(Out):    23
      Active Voice Calls(In):           5,      Active Voice Calls(Out):    7
      Max. Voice Calls(to GK):          23,      Avail. Voice Calls(to GK):  11
```

The gatekeeper displays this output when trunk groups are added, deleted, or modified or when circuits in a trunk group are deactivated or activated (similar to ISDN layer 2 down/up).

```
5d00h: cch323_process_carrier_update: Registered = 1,Event = 1,Reason = 1
5d00h: cch323_process_carrier_update: CarrierId = CARRIERA_NEWENGLAND
```

Table 34 describes the fields shown in this section of the **debug cch323 capacity** sample output.

Table 34 *debug cch323 capacity Update Field Descriptions*

Field	Description
Registered	Gateway registration: <ul style="list-style-type: none"> 0=Gateway is not registered to the gatekeeper 1=Gateway is registered to the gatekeeper at the time of the change
Event	Carriers updated: <ul style="list-style-type: none"> 0=All carriers updated 1=Single carrier updated
Reason	Reason for the update notification: <ul style="list-style-type: none"> 0=CURRENT_CAPACITY_UPDATE 1=MAX_CAPACITY_UPDATE 2=BOTH_CAPACITY_UPDATE
CarrierID	ID of the trunk group or carrier to which the change applies.

The gatekeeper displays this output whenever call capacity information is sent to the gatekeeper.

```
5d00h: cch323_fill_crm_CallCapacities: Reason = 1, GroupID = CARRIERA_NEWENGLAND
5d00h: Capacity Details:           Maximum Channels in Group:    23
    Max. Voice Calls(In) :      23,      Max. Voice Calls(Out):    23
    Active Voice Calls(In):      5,      Active Voice Calls(Out):    7
    Max. Voice Calls(to GK):     23,      Avail. Voice Calls(to GK):  11
```

Table 35 describes the fields shown in this section of the **debug cch323 capacity** sample output:

Table 35 *debug cch323 capacity Call Capacity Field Descriptions*

Field	Description
GroupID	The circuit's carrier identification (ID) or trunk group label.
Maximum Channels in Group	Maximum number of physical (or configured) circuits.
Max. Voice Calls(In)	Maximum number of allowed incoming voice and data calls.
Max. Voice Calls(Out)	Maximum number of allowed outgoing voice and data calls.
Active Voice Calls(In)	Current number of active incoming voice and data calls.
Active Voice Calls(Out)	Current number of active outgoing voice and data calls.
Max. Voice Calls(to GK)	Maximum call capacity value to be sent to the gatekeeper in the RAS message.
Avail. Voice Calls(to GK)	Available call capacity value to be sent to the gatekeeper in the RAS message.

Related Commands

Command	Description
endpoint circuit-id h323id	Associates a carrier with a non-Cisco endpoint.

debug cch323 h225

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

```
debug cch323 h225
```

```
no debug cch323 h225
```

Syntax Description This command has no keywords or arguments.

Defaults Disabled

Command Modes Privileged EXEC

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

Usage Guidelines

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 ready to receive an incoming IP call.
- **H225_SETUP**—This is the call setup state. The state machine transitions 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 transitions 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 transitions 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 the H.225 state machine to send an *alerting* message to the peer.
- H225_EVENT_ALERT_IND—This event indicates the H.225 state machine that an *alerting* message is received from the peer.
- H225_EVENT_CALLPROC—This event indicates the H.225 state machine to send a *call proceeding* message to the peer.
- H225_EVENT_CALLPROC_IND—This event indicates the H.225 state machine that a *call proceeding* message is received from the peer.
- H225_EVENT_REJECT—This event indicates the H.225 state machine to reject the *call setup* request from the peer.
- H225_EVENT_REJECT_IND—This event indicates the H.225 state machine that a *call setup* request to the peer is rejected.
- H225_EVENT_RELEASE—This event indicates the H.225 state machine to send a *release complete* message to the peer.
- H225_EVENT_RELEASE_IND—This event indicates the H.225 state machine that a *release complete* message is received from the peer.
- H225_EVENT_SETUP—This event indicates the H.225 state machine to send a *setup* message to the peer.
- H225_EVENT_SETUP_IND—This event indicates the H.225 state machine that a *setup* message is received from the peer.
- H225_EVENT_SETUP_CFM—This event indicates the H.225 state machine to send a *connect* message to the peer.
- H225_EVENT_SETUP_CFM_IND—This event indicates the H.225 state machine that a *connect* message from the peer.
- H225_EVENT_RAS_SUCCESS—This event indicates the H.225 state machine that the pending RAS operation is successful.
- H225_EVENT_RAS_FAILED—This event indicates the H.225 state machine that the pending RAS operation failed.
- H225_EVENT_H245_SUCCESS—This event indicates the H.225 state machine that the pending H.245 operation is successful.
- H225_EVENT_H245_FAILED—This event indicates the H.225 state machine that the pending H.245 operation failed.

Examples

The following is sample 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 provide the trace of the state transition of the H.245 state machine based on the processed events, use the **debug cch323 h245** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug cch323 h245
```

```
no debug cch323 h245
```

Syntax Description This command has no keywords or arguments.

Defaults Disabled

Command Modes Privileged EXEC

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

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

- Master SlaveDetermination (MSD) state machine
- Capability Exchange (CAP) state machine
- Open Logical Channel (OLC) state machine

State Definitions

The definitions are listed 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, waiting for the reply.
- H245_MS_DONE— The result is in.
- H245_CAP_NONE—This is the initial state of the capabilities exchange state machine.
- H245_CAP_WAIT—In this state, a *cap exchange message* is sent, 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, waiting for reply.
- H245_OLC_DONE: OLC done.

Event definitions

- 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
- 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 from 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 preauth

To enable diagnostic reporting of authentication, authorization, and accounting (AAA) call preauthentication for H.323 calls, use the **debug cch323 preauth** command in privileged EXEC mode. To disable diagnostic reporting, use the **no** form of this command.

debug cch323 preauth

no debug cch323 preauth

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(11)T	This command was introduced.

Examples The following example shows debug output for a single H.323 call:

```
Router# debug cch323 preauth

CCH323 preauth tracing is enabled
cch323_is_preauth_reqd is TRUE
Jan 23 18:39:56.393: In cch323_send_preauth_req for preauth_id = -1
Jan 23 18:39:56.393: Entering rpms_proc_print_preauth_req

Jan 23 18:39:56.393: Request = 0
Jan 23 18:39:56.393: Preauth id = 86514
Jan 23 18:39:56.393: EndPt Type = 1
Jan 23 18:39:56.393: EndPt = 192.168.81.102
Jan 23 18:39:56.393: Resource Service = 1
Jan 23 18:39:56.393: Call_origin = answer
Jan 23 18:39:56.393: Call_type = voip
Jan 23 18:39:56.393: Calling_num = 2230001
Jan 23 18:39:56.393: Called_num = 1#1130001
Jan 23 18:39:56.393: Protocol = 0
Jan 23 18:39:56.393: cch323_insert_preauth_tree:Created node with preauth_id = 86514 ,ccb
6852D5BC , node 651F87FC
Jan 23 18:39:56.393:rpms_proc_create_node:Created node with preauth_id = 86514
Jan 23 18:39:56.393:rpms_proc_send_aaa_req:uid got is 466725
Jan 23 18:39:56.397:rpms_proc_preauth_response:Context is for preauth_id 86514, aaa_uid
466725
Jan 23 18:39:56.397: Entering Function cch323_rpms_proc_callback_func
```

```
Jan 23 18:39:56.397:cch323_rpms_proc_callback_func:PREAUTH_SUCCESS for preauth id 86514
aaa_uid 466725 auth_serv 1688218168
```

```
Jan 23 18:39:56.397:rpms_proc_preauth_response:Deleting Tree node for preauth id 86514 uid
466725
```

```
Jan 23 18:39:56.397:cch323_get_ccb_and_delete_from_preauth_tree:Preauth_id=86514
cch323_get_ccb_and_delete_from_preauth_tree:651F87FC node and 6852D5BC ccb
```

Table 36 describes the significant fields shown in the display.

Table 36 *debug cch323 preauth Command Field Descriptions*

Field	Description
Request	Request Type—0 for preauthentication, 1 for disconnect.
Preauth id	Identifier for the preauthentication request.
EndPt Type	Call Origin End Point Type—1 for IP address, 2 for IZCT value.
EndPt	Call Origin End Point Value—An IP address or IZCT value.
Resource Service	Resource Service Type—1 for Reservation, 2 for Query.
Call_origin	Answer.
Call_type	Voice over IP (VoIP).
Calling_num	Calling Party Number (CLID).
Called_num	Called Party Number (DNIS).
Protocol	0 for H.323, 1 for SIP.
function reports	Various identifiers and status reports for executed functions.

debug cch323 ras

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

debug cch323 ras

no debug cch323 ras

Syntax Description This command has no keywords or arguments.

Defaults Disabled

Command Modes Privileged EXEC

Command History

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

Usage Guidelines

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 follow:

- 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 in the process of discovering a gatekeeper.
- CCH323_RAS_STATE_RRQ—The state machine is in the RRQ state. In this state, the gateway is in the process of 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

These are the event definitions of the different states of the RAS state machine:

- 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 example shows debug output for the **debug cch323 preauth** command:

```
Router# debug cch323 preauth

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 ccm-manager

To display debugging information about the Cisco CallManager (CCM), use the **debug ccm-manager** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug ccm-manager {backhaul {events | errors} | config-download {all | errors | events |
packets | xml} | errors | events | music-on-hold {errors | events | packets} | packets}
```

```
no debug ccm-manager
```

Syntax Description	
backhaul	(Optional) Enables debugging of the CCM backhaul. The keywords are as follows: <ul style="list-style-type: none"> events—Displays CCM backhaul events. errors—Displays CCM backhaul errors.
config-download	Enables debugging of the CCM configuration download. The keywords are as follows: <ul style="list-style-type: none"> all—Displays all CCM configuration parameters. errors—Displays CCM configuration errors. events—Displays CCM configuration events. packets—Displays CCM configuration packets. xml—Displays the CCM configuration eXtensible Markup Language (XML) parser.
errors	(Optional) Displays errors related to CCM.
events	(Optional) Displays CCM events, such as when the primary CCM server fails and control is switched to the backup CCM server.
music-on-hold	(Optional) Displays music-on-hold (MOH). The keywords are as follows: <ul style="list-style-type: none"> errors—Displays MOH errors. events—Displays MOH events. packets—Displays MOH packets.
packets	(Optional) Displays CCM packets.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.1(3)T	This command was introduced for Cisco CallManager Version 3.0 and the Cisco Voice Gateway 200 (Cisco VG200).
	12.2(2)XA	This command was implemented on Cisco 2600 series and Cisco 3600 series routers.

Release	Modification
12.2(2)XN	Support for enhanced Media Gateway Control Protocol (MGCP) voice gateway interoperability was added to Cisco CallManager Version 3.1 for the Cisco 2600 series routers, Cisco 3600 series routers, and the Cisco Voice Gateway 200 (Cisco VG200).
12.2(11)T	This command was integrated into the Cisco IOS Release 12.2(11)T and implemented on the following platforms: Cisco IAD2420 series,

Examples

The following is sample output from the **debug ccm-manager events** command:

```
Router# debug ccm-manager events
```

```
*Feb 28 22:56:05.873: cmapp_mgcpapp_go_down: Setting mgc status to NO_RESPONSE
*Feb 28 22:56:05.873: cmapp_host_fsm: New state DOWN for host 0 (172.20.71.38)
*Feb 28 22:56:05.873: cmapp_mgr_process_ev_active_host_failed: Active host 0
(172.20.71.38) failed
*Feb 28 22:56:05.873: cmapp_mgr_check_hostlist: Active host is 0 (172.20.71.38)
*Feb 28 22:56:05.877: cmapp_mgr_switchover: New actv host will be 1 (172.20.71.44)
*Feb 28 22:56:05.877: cmapp_host_fsm: Processing event GO_STANDBY for host 0
(172.20.71.38) in state DOWN
*Feb 28 22:56:05.877: cmapp_open_new_link: Open link for [0]:172.20.71.38
*Feb 28 22:56:05.877: cmbh_open_tcp_link: Opening TCP link with Rem IP 172.20.71.38, Local
IP 172.20.71.19, port 2428
*Feb 28 22:56:05.881: cmapp_open_new_link: Open initiated OK: Host 0 (172.20.71.38),
session_id=8186DEE4
*Feb 28 22:56:05.881: cmapp_start_open_link_tmr: Host 0 (172.20.71.38), tmr 0
*Feb 28 22:56:05.881: cmapp_host_fsm: New state STANDBY_OPENING for host 0 (172.20.71.38)
*Feb 28 22:56:05.881: cmapp_host_fsm: Processing event GO_ACTIVE for host 1 (172.20.71.44)
in state STANDBY_READY
*Feb 28 22:56:05.885: cmapp_mgr_send_rehome: new addr=172.20.71.44,port=2427
*Feb 28 22:56:05.885: cmapp_host_fsm: New state REGISTERING for host 1 (172.20.71.44)
```

[Table 37](#) describes the significant fields shown in the display.

Table 37 debug ccm-manager Command Field Description

Field	Description
<i>nn:nn:nn</i>	Time stamp that indicates when the Cisco CallManager event occurred.
CMAPP: <i>error message</i>	The Cisco CallManager routine in which the error event occurred.

Related Commands

Command	Description
show ccm-manager	Displays a list of Cisco CallManager servers, their current status, and their availability.

debug ccsip all

To enable all Session Initiation Protocol (SIP)-related debugging, use the **debug ccsip all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip all

no debug ccsip all

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

12.1(1)T	This command was introduced.
12.1.(3)T	The output of this command was changed..
12.2(2)XA	Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
12.2(2)XB1	This command was implemented on the Cisco AS5850 universal gateway.
12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

Usage Guidelines

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

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

Examples

The following example displays debug output from one side of the call:

```
Router# debug ccsip all

All SIP call tracing enabled
Router1#
*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_sentininvite_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_REC'D_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: HandleSIPlxxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec      : g711ulaw , bytes :160
Inband Alerting      : 0

*Mar 6 14:10:42: 0x624CFEF8 : State change from (STATE_REC'D_PROCEEDING,
SUBSTATE_PROCEEDING_PROCEEDING) to (STATE_REC'D_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_REC'D_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 Payout: 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

```

The following example displays debug output from the other side of the call:

```
Router# 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_REC'D_INVITE, SUBSTATE_REC'D_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_REC'D_INVITE,
SUBSTATE_REC'D_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 Payout: 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	Shows all SIP SPI call tracing.
debug ccsip error	Shows SIP SPI errors.
debug ccsip events	Shows all SIP SPI events tracing.
debug ccsip messages	Shows all SIP SPI message tracing.
debug ccsip states	Shows all SIP SPI state tracing.

debug ccsip calls

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) call tracing, use the **debug ccsip calls** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip calls

no debug ccsip calls

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

12.1(1)T	This command was introduced.
12.1(3)T	The output of this command was changed..
12.2(2)XA	Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
12.2(2)XB1	This command was introduced on the Cisco AS5850 universal gateway.
12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

Usage Guidelines

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

Examples

The following example displays debug output from one side of the call:

```
Router1# debug ccsip calls

SIP Call statistics tracing is enabled
Router1#
*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

```

The following example displays debug output from the other side of the call:

Router2# **debug ccsip calls**

SIP Call statistics tracing is enabled

Router2#

*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	Shows SIP SPI errors.
debug ccsip events	Shows all SIP SPI events tracing.
debug ccsip messages	Shows all SIP SPI message tracing.
debug ccsip states	Shows all SIP SPI state tracing.

debug ccsp error

To show Session Initiation Protocol (SIP) Service Provider Interface (SPI) errors, use the **debug ccsp error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsp error

no debug ccsp error

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History		
12.1(1)T		This command was introduced.
12.1(3)T		The output of this command was changed..
12.2(2)XA		Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
12.2(2)XB1		This command was implemented on the Cisco AS5850 universal gateway.
12.2(8)T		This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.
12.2(11)T		This command was integrated into Cisco IOS Release 12.2(11)T.

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

Examples The following example displays debug output from one side of the call:

```
Router1# debug ccsp error
```

```
SIP Call error tracing is enabled
```

```
Router1#
```

```
*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: HandleSIPlxxRinging: 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
```

The following example displays debug output from the other side of the call:

```
Router2# debug ccsip error

SIP Call error tracing is enabled
Router2#
*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	Shows all SIP SPI call tracing.
debug ccsip events	Shows all SIP SPI events tracing.
debug ccsip messages	Shows all SIP SPI message tracing.
debug ccsip states	Shows all SIP SPI state tracing.

debug ccsip events

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) events tracing, use the **debug ccsip events** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip events

no debug ccsip events

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Version	Description
	12.1(1)T	This command was introduced.
	12.1.(3)T	The output of this command was changed..
	12.2(2)XA	Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
	12.2(2)XB1	This command was implemented on the Cisco AS5850 universal gateway.
	12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.
	12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

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

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

```
Router1# debug ccsip events
```

```
SIP Call events tracing is enabled
```

```
Router1#
```

```
*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
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Shows all SIP SPI call tracing.
debug ccsip error	Shows SIP SPI errors.
debug ccsip messages	Shows all SIP SPI message tracing.
debug ccsip states	Shows all SIP SPI state tracing.

debug ccsip messages

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) message tracing, use the **debug ccsip messages** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip messages

no debug ccsip messages

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Version	Description
	12.1(1)T	This command was introduced.
	12.1.(3)T	The output of this command was changed..
	12.2(2)XA	Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
	12.2(2)XB1	This command was implemented on the Cisco AS5850 universal gateway.
	12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.
	12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

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

Examples The following example shows debug output from one side of the call:

```
Router1# debug ccsip messages

SIP Call messages tracing is enabled
Router1#
*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

```

The following example show debug output from the other side of the call:

```
Router2# debug ccsip messages
```

```

SIP Call messages tracing is enabled
Router2#
*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
```

debug ccsip messages

```

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

debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Shows all SIP SPI call tracing.
debug ccsip error	Shows SIP SPI errors.
debug ccsip events	Shows all SIP SPI events tracing.
debug ccsip states	Shows all SIP SPI state tracing.

debug ccsip preauth

To enable diagnostic reporting of authentication, authorization, and accounting (AAA) preauthentication for Session Initiation Protocol (SIP) calls, use the **debug ccsip preauth** command in privileged EXEC mode. To disable diagnostic reporting, use the **no** form of this command.

debug ccsip preauth

no debug ccsip preauth

Syntax Description This command has no arguments or keywords.

Defaults No default behavior or values.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(11)T	This command was introduced.

Examples The following example shows debug output for a single SIP call:

```
Router# debug ccsip preauth

SIP Call preauth tracing is enabled
Jan 23 18:43:17.898::Preauth Required
Jan 23 18:43:17.898: In sipSPISendPreauthReq for preauth_id = 86515, ccb = 67AF4E10
Jan 23 18:43:17.898: Entering rpms_proc_print_preauth_req

Jan 23 18:43:17.898: Request = 0
Jan 23 18:43:17.898: Preauth id = 86515
Jan 23 18:43:17.898: EndPt Type = 1
Jan 23 18:43:17.898: EndPt = 192.168.80.70
Jan 23 18:43:17.898: Resource Service = 1
Jan 23 18:43:17.898: Call_origin = answer
Jan 23 18:43:17.898: Call_type = voip
Jan 23 18:43:17.898: Calling_num = 2270001
Jan 23 18:43:17.898: Called_num = 1170001
Jan 23 18:43:17.898: Protocol = 1
Jan 23 18:43:17.898:sipSPISendPreauthReq:Created node with preauth_id = 86515, ccb
67AF4E10 , node 6709C280
Jan 23 18:43:17.898:rpms_proc_create_node:Created node with preauth_id = 86515
Jan 23 18:43:17.898:rpms_proc_send_aaa_req:uid got is 466728
Jan 23 18:43:17.902:rpms_proc_preauth_response:Context is for preauth_id 86515, aaa_uid
466728
Jan 23 18:43:17.902:rpms_proc_preauth_response:Deleting Tree node for preauth id 86515 uid
466728
Jan 23 18:43:17.902:sipSPIGetNodeForPreauth:Preauth_id=86515
```

```

Jan 23 18:43:17.902: ccsip_spi_process_preauth_event:67AF4E10 ccb & 6709C280 node
Jan 23 18:43:17.902: In act_preauth_response:67AF4E10 ccb
Jan 23 18:43:17.902: act_preauth_response:Deleting node 6709C280 from tree

```

Table 38 describes the significant fields shown in the display.

Table 38 *debug ccsip preauth Command Field Descriptions*

Field	Description
Request	Request Type—0 for preauthentication, 1 for disconnect.
Preauth id	Identifier for the preauthentication request.
EndPt Type	Call Origin End Point Type—1 for IP address, 2 for Interzone ClearToken (IZCT) value.
EndPt	Call Origin End Point Value—An IP address or IZCT value.
Resource Service	Resource Service Type—1 for Reservation, 2 for Query.
Call_origin	Answer.
Call_type	Voice over IP (VoIP).
Calling_num	Calling Party Number (CLID).
Called_num	Called Party Number (DNIS).
Protocol	0 for H.323, 1 for SIP.
function reports	Various identifiers and status reports for executed functions.

debug ccsip states

To show all Session Initiation Protocol (SIP) Service Provider Interface (SPI) state tracing, use the **debug ccsip states** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccsip states

no debug ccsip states

Syntax Description

This command has no arguments or keywords.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

12.1(1)T	This command was introduced.
12.2(2)XA	Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.
12.2(2)XB1	This command was implemented on the Cisco AS5850 universal gateway.
12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.
12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T.

Usage Guidelines

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

Examples

The following example shows all SIP SPI state tracing:

```
Router1# debug ccsip states

SIP Call states tracing is enabled
Router1#
*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_REC'D_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
*Jan 2 18:34:37.853:0x6220C634 :State change from (STATE_REC'D_PROCEEDING,
SUBSTATE_PROCEEDING_PROCEEDING) to (STATE_REC'D_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
```

■ **debug ccsip states**

```
*Jan 2 18:34:38.261:0x6220C634 :State change from (STATE_REC'D_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
```

Related Commands

Command	Description
debug ccsip all	Enables all SIP-related debugging.
debug ccsip calls	Shows all SIP SPI call tracing.
debug ccsip error	Shows SIP SPI errors.
debug ccsip events	Shows all SIP SPI events tracing.
debug ccsip messages	Shows all SIP SPI message tracing.

debug ccswwoice vo-debug

To display the ccswwoice function calls during call setup and teardown, use the **debug ccswwoice voo-debug** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug ccswwoice voatm-debug
```

```
no debug ccswwoice 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 ccswwoice vo-session** command, but includes additional debugging information relating to the calls.

Examples

The following example shows sample output from the **debug ccswwoice voo-debug** command:

```
Router# debug ccswwoice voo-debug

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

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

Related Commands

Command	Description
debug ccswwoice voo-debug	Displays the ccswwoice function calls during call setup and teardown.

debug ccswwoice vo-session

To display the ccswwoice function calls during call setup and teardown, use the **debug ccswwoice vo-session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccswwoice vo-session

no debug ccswwoice 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 ccswwoice vo-session** command:

```
Router# debug ccswwoice vo-session

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

Related Commands	Command	Description
	debug call rsvp-sync events	Displays the ccswwoice function calls during call setup and teardown.

debug ccswwoice vofr-debug

To display the ccswwoice function calls during call setup and teardown, use the **debug ccswwoice vofr-debug** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug ccswwoice vofr-debug
```

```
no debug ccswwoice 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 ccswwoice vofr-session** command, but includes additional debugging information relating to the calls.

Examples

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

```
Router# debug ccswwoice 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 cch323	Displays the ccfrf11 function calls during call setup and teardown.
	debug ccsvoice vo-debug	Displays the ccsvoice 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 ccswwoice vofr-session

To display the ccswwoice function calls during call setup and teardown, use the **debug ccswwoice vofr-session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug ccswwoice vofr-session

no debug ccswwoice 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 ccswwoice vofr-session** command:

```
Router# debug ccswwoice 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 cch323	Displays the ccrf11 function calls during call setup and teardown.
debug call rsvp-sync events	Displays the ccswwoice 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	events
	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.	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 =
003909     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 =
003909 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. To disable debugging output, use the **no** form of this command.

```
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. To disable debugging output, use the **no** form of this command.

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 10.0.0.1.

```
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 ces-conn

To display information from circuit emulation service (CES) clients, use the **debug ces-conn** command in privileged EXEC mode. To disable debugging output, use the **no** form of the command.

```
debug ces-conn [all | errors | events]
```

```
no debug ces-conn
```

Syntax Description

all (Optional)	Displays all error and event information.
errors (Optional)	Displays only error information.
events (Optional)	Displays only event information.

Defaults

No default behavior or values.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(5)XM	This command is supported on Cisco 3600 series routers.
12.2(4)T	This command was integrated into Cisco IOS Release 12.2(4)T.

Examples

The following example shows debug output for a CES connection:

```
Router# debug ces-conn all
CES all debugging is on
Router#

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.

Router(config)# connect conn1 t1 3/0 1 atm1/0 1/100

Router(config-ces-conn)# exit
Router(config)#
*Mar  6 18:32:27:CES_CLIENT:vc QoS parameters are PCR = 590, CDV =
5000, CAS_ENABLED = 1,partial fill = 0, multiplier = 8, cbr rate = 64,
clock recovery = 0,service_type = 3, error method = 0,sdt_size = 196,
billing count = 0
*Mar  6 18:32:27:CES_CLIENT:attempt 1 to activate segment>
```

debug cgma

To turn on debugging for the Cisco Gateway Management Agent (CGMA), use the **debug cgma** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug cgma

no debug cgma

Syntax Description This command has no arguments or keywords.

Defaults Debugging is turned off by default.

Command Modes Privileged EXEC

Command History	Release	Modification
	12.2(2)XB	This command was introduced on the Cisco 2600 series, Cisco 3600 series, Cisco AS5300, Cisco AS5350, and the Cisco AS5400.
	12.2(2)XB1	This command was implemented on the Cisco AS5800 for this Cisco IOS release only.
	12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T for the Cisco 2600 series, the Cisco 3600 series, and the Cisco 7200 series. Note This command is not supported on the Cisco AS5300, Cisco AS5350, Cisco AS5400, Cisco AS5800, and Cisco AS5850 in this release.
	12.2(11)T	This command was integrated into Cisco IOS Release 12.2(11)T and support was added for the Cisco AS5300, Cisco AS5350, Cisco AS5400, Cisco AS5800, and Cisco AS5850 platforms.

Examples The following example shows that debugging is turned on:

```
Router# debug cgma

cgma debugging is on
```

The following example shows debug output from a Cisco IOS gateway connected to the third-party management system:

```
Router# debug cgma

Router#
*Mar 1 00:02:02:cgma_timer_start:timer_type=0
*Mar 1 00:02:02:cgma_timer_start:timer for fd 1 started
*Mar 1 00:02:02:CGMA RECV :new ARMA connectionn from 10.1.1.200 on socket fd = 1
*Mar 1 00:02:02:CGMA RECV :waiting on socket_recv() on an CGMA APP socket
*Mar 1 00:02:02:CGMA RECV :socket fd = 1 receive a string :
```

```

<handshake_q hname="client" pname="pj" version="2.1">
  msg is <handshake_q hname="client" pname="pj" version="2.1">

*Mar 1 00:02:02:retrieve_msg_and_enqueue

*Mar 1 00:02:02:parser_msg_and_actioncgma_get_tag
  i = 0
tag is 0

*Mar 1 00:02:02:parser_msg_and_action:found handshake timer, state = 1, fd = 1
*Mar 1 00:02:02:cgma_timer_stop:timer_type=0
*Mar 1 00:02:02:cgma_timer_stop():stop timer for fd 1
*Mar 1 00:02:02:cgma_timer_start:timer_type=1
*Mar 1 00:02:02:cgma_timer_start:timer for fd 1 started attributes = 3
  The Length of the message is 50
  The message is handshake_q hname="client" pname="pj" version="2.1"
Tag:*ptr is h
Tag:*ptr is n
Tag:*ptr is a
Tag:*ptr is m
Tag:*ptr is e
cgma_find_attribute:tag=|hname| index = 0
  value *ptr is c
  value *ptr is l
  value *ptr is i
  value *ptr is e
  value *ptr is n
  value *ptr is t
cgma_find_attribute:tag=|hname| index = 0
cgma_store_attribute
Tag:*ptr is p
Tag:*ptr is n
Tag:*ptr is a
Tag:*ptr is m
Tag:*ptr is e
cgma_find_attribute:tag=|pname| index = 0
  value *ptr is p
  value *ptr is j
cgma_find_attribute:tag=|pname| index = 0
cgma_store_attribute
Tag:*ptr is v
Tag:*ptr is e
Tag:*ptr is r
Tag:*ptr is s
Tag:*ptr is i
Tag:*ptr is o
Tag:*ptr is n
cgma_find_attribute:tag=|version| index = 0
  value *ptr is 2
  value *ptr is .
  value *ptr is 1
cgma_find_attribute:tag=|version| index = 0
cgma_store_attribute
cgma_get_params: ret is 0

*Mar 1 00:02:02:cgma_strtok
*Mar 1 00:02:02:Outgoing Handshake Message is <handshake_r hname="Router_A" pname="CGMA
Agent" version="12.2(6.8)T2,">

*Mar 1 00:02:02:cgma_send_arma_msgcgma_send_arma_msg:Message (size 77) Enqueued is
<handshake_r hname="Router_A" pname="CGMA Agent" version="12.2(6.8)T2,">

```

```
*Mar 1 00:02:02:cgma_send_msg() :sending (76) bytes ...  
<handshake_r hname="Router_A" pname="CGMA Agent" version="12.2(6.8)T2,">  
  
*Mar 1 00:02:02:cgma_send_msg_to_one_conn :fd(1) sending (76) bytes ...  
<handshake_r hname="Router_A" pname="CGMA Agent" version="12.2(6.8)T2,">  
  
*Mar 1 00:02:02:socket_send() :ret :76, errno :0>> loop = 20
```

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. To disable debugging output, use the **no** form of this command.

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 **show 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. To disable debugging output, use the **no** form of this command.

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(Channel13/2-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:

```
Channel13/2: 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. To disable debugging output, use the **no** form of this command. **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 11, but no cip_info
```

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

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

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

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

Related Commands

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. To disable debugging output, use the **no** form of this command.

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 39](#) describes the significant fields in the display.

Table 39 *debug channel packets Command 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. To disable debugging output, use the **no** form of this command.

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 isis 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 isis packets** privileged EXEC command. To disable debugging output, use the **no** form of this command.

debug clns isis packets

no debug clns isis packets

Syntax Description

This command has no arguments or keywords.

Examples

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

```
Router# debug clns isis 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.ef00.0000.0001.5940.1600.0906.4023.00, 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. To disable debugging output, use the **no** form of this command.

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
```