debug backhaul-session-manager session through debug channel packets

- debug backhaul-session-manager session through debug channel packets, page 1

default backhaul-session-manager session through debug channel packets
debug backhaul-session-manager session

To debug all the available sessions or a specified session, use the **debug backhaul-session-manager session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug backhaul-session-manager session \{state| xport\} \{all| session-id\}
```

```
no debug backhaul-session-manager session \{state| xport\} \{all| session-id\}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| **state** | Shows information about state transitions. Possible states are as follows:  
SESS_SET_IDLE: A session-set has been created.  
SESS_SET_OOS: A session(s) has been added to session-group(s). No ACTIVE notification has been received from Virtual Switch Controller (VSC).  
SESS_SET_ACTIVE_IS: An ACTIVE notification has been received over one in-service session-group. STANDBY notification has not been received on any available session-group(s).  
SESS_SET_STANDBY_IS: A STANDBY notification is received, but there is no in-service active session-group available.  
SESS_SET_FULL_IS: A session-group in-service that has ACTIVE notification and at least one session-group in-service that has STANDBY notification.  
SESS_SET_SWITCH_OVER: An ACTIVE notification is received on session-group in-service, which had received STANDBY notification.  |
| **xport** | Provides traces for all packets (protocol data units (PDUs)), application PDUs, and also session-manager messages. |
| **all** | All available sessions. |
| **session-id** | A specified session. |

### Command Default

Debugging for backhaul-session-manager session is not enabled.

### Command Modes

Privileged EXEC
debug backhaul-session-manager session through debug channel packets

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>Support for this command was introduced on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 2600 series, Cisco 3600 series, and Cisco MC3810.</td>
</tr>
<tr>
<td>12.2(2)XB</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was implemented on Cisco IAD2420 series integrated access devices (IADs). This command is not supported on the access servers in this release.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Examples

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

```
Router# debug backhaul-session-manager session all
```
```
23:49:14:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:19:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:24:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:29:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:34:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:49:34:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 1 (CONN-FAILED)
```

The following example displays output for the `debug backhaul-session-managersession state all` command:

```
Router# debug backhaul-session-manager session state all
```
The following example displays output for the `debug backhaul-session-manager session xport all` command:

```
Router# debug bsm_command:DEBUG_BSM_SESSION_STATE_ALL
```

The following example displays output for the `debug backhaul-session-manager session xport all` command:

```
Router# debug backhaul-session-manager session xport all Router#
debug_bsm_command:DEBUG_BSM_SESSION_XPORT
23:51:39:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
23:51:42:SESSION:XPORT:sig rcvd. session = 33, connid = 0x80BA14EC, sig = 5 (CONN-RESET)
23:51:44:SESSION:XPORT:sig rcvd. session = 34, connid = 0x80BA12FC, sig = 5 (CONN-RESET)
```

Use caution when enabling this debug command in a live system. It produces significant amounts of output, which could lead to a disruption of service.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug backhaul-session-manager set</code></td>
<td>Traces state changes and receives messages and events for all available session-sets or a specified session-set.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager set

To trace state changes and receive messages and events for all the available session sets or a specified session set, use the `debug backhaul-session-manager set` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug backhaul-session-manager set {all| name set-name}
no debug backhaul-session-manager set {all| name set-name}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>all</th>
<th>All available session sets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>name set-name</td>
<td></td>
<td>A specified session set.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging for backhaul session sets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>Support for this command was introduced on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 2600 series, Cisco 3600 series, and Cisco MC3810.</td>
</tr>
<tr>
<td>12.2(2)XB</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco IAD2420 series integrated access devices (IADs). This command is not supported on the access servers in this release.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was implemented on Cisco AS5350, Cisco AS5400, and Cisco AS5850 platforms.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>
The following is output for the `debug backhaul-session-manager set all` command for all available session sets:

```
Router# debug backhaul-session-manager set all
Router# debug bsm_command:DEBUG_BSM_SET_ALL
    Function set_proc_event() is called
Session-Set :test-set
Old State :BSM_SET_OOS
New State :BSM_SET_OOS
Active-Grp :NONE
    Session-Grp :g-11
Old State :Group-None
New State :Group-None
Event rcvd :EVT_GRP_INS
BSM:Event BSM_SET_UP is sent to user
Session-Set :test-set
Old State :BSM_SET_OOS
New State :BSM_SET_OOS
Active-Grp :g-11
    Session-Grp :g-11
Old State :Group-None
New State :Group-Active
Event rcvd :BSM_ACTIVE_TYPE
```

The following is output for the `debug backhaul-session-manager set name set1` command:

```
Router# debug backhaul-session-manager set name set1
Router# debug bsm_command:DEBUG_BSM_SET_NAME
Router# Function set_proc_event() is called
Session-Set :test-set
Old State :BSM_SET_OOS
New State :BSM_SET_OOS
Active-Grp :NONE
    Session-Grp :g-11
Old State :Group-None
New State :Group-None
Event rcvd :EVT_GRP_INS
Router# BSM:Event BSM_SET_UP is sent to user
Session-Set :test-set
Old State :BSM_SET_OOS
New State :BSM_SET_OOS
Active-Grp :g-11
    Session-Grp :g-11
Old State :Group-None
New State :Group-Active
Event rcvd :BSM_ACTIVE_TYPE
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug backhaul-session-manager session</code></td>
<td>Debugs all available sessions or a specified session.</td>
</tr>
</tbody>
</table>
debug backup

To monitor the transitions of an interface going down then back up, use the debug backup command in privileged EXEC mode. To disable debugging output, use the no form of this command.

debug backup
no debug backup

Syntax Description
This command has no arguments or keywords.

Command Default
No default behavior or values.

Command Modes
Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

Usage Guidelines
The debug backup command is useful for monitoring dual X.25 interfaces configured as primary and backup in a Telco data communication network (DCN).

Examples
The following example shows how to start the debug backup command:

Router# debug backup

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup active interface</td>
<td>Activates primary and backup lines on specific X.25 interfaces.</td>
</tr>
<tr>
<td>show backup</td>
<td>Displays interface backup status.</td>
</tr>
</tbody>
</table>
**debug bert**

To display information on the bit error rate testing (BERT) feature, use the `debug bert` command in privileged EXEC mode. To disable debugging output, use the `no debug bert` form of this command.

```plaintext
debug bert
no debug bert
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(2)XD</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

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

**Examples**

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

```plaintext
Router# debug bert
Bit Error Rate Testing debugging is on
Router# no debug bert
Bit Error Rate Testing debugging is off
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bert abort</td>
<td>Aborts a bit error rate testing session.</td>
</tr>
<tr>
<td>bert controller</td>
<td>Starts a bit error rate test for a particular port on a Cisco AS5300 router.</td>
</tr>
<tr>
<td>bert profile</td>
<td>Sets up various bit error rate testing profiles.</td>
</tr>
</tbody>
</table>
debug bfd

To display debugging messages about Bidirectional Forwarding Detection (BFD), use the `debug bfd` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

Cisco IOS Release 12.2(18)SXE, 12.4(4)T, and 12.2(33)SRA

```
depth bfd {event| packet [ip-address| ipv6-address]}
no debug bfd {event| packet [ip-address| ipv6-address]}
```

Cisco IOS Release 12.0(31)S

```
depth bfd {event| packet [ip-address] | ipc-error| ipc-event| oir-error| oir-event}
no debug bfd {event| packet [ip-address] | ipc-error| ipc-event| oir-error| oir-event}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>event</code></td>
<td>Displays debugging information about BFD state transitions.</td>
</tr>
<tr>
<td><code>packet</code></td>
<td>Displays debugging information about BFD control packets.</td>
</tr>
<tr>
<td><code>ip-address</code></td>
<td>(Optional) Displays debugging information about BFD only for the specified IP address.</td>
</tr>
<tr>
<td><code>ipv6-address</code></td>
<td>(Optional) Displays debugging information about BFD only for the specified IPv6 address.</td>
</tr>
<tr>
<td><code>ipc-error</code></td>
<td>(Optional) Displays debugging information with interprocess communication (IPC) errors on the Route Processor (RP) and line card (LC).</td>
</tr>
<tr>
<td><code>ipc-event</code></td>
<td>(Optional) Displays debugging information with IPC events on the RP and LC.</td>
</tr>
<tr>
<td><code>oir-error</code></td>
<td>(Optional) Displays debugging information with online insertion and removal (OIR) errors on the RP and LC.</td>
</tr>
<tr>
<td><code>oir-event</code></td>
<td>(Optional) Displays debugging information with OIR events on the RP and LC.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC
**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(18)SXÉ</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(31)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(31)S.</td>
</tr>
<tr>
<td>12.4(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(4)T.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was modified. Support for IPv6 was added.</td>
</tr>
<tr>
<td>15.1(2)T</td>
<td>This command was modified. Support for IPv6 was added to Cisco IOS Release 15.1(2)T.</td>
</tr>
<tr>
<td>15.1(1)SG</td>
<td>This command was integrated into Cisco IOS Release 15.1(1)SG.</td>
</tr>
<tr>
<td>15.1(1)SY</td>
<td>This command was modified. Support for IPv6 was added to Cisco IOS Release 15.1(1)SY.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bfd` command can be used to troubleshoot the BFD feature.

Because BFD is designed to send and receive packets at a very high rate of speed, consider the potential effect on system resources before enabling this command, especially if there are a large number of BFD peers. The `debug bfd packet` command should be enabled only on a live network at the direction of Cisco Technical Assistance Center personnel.

**Examples**

The following example shows output from the `debug bfd packet` command. The IP address has been specified in order to limit the packet information to one interface:

```
Router# debug bfd packet 172.16.10.5
BFD packet debugging is on
*Jan 26 14:47:37.645: Tx*IP: dst 172.16.10.1, plen 24. BFD: diag 2, St/D/P/F (1/0/0/0), mult 5, len 24, loc/rem discr 1 1, tx 1000000, rx 1000000 1000000, timer 1000 ms, #103
*Jan 26 14:47:37.645: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from FULL to DOWN, Neighbor Down: BFD node down
*Jan 26 14:47:50.685: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from LOADING to FULL, Loading Done
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (1/0/0/0), mult 4, len 24, loc/rem discr 2 1, tx 1000000, rx 1000000 1000000, timer 4000 ms, #50
*Jan 26 14:48:00.905: %OSPF-5-ADJCHG: Process 10, Nbr 172.16.10.12 on Ethernet1/4 from LOADING to FULL, Loading Done
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (1/0/0/0), mult 5, len 24, loc/rem discr 1 2, tx 1000000, rx 1000000 1000000, timer 1000 ms, #131
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (1/0/0/0), mult 4, len 24, loc/rem discr 2 1, tx 1000000, rx 1000000 1000000, timer 4000 ms, #41
*Jan 26 14:48:00.905: Rx IP: src 172.16.10.1, plen 24. BFD: diag 0, St/D/P/F (1/0/0/0), mult 5, len 24, loc/rem discr 1 2, tx 1000000, rx 1000000 1000000, timer 1000 ms, #153
```
The following example shows output from the `debug bfd event` command when an interface between two BFD neighbor routers fails and then comes back online:

```
Router# debug bfd event
22:53:49: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event DETECT TIMER EXPIRED, state UP -> FAILING
. .
22:56:37: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event RX IHY 0, state FAILING -> DOWN
22:56:37: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event RX IHY 0, state DOWN -> INIT
22:56:37: Session [172.16.10.1,172.16.10.2,Fa0/1,1], event RX IHY 1, state INIT -> UP
```

The table below describes the significant fields shown in the display.

**Table 1: debug bfd event Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bfd_neighbor - action:DESTROY</td>
<td>The BFD neighbor will tear down the BFD session.</td>
</tr>
<tr>
<td>Session [172.16.10.1, 172.16.10.2, Fa0/1,1]</td>
<td>IP addresses of the BFD neighbors holding this session that is carried over FastEthernet interface 0/1.</td>
</tr>
<tr>
<td>event DETECT TIMER EXPIRED</td>
<td>The BFD neighbor has not received BFD control packets within the negotiated interval and the detect timer has expired.</td>
</tr>
<tr>
<td>state UP -&gt; FAILING</td>
<td>The BFD event state is changing from Up to Failing.</td>
</tr>
<tr>
<td>Session [172.16.10.1, 172.16.10.2, Fa0/1,1], event RX IHY 0</td>
<td>The BFD session between the neighbors indicated by the IP addresses that is carried over FastEthernet interface 0/1 is changing state from Failing to Down. The I Hear You (IHY) bit value is shown as 0 to indicate that the remote system is tearing down the BFD session.</td>
</tr>
<tr>
<td>event RX IHY 0, state DOWN -&gt; INIT</td>
<td>The BFD session is still considered down, and the IHY bit value still is shown as 0, and the session state changes from DOWN to INIT to indicate that the BFD session is again initializing, as the interface comes back up.</td>
</tr>
<tr>
<td>event RX IHY 1, state INIT -&gt; UP</td>
<td>The BFD session has been reestablished, and the IHY bit value changes to 1 to indicate that the session is live. The BFD session state changes from INIT to UP.</td>
</tr>
</tbody>
</table>
The following example shows output from the `debug bfd packet` command when an interface between two BFD neighbor routers fails and then comes back online. The diagnostic code changes from 0 (No Diagnostic) to 1 (Control Detection Time Expired) because no BFD control packets could be sent (and therefore detected by the BFD peer) after the interface fails. When the interface comes back online, the diagnostic code changes back to 0 to signify that BFD packets can be sent and received by the BFD peers.

```
Router# debug bfd packet
23:03:25: Rx IP: src 172.16.10.2, plen 24. BFD: diag 0, H/D/P/F (0/0/0/0), mult 3, len 24, loc/rem discr 5 1, tx 1000000, rx 100007
23:03:25: Tx IP: dst 172.16.10.2, plen 24. BFD: diag 1, H/D/P/F (0/0/0/0), mult 5, len 24, loc/rem discr 1 5, tx 1000000, rx 1000008
23:03:25: Tx IP: dst 172.16.10.2, plen 24. BFD: diag 1, H/D/P/F (1/0/0/0), mult 5, len 24, loc/rem discr 1 5, tx 1000000, rx 1000009
```

The table below describes the significant fields shown in the display.

**Table 2: debug bfd packet Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx IP: src 172.16.10.2</td>
<td>The router has received this BFD packet from the BFD router with source address 172.16.10.2.</td>
</tr>
<tr>
<td>plen 24</td>
<td>Length of the BFD control packet, in bytes.</td>
</tr>
<tr>
<td>diag 0</td>
<td>A diagnostic code specifying the local system’s reason for the last transition of the session from Up to some other state. State values are as follows:</td>
</tr>
<tr>
<td></td>
<td>• 0--No Diagnostic</td>
</tr>
<tr>
<td></td>
<td>• 1--Control Detection Time Expired</td>
</tr>
<tr>
<td></td>
<td>• 2--Echo Function Failed</td>
</tr>
<tr>
<td></td>
<td>• 3--Neighbor Signaled Session Down</td>
</tr>
<tr>
<td></td>
<td>• 4--Forwarding Plane Reset</td>
</tr>
<tr>
<td></td>
<td>• 5--Path Down</td>
</tr>
<tr>
<td></td>
<td>• 6--Concentrated Path Down</td>
</tr>
<tr>
<td></td>
<td>• 7--Administratively Down</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| H/D/P/F (0/0/0/0) | H bit--Hear You bit. This bit is set to 0 if the transmitting system either is not receiving BFD packets from the remote system or is tearing down the BFD session. During normal operation the Hear You bit is set to 1.  
D bit--Demand Mode bit. If the Demand Mode bit is set, the transmitting system wants to operate in demand mode. BFD has two modes—asynchronous and demand. The Cisco implementation of BFD supports only asynchronous mode.  
P bit--Poll bit. If the Poll bit is set, the transmitting system is requesting verification of connectivity or of a parameter change.  
F bit--Final bit. If the Final bit is set, the transmitting system is responding to a received BFD control packet that had a Poll (P) bit set. |
| mult 3 | Detect time multiplier. The negotiated transmit interval, multiplied by the detect time multiplier, determines the detection time for the transmitting system in BFD asynchronous mode.  
The detect time multiplier is similar to the hello multiplier in IS-IS, which is used to determine the hold timer: (hellointerval) * (hellomultiplier) = hold timer. If a hello packet is not received within the hold-timer interval, a failure has occurred.  
Similarly, for BFD: (transmit interval) * (detect multiplier) = detect timer. If a BFD control packet is not received from the remote system within the detect-timer interval, a failure has occurred. |
| len 24 | The BFD packet length. |
| loc/rem discr 5 1 | The values for My Discriminator (local) and Your Discriminator (remote) BFD neighbors.  
- My Discriminator--Unique, nonzero discriminator value generated by the transmitting system, used to demultiplex multiple BFD sessions between the same pair of systems.  
- Your Discriminator--The discriminator received from the corresponding remote system. This field reflects the received value of My Discriminator, or is zero if that value is unknown. |
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tx 1000000</td>
<td>Desired minimum transmit interval.</td>
</tr>
<tr>
<td>rx 100007</td>
<td>Required minimum receive interval.</td>
</tr>
</tbody>
</table>
debug bgp ipv6 dampening

To display debugging messages for IPv6 Border Gateway Protocol (BGP) dampening, use the debug bgp ipv6 dampening command in privileged EXEC mode. To disable debugging messages for IPv6 BGP dampening, use the no form of this command.

```
deebug bgp ipv6 {unicast|multicast} dampening [prefix-list prefix-list-name]
no debug bgp ipv6 {unicast|multicast} dampening [prefix-list prefix-list-name]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast</td>
<td>Specifies IPv6 unicast address prefixes.</td>
</tr>
<tr>
<td>multicast</td>
<td>Specifies IPv6 multicast address prefixes.</td>
</tr>
<tr>
<td>prefix-list</td>
<td>(Optional) Name of an IPv6 prefix list.</td>
</tr>
<tr>
<td>prefix-list-name</td>
<td></td>
</tr>
</tbody>
</table>

### Command Default

Debugging for IPv6 BGP dampening packets is not enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(21)ST</td>
<td>This command was integrated into Cisco IOS Release 12.0(21)ST.</td>
</tr>
<tr>
<td>12.0(22)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(22)S.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>The prefix-list keyword was added.</td>
</tr>
<tr>
<td>12.0(24)S</td>
<td>The prefix-list keyword was added.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
<tr>
<td>12.2(25)SG</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)SG.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.1</td>
<td>This command was introduced on Cisco ASR 1000 Series Routers.</td>
</tr>
</tbody>
</table>
**Usage Guidelines**

The `debug bgp ipv6 dampening` command is similar to the `debug ip bgp dampening` command, except that it is IPv6-specific.

Use the `prefix-list` keyword and an argument to filter BGP IPv6 dampening debug information through an IPv6 prefix list.

By default, the network server sends the output from debug commands and system error messages to the console. To redirect debugging output, use the `logging` command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

**Examples**

The following is sample output from the `debug bgp ipv6 dampening` command:

```
Router# debug bgp ipv6 dampening
00:13:28:BGP(1):charge penalty for 2000:0:0:1::/64 path 2 1 with halflife-time 15
reuse/suppress 750/2000
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:13:28:BGP(1):charge penalty for 2000:0:0:1:1::/80 path 2 1 with halflife-time 15
reuse/suppress 750/2000
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:13:28:BGP(1):charge penalty for 2000:0:0:1::/64 path 2 1 with halflife-time 15
reuse/suppress 750/2000
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:13:28:BGP(1):charge penalty for 2000:0:0:1:1::/80 path 2 1 with halflife-time 15
reuse/suppress 750/2000
00:13:28:BGP(1):flapped 1 times since 00:00:00. New penalty is 1000
00:13:28:BGP(1):charge penalty for 2000:0:0:1::/64 path 2 1 with halflife-time 15
reuse/suppress 750/2000
00:13:28:BGP(1):flapped 2 times since 00:02:35. New penalty is 1892
00:18:28:BGP(1):suppress 2000:0:0:1:1::/80 path 2 1 for 00:27:30 (penalty 2671)
00:18:28:halflife-time 15, reuse/suppress 750/2000
00:18:28:BGP(1):suppress 2000:0:0:1:1::/64 path 2 1 for 00:27:20 (penalty 2664)
00:18:28:halflife-time 15, reuse/suppress 750/2000
```

The following example shows output for the `debug bgp ipv6 dampening` command filtered through the prefix list named marketing:

```
Router# debug bgp ipv6 dampening prefix-list marketing
00:16:08:BGP(1):charge penalty for 2001:0DB8::/64 path 30 with halflife-time 15
reuse/suppress 750/2000
00:16:08:BGP(1):flapped 1 times since 00:00:00. New penalty is 10
```

The table below describes the fields shown in the display.

**Table 3: debug bgp ipv6 dampening Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>penalty</td>
<td>Numerical value of 1000 assigned to a route by a router configured for route dampening in another autonomous system each time a route flaps. Penalties are cumulative. The penalty for the route is stored in the BGP routing table until the penalty exceeds the suppress limit. If the penalty exceeds the suppress limit, the route state changes from history to damp.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>flapped</td>
<td>Number of times a route is available, then unavailable, or vice versa.</td>
</tr>
<tr>
<td>halflife-time</td>
<td>Amount of time (in minutes) by which the penalty is decreased after the route is assigned a penalty. The halflife-time value is half of the half-life period (which is 15 minutes by default). Penalty reduction happens every 5 seconds.</td>
</tr>
<tr>
<td>reuse</td>
<td>The limit by which a route is unsuppressed. If the penalty for a flapping route decreases and falls below this reuse limit, the route is unsuppressed. That is, the route is added back to the BGP table and once again used for forwarding. The default reuse limit is 750. Routes are unsuppressed at 10-second increments. Every 10 seconds, the router determines which routes are now unsuppressed and advertises them to the world.</td>
</tr>
<tr>
<td>suppress</td>
<td>Limit by which a route is suppressed. If the penalty exceeds this limit, the route is suppressed. The default value is 2000.</td>
</tr>
<tr>
<td>maximum suppress limit (not shown in sample output)</td>
<td>Maximum amount of time (in minutes) a route is suppressed. The default value is four times the half-life period.</td>
</tr>
<tr>
<td>damp state (not shown in sample output)</td>
<td>State in which the route has flapped so often that the router will not advertise this route to BGP neighbors.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp ipv6 updates</td>
<td>Displays debugging messages for IPv6 BGP update packets.</td>
</tr>
</tbody>
</table>
debug bgp ipv6 updates

To display debugging messages for IPv6 Border Gateway Protocol (BGP) update packets, use the debug bgp ipv6 updates command in privileged EXEC mode. To disable debugging messages for IPv6 BGP update packets, use the no form of this command.

```plaintext
debug bgp ipv6 {unicast|multicast} updates [ipv6-address] [prefix-list prefix-list-name] [in|out]
no debug bgp ipv6 {unicast|multicast} updates [ipv6-address] [prefix-list prefix-list-name] [in|out]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicast</td>
<td>Specifies IPv6 unicast address prefixes.</td>
</tr>
<tr>
<td>multicast</td>
<td>Specifies IPv6 multicast address prefixes.</td>
</tr>
<tr>
<td>ipv6-address</td>
<td>(Optional) The IPv6 address of a BGP neighbor. This argument must be in the form documented in RFC 2373 where the address is specified in hexadecimal using 16-bit values between colons.</td>
</tr>
<tr>
<td>prefix-list prefix-list-name</td>
<td>(Optional) Name of an IPv6 prefix list.</td>
</tr>
<tr>
<td>in</td>
<td>(Optional) Indicates inbound updates.</td>
</tr>
<tr>
<td>out</td>
<td>(Optional) Indicates outbound updates.</td>
</tr>
</tbody>
</table>

### Command Default

Debugging for IPv6 BGP update packets is not enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.0(21)ST</td>
<td>This command was integrated into Cisco IOS Release 12.0(21)ST.</td>
</tr>
<tr>
<td>12.0(22)S</td>
<td>This command was integrated into Cisco IOS Release 12.0(22)S.</td>
</tr>
<tr>
<td>12.2(13)T</td>
<td>The prefix-list keyword was added.</td>
</tr>
<tr>
<td>12.0(24)S</td>
<td>The prefix-list keyword was added.</td>
</tr>
<tr>
<td>12.2(14)S</td>
<td>This command was integrated into Cisco IOS Release 12.2(14)S.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB.</td>
</tr>
</tbody>
</table>
Usage Guidelines

The `debug bgp ipv6 updates` command is similar to the `debug ip bgp updates` command, except that it is IPv6-specific.

Use the `prefix-list` keyword to filter BGP IPv6 updates debugging information through an IPv6 prefix list.

By default, the network server sends the output from `debug` commands and system error messages to the console. To redirect debugging output, use the `logging` command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on `debug` commands and redirecting debugging output, refer to the Release 12.2 Cisco IOS Debug Command Reference.

Examples

The following is sample output from the `debug bgp ipv6 updates` command:

```
Router# debug bgp ipv6 updates
14:04:17:BGP(1):2000:0:0:2::2 computing updates, afi 1, neighbor version 0, table version 1, starting at ::
14:04:17:BGP(1):2000:0:0:2::2 update run completed, afi 1, ran for 0ms, neighbor version 0, start version 1, throttled to 1
14:04:19:BGP(1):sourced route for 2000:0:0:2::1/64 path #0 changed (weight 32768)
14:04:19:BGP(1):2000:0:0:2::1/64 route sourced locally
14:04:19:BGP(1):2000:0:0:3::2/64 route sourced locally
14:04:19:BGP(1):2000:0:0:4::2/64 route sourced locally
14:04:22:BGP(1):2000:0:0:2::2 computing updates, afi 1, neighbor version 1, table version 6, starting at ::
14:04:22:BGP(1):2000:0:0:2::2 send UPDATE (format) 2000:0:0:2::1/64, next 2000:0:0:2::1, metric 0, path
14:04:22:BGP(1):2000:0:0:2::2 send UPDATE (format) 2000:0:0:2:1::/80, next 2000:0:0:2:1, metric 0, path
14:04:22:BGP(1):2000:0:0:2::2 send UPDATE (prepend, chgflags:0x208) 2000:0:0:3::2/64, next 2000:0:0:2::1, metric 0, path
14:04:22:BGP(1):2000:0:0:2::2 send UPDATE (prepend, chgflags:0x208) 2000:0:0:4::2/64, next 2000:0:0:2::1, metric 0, path
```

The following is sample output from the `debug bgp ipv6 updates prefix-list sales` command filtered through the prefix list named sales:

```
Router# debug bgp ipv6 updates prefix-list sales
00:18:26:BGP(1):2000:8493:1::2 send UPDATE (prepend, chgflags:0x208) 7878:7878::/64, next 2000:0:0:4::2/64, next 2000:0:0:2::1, metric 0, path
```

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(25)SG</td>
<td>This command was integrated into Cisco IOS Release 12.2(25)SG.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>Cisco IOS XE Release 2.1</td>
<td>This command was introduced on Cisco ASR 1000 Series Routers.</td>
</tr>
</tbody>
</table>
### Table 4: `debug bgp ipv6 updates` Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP(1):</td>
<td>BGP debugging for address family index (afi) 1.</td>
</tr>
<tr>
<td>afi</td>
<td>Address family index.</td>
</tr>
<tr>
<td>neighbor version</td>
<td>Version of the BGP table on the neighbor from which the update was received.</td>
</tr>
<tr>
<td>table version</td>
<td>Version of the BGP table on the router from which you entered the <code>debug bgp ipv6 updates</code> command.</td>
</tr>
<tr>
<td>starting at</td>
<td>Starting at the network layer reachability information (NLRI). BGP sends routing update messages containing NLRI to describe a route and how to get there. In this context, an NLRI is a prefix. A BGP update message carries one or more NLRI prefixes and the attributes of a route for the NLRI prefixes; the route attributes include a BGP next hop gateway address, community values, and other information.</td>
</tr>
<tr>
<td>route sourced locally</td>
<td>Indicates that a route is sourced locally and that updates are not sent for the route.</td>
</tr>
<tr>
<td>send UPDATE (format)</td>
<td>Indicates that an update message for a reachable network should be formatted. Addresses include prefix and next hop.</td>
</tr>
<tr>
<td>send UPDATE (prepend, chgflags:0x208)</td>
<td>Indicates that an update message about a path to a BGP peer should be written.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug bgp ipv6 dampening</code></td>
<td>Displays debugging messages for IPv6 BGP dampening packets.</td>
</tr>
</tbody>
</table>
**debug bgp l2vpn vpls updates**

To enable debugging of the L2VPN VPLS address family updates from the BGP table, use the `debug bgp l2vpn vpls updates` command in privileged EXEC mode. To disable the display of the messages, use the `no` form of this command.

```
Device> enable
Device# debug bgp l2vpn vpls updates
BGP updates debugging is on for address family: L2VPN VplS
```

### Syntax Description

<table>
<thead>
<tr>
<th>Access List</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>access-list</code></td>
<td>(Optional) Number of an access list used to filter debugging messages.</td>
</tr>
<tr>
<td><code>expanded-access-list</code></td>
<td>(Optional) Number of an expanded access list used to filter debugging messages.</td>
</tr>
<tr>
<td><code>bgp-neighbor-address</code></td>
<td>(Optional) BGP neighbor address in the format A.B.C.D.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>(Optional) Specifies debugging messages for BGP update events.</td>
</tr>
<tr>
<td><code>in</code></td>
<td>Specifies debugging messages for inbound BGP update information.</td>
</tr>
<tr>
<td><code>out</code></td>
<td>Specifies debugging messages for outbound BGP update information.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 3.8S</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

### Examples

The following shows how to enable the `debug bgp l2vpn vpls updates` command:
## Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ip bgp updates</code></td>
<td>Displays information about the processing of BGP updates.</td>
</tr>
<tr>
<td><code>show bgp l2vpn vpls</code></td>
<td>Displays L2VPN VPLS address family information from the BGP table.</td>
</tr>
</tbody>
</table>
debug bgp nsap

To enable the display of Border Gateway Protocol (BGP) debugging information specific to the network service access point (NSAP) address family, use the **debug bgp nsap** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug bgp nsap**

**no debug bgp nsap**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging of BGP NSAP address-family code is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>Cisco IOS XE 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The **debug bgp nsap** command is similar to the **debug ip bgp** command, except that it is specific to the NSAP address family.

**Note**

By default, the network server sends the output from **debug** commands and system error messages to the console. To redirect debug output, use the **logging** command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

**Examples**

The following example shows output for the **debug bgp nsap** command. The BGP(4) identifies that BGP version 4 is operational.

```
Router# debug bgp nsap
00:46:46: BGP(4): removing CLNS route to 49.0101
00:46:46: BGP(4): removing CLNS route to 49.0303
00:46:46: BGP(4): 10.1.2.1 removing CLNS route 49.0101.1111.1111.1111.1111.00 to eBGP-neighbor
00:46:46: BGP(4): 10.1.2.1 removing CLNS route 49.0101.1111.1111.1111.1111.00 to eBGP-neighbor
```
00:46:59: BGP(4): Applying map to find origin for prefix 49.0202.2222
00:46:59: BGP(4): Applying map to find origin for prefix 49.0202.3333

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp nsap dampening</td>
<td>Displays debug messages for BGP NSAP prefix dampening events.</td>
</tr>
<tr>
<td>debug bgp nsap updates</td>
<td>Displays debug messages for BGP NSAP prefix update packets.</td>
</tr>
</tbody>
</table>
debug bgp nsap dampening

To display debug messages for Border Gateway Protocol (BGP) network service access point (NSAP) prefix address dampening, use the `debug bgp nsap dampening` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug bgp nsap dampening [filter-list access-list-number]`

`no debug bgp nsap dampening [filter-list access-list-number]`

**Syntax Description**

| filter-list access-list-number | (Optional) Displays debug messages for BGP NSAP dampening events that match the access list. The acceptable access list number range is from 1 to 199. |

**Command Default**

Debugging for BGP NSAP dampening events is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>Cisco IOS XE 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bgp nsap dampening` command is similar to the `debug ip bgp dampening` command, except that it is specific to the NSAP address family.

**Note**

By default, the network server sends the output from `debug` commands and system error messages to the console. To redirect debug output, use the `logging` command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

**Examples**

The following example shows output for the `debug bgp nsap dampening` command:

```
Router# debug bgp nsap dampening
```
Only one line of output is displayed unless the `bgpdampening` command is configured with a route map in NSAP address family configuration mode. The following example shows output for the `debug bgp nsap dampening` command when a route map is configured:

```
20:07:19: BGP(4): charge penalty for 49.0404 path 65202 65404 with halflife-time 15
reuse/suppress 750/2000
20:07:19: BGP(4): flapped 1 times since 00:00:00. New penalty is 1000
20:08:59: BGP(4): charge penalty for 49.0404 path 65202 65404 with halflife-time 15
reuse/suppress 750/2000
20:08:59: BGP(4): flapped 2 times since 00:01:39. New penalty is 1928
20:10:04: BGP(4): charge penalty for 49.0404 path 65202 65404 with halflife-time 15
reuse/suppress 750/2000
20:10:04: BGP(4): flapped 3 times since 00:02:44. New penalty is 2839
20:10:48: BGP(4): suppress 49.0404 path 65202 65404 for 00:28:10 (penalty 2752)
```

The table below describes the significant fields shown in the display.

**Table 5: debug bgp nsap dampening Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>penalty</td>
<td>Numerical value of 1000 assigned to a route by a router configured for route dampening in another autonomous system each time a route flaps. Penalties are cumulative. The penalty for the route is stored in the BGP routing table until the penalty exceeds the suppress limit. If the penalty exceeds the suppress limit, the route state changes from history to damp.</td>
</tr>
<tr>
<td>halflife-time</td>
<td>Amount by which the penalty is decreased after the route is assigned a penalty. The half-life-time value is half of the half-life period (which is 15 minutes by default). Penalty reduction occurs every 5 seconds.</td>
</tr>
<tr>
<td>flapped</td>
<td>Number of times a route is available, then unavailable, or vice versa.</td>
</tr>
<tr>
<td>reuse</td>
<td>The limit by which a route is unsuppressed. If the penalty for a flapping route decreases and falls below this reuse limit, the route is unsuppressed. That is, the route is added back to the BGP table and once again used for forwarding. The default reuse limit is 750. Unsuppressing of routes occurs at 10-second increments. Every 10 seconds, the router learns which routes are now unsuppressed and advertises them throughout the network.</td>
</tr>
<tr>
<td>suppress</td>
<td>Limit by which a route is suppressed. If the penalty exceeds this limit, the route is suppressed. The default value is 2000.</td>
</tr>
<tr>
<td>maximum suppress limit (not shown in sample output)</td>
<td>Maximum amount of time a route is suppressed. The default value is four times the half-life period.</td>
</tr>
</tbody>
</table>
### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>damp state (not shown in sample output)</td>
<td>State in which the route has flapped so often that the router will not advertise this route to BGP neighbors.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp nsap</td>
<td>Displays debug messages for BGP NSAP packets.</td>
</tr>
<tr>
<td>debug bgp nsap updates</td>
<td>Displays debug messages for BGP NSAP update events.</td>
</tr>
</tbody>
</table>
debug bgp nsap updates

To display debug messages for Border Gateway Protocol (BGP) network service access point (NSAP) prefix address update packets, use the `debug bgp nsap updates` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bgp nsap updates [ ip-address ] [ in| out ] [ filter-set clns-filter-set-name ]
no debug bgp nsap updates [ ip-address ] [ in| out ] [ filter-set clns-filter-set-name ]
```

**Syntax Description**

- `ip-address` (Optional) The IP address of a BGP neighbor.
- `in` (Optional) Indicates inbound updates.
- `out` (Optional) Indicates outbound updates.

**Command Default**

Debugging for BGP NSAP prefix update packets is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRB.</td>
</tr>
<tr>
<td>Cisco IOS XE 2.6</td>
<td>This command was integrated into Cisco IOS XE Release 2.6.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The `debug bgp nsap updates` command is similar to the `debug ip bgp updates` command, except that it is specific to the NSAP address family.

Use the `ip-address` argument to display the BGP update debug messages for a specific BGP neighbor. Use the `clns-filter-set-name` argument to display the BGP update debug messages for a specific NSAP prefix.
By default, the network server sends the output from `debug` commands and system error messages to the console. To redirect debug output, use the `logging` command options within global configuration mode. Destinations include the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server.

### Examples

The following example shows output for the `debug bgp nsap updates` command:

```
Router# debug bgp nsap updates
02:13:45: BGP(4): 10.0.3.4 send UPDATE (format) 49.0101, next 49.0303.3333.3333.3333.3333.00, metric 0, path 65202 65101
02:13:45: BGP(4): 10.0.3.4 send UPDATE (format) 49.0202, next 49.0303.3333.3333.3333.3333.00, metric 0, path 65202
02:13:45: BGP(4): 10.0.3.4 send UPDATE (format) 49.0303, next 49.0303.3333.3333.3333.3333.00, metric 0, path
02:13:45: BGP(4): 10.0.2.2 send UPDATE (format) 49.0404, next 49.0303.3333.3333.3333.3333.3333.00, metric 0, path 65404
```

The table below describes the significant fields shown in the display.

### Table 6: debug bgp nsap updates Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGP(4):</td>
<td>BGP debug for address family index (afi) 4.</td>
</tr>
<tr>
<td>route sourced locally (not shown in display)</td>
<td>Indicates that a route is sourced locally and that updates are not sent for the route.</td>
</tr>
<tr>
<td>send UPDATE (format)</td>
<td>Indicates that an update message for a reachable network should be formatted. Addresses include NSAP prefix and next hop.</td>
</tr>
<tr>
<td>rcv UPDATE (not shown in display)</td>
<td>Indicates that an update message about a path to a BGP peer has been received. Addresses include NSAP prefix.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bgp nsap</td>
<td>Displays debug messages for BGP NSAP packets.</td>
</tr>
<tr>
<td>debug bgp nsap dampening</td>
<td>Displays debug messages for BGP NSAP prefix dampening events.</td>
</tr>
</tbody>
</table>
debug bgp vpnv6 unicast

To display Border Gateway Protocol (BGP) virtual private network (VPN) debugging output, use the `debug bgp vpnv6 unicast` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bgp vpnv6 unicast
no debug bgp vpnv6
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SB.</td>
</tr>
<tr>
<td>12.2(33)SXI</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXI.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug bgp vpnv6 unicast` command to help troubleshoot the BGP VPN.

**Note**

By default, the network server sends the output from debug commands and system error messages to the console. To redirect debugging output, use the logging command options within global configuration mode. Destinations are the console, virtual terminals, internal buffer, and UNIX hosts running a syslog server. For complete information on debug commands and redirecting debugging output, refer to the Cisco IOS Debug Command Reference, Release 12.4.

**Examples**

The following example enables BGP debugging output for IPv6 VPN instances:

```
Router# debug bgp vpnv6 unicast
```
debug bri-interface

To display debugging information on ISDN BRI routing activity, use the `debug bri-interface` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bri-interface
no debug bri-interface
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

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

⚠️ **Caution**

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

**Examples**

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

```
Router# debug bri-interface
BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: write_sid: wrote 15 for subunit 0, slot 1.
BRI: write_sid: wrote 17 for subunit 0, slot 1.
BRI: write_sid: wrote 6 for subunit 0, slot 1.
BRI: write_sid: wrote 11 for subunit 0, slot 1.
BRI: write_sid: wrote 13 for subunit 0, slot 1.
BRI: write_sid: wrote 29 for subunit 0, slot 1.
BRI: write_sid: wrote 1B for subunit 0, slot 1.
BRI: Starting Power Up timer for unit = 0.
BRI: write_sid: wrote 3 for subunit 0, slot 1.
BRI: Starting T3 timer after expiry of PUP timeout for unit = 0, current state is F4.
BRI: write_sid: wrote FF for subunit 0, slot 1.
BRI: Activation for unit = 0, current state is F7.
BRI: enable channel B1
BRI: write_sid: wrote 14 for subunit 0, slot 1.
%LINK-3-UPDOWN: Interface BRIO: B-Channel 1, changed state to up
%LINK-5-CHANGED: Interface BRIO: B-Channel 1, changed state to up.!!!
BRI: disable channel B1
BRI: write_sid: wrote 15 for subunit 0, slot 1.
%LINK-3-UPDOWN: Interface BRIO: B-Channel 1, changed state to down
%LINK-5-CHANGED: Interface BRIO: B-Channel 1, changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface BRIO: B-Channel 1, changed state to down
```
The following line indicates that an internal command was written to the interface controller. The subunit identifies the first interface in the slot.

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

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

BRI: Starting Power Up timer for unit = 0.

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

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

The following line indicates that the channel was disabled:

BRI: disable channel B1

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

## Related Commands

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

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

`debug bsc event [ number ]`

`no debug bsc event [ number ]`

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>number</th>
<th>(Optional) Group number.</th>
</tr>
</thead>
</table>

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command traces all interfaces configured with a `bscprotocol-group number` command.

**Examples**

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

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bsc packet</td>
<td>Displays all frames traveling through the Bisync feature.</td>
</tr>
<tr>
<td>debug bstun events</td>
<td>Displays BSTUN connection events and status.</td>
</tr>
</tbody>
</table>
debug bsc packet

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

default bsc packet [group number] [buffer-size bytes]
no default bsc packet [group number] [buffer-size bytes]

Syntax Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group number</td>
<td>(Optional) Group number.</td>
</tr>
<tr>
<td>buffer-size</td>
<td>(Optional) Number of bytes displayed per packet</td>
</tr>
<tr>
<td></td>
<td>(defaults to 20).</td>
</tr>
</tbody>
</table>

Command Default

The default number of bytes displayed is 20.

Command Modes

Privileged EXEC

Usage Guidelines

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

Examples

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

Router# debug bsc packet
0:23:33: BSC: Serial2 :NDI-rx : 27 bytes 401A400227F5C31140C11D60C0C5D3D3D51D4013
0:23:33: BSC: Serial2 :SDI-tx : 12 bytes 00323237FF3232606040402D
0:23:33: BSC: Serial2 :SDI-rx : 2 bytes 1070
0:23:33: BSC: Serial2 :SDI-tx : 27 bytes 401A400227F5C31140C11D60C8C5D3D3D51D4013
0:23:33: BSC: Serial2 :SDI-rx : 2 bytes 1061
0:23:33: BSC: Serial2 :SDI-tx : 5 bytes 00323237FF

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bsc event</td>
<td>Displays all events occurring in the Bisync feature.</td>
</tr>
<tr>
<td>debug bstun events</td>
<td>Displays BSTUN connection events and status.</td>
</tr>
</tbody>
</table>
**debug bstun events**

To display BSTUN connection events and status, use the `debug bstun events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bstun events [number]
no debug bstun events [number]
```

**Syntax Description**

- **number** (Optional) Group number.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

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

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

**Note**

Also refer to the `debug bscp packet` and `debug bscevent` commands. Currently, these two commands support the only protocol working through the BSTUN tunnel. Sometimes frames do not go through the tunnel because they have been discarded at the Bisync protocol level.

**Examples**

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

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

**Note**

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

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

```
Router# debug bstun events
BSTUN: Change state for peer (C1[1])_172.16.12.22/1976 (closed->opening)  
BSTUN: Change state for peer (C1[1])_172.16.12.22/1976 (opening->open wait)  
%BSTUN-6-OPENING: CONN: opening peer (C1[1])_172.16.12.22/1976, 3  
BSTUN: tcpd sender in wrong state, dropping packet  
BSTUN: tcpd sender in wrong state, dropping packet  
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bsc event</td>
<td>Displays all events occurring in the Bisync feature.</td>
</tr>
<tr>
<td>debug bsc packet</td>
<td>Displays all frames traveling through the Bisync feature.</td>
</tr>
<tr>
<td>debug bstun packet</td>
<td>Displays packet information on packets traveling through the BSTUN links.</td>
</tr>
</tbody>
</table>
debug bstun packet

To display packet information on packets traveling through the BSTUN links, use the `debug bstun packet` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

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

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group number</td>
<td>(Optional) BSTUN group number.</td>
</tr>
<tr>
<td>buffer-size bytes</td>
<td>(Optional) Number of bytes displayed per packet (defaults to 20).</td>
</tr>
</tbody>
</table>

**Command Default**

The default number of bytes displayed is 20.

**Command Modes**

Privileged EXEC

**Examples**

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

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug bstun events</td>
<td>Displays BSTUN connection events and status.</td>
</tr>
</tbody>
</table>
debug bundle errors

To enable the display of information on bundle errors, use the `debug bundle errors` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug bundle errors
no debug bundle errors
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bump</td>
<td>Configures the bumping rules for a VC class that can be assigned to a VC bundle.</td>
</tr>
<tr>
<td>bundle</td>
<td>Creates a bundle or modifies an existing bundle to enter bundle configuration mode.</td>
</tr>
<tr>
<td>debug bundle events</td>
<td>Enables display of bundle events when use occurs.</td>
</tr>
</tbody>
</table>
**debug bundle events**

To enable display of bundle events when use occurs, use the `debug bundle events` command in privileged EXEC mode. To disable the display, use the `no` form of this command.

```plaintext
debug bundle events
no debug bundle events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug bstun packet</code></td>
<td>Enables the display of information on bundle errors.</td>
</tr>
</tbody>
</table>
debug call-home diagnostic-signature

To enable the debugging of call-home diagnostic signature flags on a device, use the `debug call-home diagnostic-signature` command in privileged EXEC mode. To disable debugging, use the `no` form of this command.

```
debug call-home diagnostic-signature {action | all | api | cli | download | event-registration | parsing}
no debug call-home diagnostic-signature {action | all | api | cli | download | event-registration | parsing}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>action</code></td>
<td>Displays debugging information associated with the execution of any call-home diagnostic signature action defined in the diagnostic signature file.</td>
</tr>
<tr>
<td><code>all</code></td>
<td>Displays debugging information about all flags associated with the call-home diagnostic signature.</td>
</tr>
<tr>
<td><code>api</code></td>
<td>Displays debugging information associated with call-home diagnostic signature internal operations or function calls.</td>
</tr>
<tr>
<td><code>cli</code></td>
<td>Displays debugging information associated with the call-home diagnostic signature to run the CLI commands as part of the diagnostic signature actions.</td>
</tr>
<tr>
<td><code>download</code></td>
<td>Displays debugging information associated with the downloading of call-home diagnostic signature files from the HTTP/HTTPS servers.</td>
</tr>
<tr>
<td><code>event-registration</code></td>
<td>Displays debugging information associated with the registration of call-home diagnostic signature events.</td>
</tr>
<tr>
<td><code>parsing</code></td>
<td>Displays debugging information associated with the parsing of call-home diagnostic-signature files.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.3(2)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug call-home diagnostic-signature action` command:

```
Device# debug call-home diagnostic-signature action
```
The following is sample output from the **debug call-home diagnostic-signature api** command:

```
Device# debug call-home diagnostic-signature api
```

```
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_all_lock[101], lock callhome and ds mutex
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_ds_lock[42], lock call home ds semaphore
Jan 29 10:41:24.902 CST: DS-API-TRACE: call_home_all_unlock[109], unlock callhome and ds mutex
```

The following is sample output from the **debug call-home diagnostic-signature cli** command:

```
Device# debug call-home diagnostic-signature cli
```

```
Jan 29 10:44:31.402 CST: DS-CLI-TRACE: call_home_ds_eem_cmd_run[981], the first 100 chars of output cmd: show version
Cisco IOS Software, C1861 Software (C1861-ADVENTERPRISEK9-M), Experimental Version 15.
Jan 29 10:44:31.442 CST: DS-CLI-TRACE: call_home_ds_eem_cmd_run[981], the first 100 chars of output cmd: show logging
Syslog logging: enabled (0 messages dropped, 3 messages rate-limited, 0 flushes, 0 over
```

The following is sample output from the **debug call-home diagnostic-signature download** command:

```
Device# debug call-home diagnostic-signature download
```

```
Jan 29 10:40:11.050 CST: DS-DNLD-TRACE: call_home_ds_update_thread_create[239], Creating a download process
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: call_home_ds_update_thread_create[239], Creating a download process
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[370], Collecting XML content prolog values
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[489], System Name:CH1861-1
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[494], Unable to get SNMP contact string
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[550], Collecting XML content epilog values
Jan 29 10:40:11.054 CST: DS-DNLD-TRACE: ds_collect_content_prolog_values[621], Collecting XML DS request values
```

The following is sample output from the **debug call-home diagnostic-signature event-registration** command:

```
Device# debug call-home diagnostic-signature event-registration
```

```
Jan 29 10:40:16.734 CST: DS-REG-TRACE: call_home_ds_event_reg[515], ds "6030", index 0, event number 1
```

```
Jan 29 10:40:16.734 CST: DS-REG-TRACE: call_home_ds_esid_reg[323], ds "6030", index 0, T = 41, S = 3FCH
```

```
```

```
```

```
The following is sample output from the `debug call-home diagnostic-signature parsing` command:

```
Device# debug call-home diagnostic-signature parsing
Jan 29 10:40:16.734 CST: DS-REG-TRACE: call_home_ds_content_reparse[3108], reparse ds "6030" content
Jan 29 10:40:16.734 CST: DS-PARSE-TRACE: ds_content_var_reparse[2915], reparse ds var in 6030
Jan 29 10:40:16.734 CST: DS-PARSE-TRACE: ds_sys_var_local_queue_init[2860], copy sys var ds_signature_id
Jan 29 10:40:16.734 CST: DS-PARSE-TRACE: ds_sys_var_local_queue_init[2860], copy sys var ds_hostname
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>call-home diagnostic-signature</strong></td>
<td>Downloads, installs, and uninstalls diagnostic signature files on a device.</td>
</tr>
<tr>
<td><strong>show call-home diagnostic-signature statistics</strong></td>
<td>Displays statistics and attributes of a diagnostic signature file on a device.</td>
</tr>
</tbody>
</table>
**debug call-mgmt**

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

```plaintext
debug call-mgmt
no debug call-mgmt
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output after the `debug call-mgmt` command has been enabled:

```
Router# debug call-mgmt
Call Management debugging is on
Router# Dec 26 13:57:27.710: msg_to_calls_mgmt: msg type CPM_NEW_CALL_CSM_CONNECT received
Dec 26 13:57:27.714: In actv_c_proc_message, 
   access type CPM_INSERT_NEW_CALL, 
   call type CPM_ISDN_ANALOG: 
   CSM completed connecting a new modem call

Dec 26 13:57:45.906: msg_to_calls_mgmt: msg type CPM_NEW_CALL_ISDN_CONNECT received
Dec 26 13:57:45.906: In actv_c_proc_message, 
   access type CPM_INSERT_NEW_CALL, 
   call type CPM_ISDN_ANALOG: 
   Added a new ISDN analog call to the active-calls list
   CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
   Mdm-Slot#1, Mdm-Port#5, TTY#219

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

Router# clear int as1/03
Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received
Dec 26 13:58:26.538: In activ_c_proc_message,
   access type CPM_REMOVE_DISC_CALL,
   call type CPM_ISDN_ANALOG:
   Removed a disconnected ISDN analog call
   CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1
   Mdm-Slot#1, Mdm-Port#3, TTY#219

The table below describes the significant fields shown in the display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPM_NEW_CALL_CSM_CONNECT</td>
<td>Indicates the arrival of a new call.</td>
</tr>
<tr>
<td>access type CPM_INSERT_NEW_CALL, call type CPM_ISDN_ANALOG:</td>
<td>Indicates that the new call is an analog ISDN B channel call (either a voice call or a call over an analog modem), rather than a digital (V.110) call.</td>
</tr>
<tr>
<td>CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Mdm-Slot#1, Mdm-Port#3, TTY#219</td>
<td>Indicates that the call is connected via the B channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).</td>
</tr>
<tr>
<td>Dec 26 13:58:25.682: Call mgmt per minute statistics: active list length: 1 history list length: 3</td>
<td>Displays periodic statistics that give the allocation state of each DSX1 interface present in the system, as well as the number of current (active) and recent (history) calls.</td>
</tr>
<tr>
<td>Dec 26 13:58:26.538: msg_to_calls_mgmt: msg type CPM_VOICE_CALL_REJ_NO_MOD_AVAIL received</td>
<td>Indicates that the analog ISDN B channel call has been disassociated from a modem.</td>
</tr>
</tbody>
</table>
Indicate that the analog ISDN B channel call has been disconnected.

Indicates that the call has been disconnected via the B channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access type CPM_REMOVE_DISC_CALL, call type CPM_ISDN_ANALOG: Removed a disconnected ISDN analog call</td>
<td>Indicates that the analog ISDN B channel call has been disconnected.</td>
</tr>
<tr>
<td>CC-Slot#7, DSX1-Ctrlr#17, DS0-Timeslot#1 Dec 26 13:58:26.538: Mdm-Slot#1, Mdm-Port#3, TTY#219</td>
<td>Indicates that the call has been disconnected via the B channel on Serial7/17:1 to the asynchronous modem resource 1/03 (interface async1/03, also known as line tty219).</td>
</tr>
</tbody>
</table>
debug call fallback detail

To display details of the call fallback, use the debugcallfallbackdetail command in privileged EXEC mode. To disable debugging output, use the no form of this command.

dump call fallback detail
no dump call fallback detail

Syntax Description This command has no arguments or keywords.

Command Default Disabled

Command Modes Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T3</td>
<td>This command was implemented on the Cisco 7500 series routers routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines Every time a call request is received, the debugcallfallbackdetail command displays in the command-line interface (CLI) cache lookup and call acceptance/rejection information. Use this command to monitor call requests as they enter the call fallback subsystem.

If you have a large amount of calls in your router, enabling this command can cause delays in your routing functions as the debug statistics are constantly compiled and sent to your terminal. Also, debug messages on your terminal may make for difficult CLI configuring.

Examples The following example depicts a call coming in to 10.1.1.4 with codec g729r8. Because there is no cache entry for this destination, a probe is sent and values are inserted into the cache. A lookup is performed again, entry is found, and a fallback decision is made to admit the call.

Router# debug call fallback detail
When cache is empty:
dump call fallback detail:
2d19h:fb_lookup_cache:10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:No entry found.
2d19h:fb_check:No entry exists, enqueueing probe info... 10.1.1.4, codec:g729r8
2d19h:fb_main:Got FB_APP_INQ event
The following example depicts a call coming into 10.1.1.4 with codec g729r8. A lookup is performed, entry is found, and a fallback decision is made to admit the call.

Router# debug call fallback detail
When cache is full:
2d19h:fb_lookup_cache:10.1.1.4, codec:g729r8
2d19h:fb_lookup_cache:Found entry.
2d19h:fb_check:Returned FB_CHECK_TRUE, 10.1.1.4, codec:g729r8
2d19h:fb_main:calling callback function with:TRUE
**debug call fallback probe**

To display details of the call fallback probes, use the `debug call fallback probe` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call fallback probe
no debug call fallback probe
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>The <code>call fallback</code> and <code>call fallback reject-cause-code</code> commands were introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was implemented on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(4)T3</td>
<td>This command was implemented on the Cisco 7500 series routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Every time a probe is received, the `debug call fallback probe` command displays in the command-line interface (CLI) network traffic information collected by the probe. Use this command to monitor the network traffic information the probes carry as they enter the call fallback subsystem and log cache entries.

If you have frequent return of probes to your router, enabling this command can cause delays in your routing functions as the debug statistics are constantly compiled and sent to your terminal. Also, debug messages on your terminal may make for difficult CLI configuring.

**Examples**

The following example depicts a call coming in to 10.1.1.4 and codec type g729r8. Because there is no cache entry for this IP address, a g729r8 probe is initiated. The probe consists of 20 packet returns with an average delay of 43 milliseconds. The "jitter out" is jitter from source to destination router and "jitter in" is jitter from destination to source router. The delay, loss, and Calculated Planning Impairment Factor (ICPIF) values following g113_calc_icpif are the instantaneous values, whereas those values following "New smoothed values" are the values after applying the smoothing with weight 65.

```
Router# debug call fallback probe
```
2d19h:fb_initiate_probe: Probe payload is 32
2d19h:fb_main: NumOFRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0-> 10.1.1.4, codec:g729r8
2d19h:g113_calc_icpif(delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0
2d19h:fb_main: Probe timer expired, 10.1.1.4, codec:g729r8
2d19h:fb_main: NumOFRTT=20, RTTSum=120, loss=0, delay=43, jitter in=0, jitter out=0-> 10.1.1.4, codec:g729r8
2d19h:g113_calc_icpif(delay (w/codec delay)=43, loss=0, expect_factor=10) Icpif=0
2d19h:fb_main: New smoothed values: inst_weight=65, ICPIF=0, Delay=43, Loss=0 -> 10.1.1.4, codec:g729r8
debug call filter detail

To display details of the debug trace inside the generic call filter module (GCFM), use the debug call filter detail command in privileged EXEC mode. To disable debugging output, use the no form of this command.

d debug call filter detail
no debug call filter detail

Syntax Description

This command has no arguments or keywords.

Command Default

No default behavior or values

Command Modes

Privileged EXEC

Command History

Release Modification
12.3(4)T This command was introduced.

Examples

The following sample output from the debug call filter detail command shows the detailed activity of the GCFM, which is the internal module that controls the debug filtering.

Router# debug call filter detail
5d18h: gcfm_call_get_hash_address: hashtable index = 345
5d18h: gcfm_call_search_hash:no found
5d18h: gcfm_init_call_record:
5d18h: gcfm_init_percall_matchlist:
5d18h: === list 1: service_state=2, callp's: 0
5d18h: gcfm_call_get_hash_address: hashtable index = 345
5d18h: gcfm_call_enlist: count before this enlist 0 on 624D6000
5d18h: gcfm_call_enlist: tail is empty guid=C2E4C789-214A-11D4-804C-000A8A389BA8
5d18h: gcfm_call_get_hash_address: hashtable index = 345
5d18h: gcfm_call_search_hash: search requested guid=C2E4C789-214A-11D4-804C-000A8A389BA8
vs the entry guid=C2E4C789-214A-11D4-804C-000A8A389BA8
5d18h: gcfm_call_search_hash: found
5d18h: gcfm_update_percall_condlist_context:
5d18h: gcfm_update_percall_condlist_context: check cond = 2
5d18h: gcfm_copy_match_cond:
5d18h: gcfm_update_cond_through_matchlist:
5d18h: gcfm_check_percond_with_matchlist: check match-list 1
5d18h: gcfm_matchlist_perCond_Check:
5d18h: gcfm_matchlist_percond_check: check cond=2
5d18h: gcfm_matchlist_percond_check: compare 42300 to configured 42300
5d18h: gcfm_check_cond_tel_number:
5d18h: gcfm_check_cond_tel_number: matched
5d18h: gcfm_matchlist_percond_check: checked result is 1
5d18h: gcfm_is_bitfield_identical:
5d18h: gcfm_update_cond_through_matchlist: service=1, percallmatchlist tag=1,current_status
   1, service_filter=0
5d18h: gcfm_percall_notify_condition: not linked call record

The table below describes the significant fields shown in the display.
### Table 8: debug call filter detail Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5d18h: gcfm_init_percall_matchlist:</td>
<td>Shows that the filtering has been initiated.</td>
</tr>
<tr>
<td>5d18h: gcfm_call_enlist: tail is empty</td>
<td>guid=C2E4C789-214A-11D4-804C-000A8A389B8A8</td>
</tr>
<tr>
<td>5d18h: gcfm_check_percond_with_matchlist:</td>
<td>check match-list 1</td>
</tr>
<tr>
<td>5d18h: gcfm_matchlist_percond_check: checked</td>
<td>result is 1</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug call filter inout</td>
<td>Displays the debug trace inside the GCFM.</td>
</tr>
<tr>
<td>debug condition match-list</td>
<td>Runs a filtered debug on a voice call.</td>
</tr>
<tr>
<td>show call filter components</td>
<td>Displays the components used for filtering calls.</td>
</tr>
</tbody>
</table>
debug call filter inout

To display the debug trace inside the generic call filter module (GCFM), use the debug call filter inout command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug call filter inout
no debug call filter inout
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(4)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

The following sample output from the `debug call filter inout` command shows the incoming and outgoing activity of the GCFM, which is the internal module that controls the debug filtering.

```
Router# debug call filter inout
5d18h: gcfm_generate_guid: component ISDN gets guid
5d18h: gcfm_percall_register: component ISDN
5d18h: gcfm_percall_register: component ISDN return selected=0
5d18h: gcfm_percall_notify_condition: component ISDN for sync=1
5d18h: gcfm_check_percall_status: component ISDN return selected=0
5d18h: gcfm_check_percall_status: component TGRM
5d18h: gcfm_perccall_register: component VTSP for return selected value 0
5d18h: gcfm_perccall_register: component VTSP for sync=1
5d18h: gcfm_perccall_register: component VTSP for return selected value 0
5d18h: gcfm_perccall_register: component CCAPI
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION for return selected=0
5d18h: gcfm_perccall_register: component VOICE-IVR-V2
5d18h: gcfm_perccall_register: component VOICE-IVR-V2 for return selected value 0
5d18h: gcfm_perccall_register: component NUMBER-TRANSLATION
5d18h: gcfm_perccall_register: component NUMBER-TRANSLATION
5d18h: gcfm_perccall_register: component DIAL-PEER return selected=0
5d18h: gcfm_check_percall_status: component DIAL-PEER
```
5d18h: gcfm_check_percall_status: component DIAL-PEER return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component DIAL-PEER
5d18h: gcfm_calclear_condition: component CCAPI
5d18h: gcfm_calclear_condition: component VOICE-IVR-V2
5d18h: gcfm_calclear_condition: component VOICE-IVR-V2 successfully
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION
5d18h: gcfm_check_percall_status: component NUMBER-TRANSLATION return selected=0

Table 9: debug call filter inout Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcfm_generate_guid:</td>
<td>Shows that a GUID has been generated.</td>
</tr>
<tr>
<td>gcfm_perccall_register:</td>
<td>Shows components that have been registered for the call.</td>
</tr>
<tr>
<td>gcfm_perccall_notify_condition:</td>
<td>Shows that a component has been notified of the call.</td>
</tr>
<tr>
<td>gcfm_check_percall_status:</td>
<td>Shows the status of a component of the call.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
gcfm_percall_register: | Shows that a component has been registered.
gcfm_clear_condition: | Shows that a condition is cleared for a component.
gcfm_percall_deregister: | Shows that a component has been deregistered.
gcfm_terminate_track_guid: | Shows that the router is no longer tracking the GUID.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
</table>
display debug call filter detail | Displays the details of the debug trace inside the GCFM. |
display debug condition match-list | Runs a filtered debug on a voice call. |
display show call filter components | Displays the components used for filtering calls. |
debug call rsvp-sync events

To display events that occur during Resource Reservation Protocol (RSVP) setup, use the `debug call rsvp-sync events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default call rsvp-sync events
no default call rsvp-sync events
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(X)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(5)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(5)T.</td>
</tr>
<tr>
<td>12.2(X)B</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(1)T</td>
<td>Support for the command was implemented in Cisco AS5850 images.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

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

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

```
00:03:25: QoS Primitive Event for Call id 0x1 : QoS Listen
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498
00:03:25: Hashed entry 0x1 in call table 0x61FC2498
00:03:25: Entry Not found
00:03:25: Parameters: localip: 10.19.101.117
00:03:25: remoteip: 10.19.101.116
00:03:25: QoSpcb : 0x61FC34D8
00:03:25: Response Status : 0
Starting timer for call with CallId 0x1 for 10000 secs
00:03:25: Handling QoS Primitive QoS Listen
```
00:03:25: Establishing RSVP RESV state : rsvp_request_reservation()
00:03:25: RSVP Confirmation required
00:03:25: QoS Primitive Event for Call id 0x1 : QoS Resv
00:03:25: Lookup to be done on hashkey 0x1 in hash table 0x61FC2498
00:03:25: Hashed entry 0x1 in call table 0x61FC2498
00:03:25: Initiating RVSP PATH messages to be Sent : reg_invoke_rsvp_advertise_sender()
00:03:25: RESV notification event received is : 2
00:03:25: Received RESVCONFIRM
00:03:25: RESV CONFIRM message received from 10.19.101.116 for RESV setup from 10.19.101.117
00:03:25: RESV event received is : 0
00:03:25: RESERVATIONS ESTABLISHED : CallId: 1     Stop timer and notify Session Protocol
of Success (ie. if notification requested)
00:03:25: Invoking spQoSresvCallback with Success

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call rsvp-sync</td>
<td>Enables synchronization between RSVP and the H.323 voice signaling protocol.</td>
</tr>
<tr>
<td>call rsvp-sync resv-timer</td>
<td>Sets the timer for RSVP reservation setup.</td>
</tr>
<tr>
<td>debug call rsvp-sync func-trace</td>
<td>Displays messages about the software functions called by RSVP synchronization.</td>
</tr>
<tr>
<td>show call rsvp-sync conf</td>
<td>Displays the RSVP synchronization configuration.</td>
</tr>
<tr>
<td>show call rsvp-sync stats</td>
<td>Displays statistics for calls that attempted RSVP reservation.</td>
</tr>
</tbody>
</table>
debug call rsvp-sync func-trace

To display messages about software functions called by Resource Reservation Protocol (RSVP), use the `debug call rsvp-sync func-trace` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug call rsvp-sync func-trace
no debug call rsvp-sync func-trace
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XI1</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(5)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(5)T.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

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

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

```
00:03:41: Entering Function QoS_Listen
00:03:41: Parameters:localip:10.10.101.116 :localport:17568
00:03:41:remoteip:10.10.101.117 :remoteport:0
00:03:41: Entering Function qos_dequeue_event
00:03:41: Entering Function process_queue_event
00:03:41: QoS Primitive Event for Call id 0x2 :QoS Listen
00:03:41: Entering Function get_pcb
00:03:41: Entering Function hash_tbl_lookup
00:03:41:Lookup to be done on hashkey 0x2 in hash table 0x61FAECDB
00:03:41: Entering Function hash_func
00:03:41:Hashed entry 0x2 in call table 0x61FAECDB
00:03:41:Entry Not found
00:03:41: Entering Function qos_dequeue_pcb
00:03:41: Entering Function qos_initialize_pcb
```
00:03:41: Parameters: localip:10.10.101.116
00:03:41: remoteip:10.10.101.117
00:03:41: QoSpcb :0x61FAFD18
00:03:41: Response Status :0
00:03:41: Entering Function hash_tbl_insert_entry
00:03:41: Entering Function hash_func
00:03:41: Handling QoS Primitive QoS Listen
00:03:41: Entering Function qos_dequeue_hash_port_entry
00:03:41: Entering Function qos_port_tbl_insert_entry
00:03:41: Entering Function hash_func
00:03:41: Doing RSVP Listen :rsvp_add_ip_listen_api()

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call rsvp-sync</td>
<td>Enables synchronization between RSVP and the H.323 voice signaling protocol.</td>
</tr>
<tr>
<td>call rsvp-sync resv-timer</td>
<td>Sets the timer for RSVP reservation setup.</td>
</tr>
<tr>
<td>debug call rsvp-sync events</td>
<td>Displays the events that occur during RSVP synchronization.</td>
</tr>
<tr>
<td>show call rsvp-sync conf</td>
<td>Displays the RSVP synchronization configuration.</td>
</tr>
<tr>
<td>show call rsvp-sync stats</td>
<td>Displays statistics for calls that attempted RSVP reservation.</td>
</tr>
</tbody>
</table>
debug call threshold

To see details of the trigger actions, use the `debug call threshold` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

`debug call threshold module`

`no debug call threshold`

**Syntax Description**

<table>
<thead>
<tr>
<th>module</th>
<th>The <code>module</code> argument can be one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• core</td>
<td>- Traces the resource information.</td>
</tr>
<tr>
<td>• detail</td>
<td>- Traces for detail information.</td>
</tr>
</tbody>
</table>

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>The command was integrated into Cisco IOS Release 12.2(4)T. Support for the Cisco AS5300, Cisco AS5350, and Cisco AS5400 is not included in this release.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T and implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>Support for this command was implemented on Cisco AS5850, Cisco AS5800, Cisco AS5300, Cisco AS5350, and Cisco AS5400 series images.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug call threshold core` command:
```
Router# debug call threshold core
RSCCAC Core info debugging is on
```

The following is sample output from the `debug call threshold detail` command:
```
Router# debug call threshold detail
All RSCCAC info debugging is on
```
debug call treatment action

To debug the call treatment actions, use the **debug call treatment action** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
debug call treatment action
no debug call treatment action
```

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

**Command Default**
Disabled

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XA</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>The command was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 platform.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(8)T and</td>
</tr>
<tr>
<td></td>
<td>implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>Support for this command was implemented on Cisco AS5850, Cisco</td>
</tr>
<tr>
<td></td>
<td>AS5800, Cisco AS5300, Cisco AS5350, and Cisco AS5400 series images.</td>
</tr>
</tbody>
</table>

**Examples**
Debug actions are performed on calls by call treatment. The following sample output shows that call treatment is turned on:

```
Router# debug call treatment action
Call treatment action debugging is on
```
debug callback

To display callback events when the router is using a modem and a chat script to call back on a terminal line, use the `debugcallback` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug callback
no debug callback
```

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

**Command Modes**
Privileged EXEC

**Usage Guidelines**
This command is useful for debugging chat scripts on PPP and AppleTalk Remote Access Protocol (ARAP) lines that use callback mechanisms. The output provided by the `debugcallback` command shows you how the call is progressing when used with the `debugppp` or `debugarap` commands.

**Examples**
The following is sample output from the `debugcallback` command:

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cable env</td>
<td>Displays ARAP events.</td>
</tr>
<tr>
<td>debug ppp</td>
<td>Displays information on traffic and exchanges in an internetwork implementing the PPP.</td>
</tr>
</tbody>
</table>
debug capf-server

To collect debug information about the CAPF server, use the `debug capf-server` command in privileged EXEC mode. To disable collection of debug information, use the `no` form of this command.

```
debug capf-server {all| error| events| messages}
no debug capf-server
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Collect all CAPF information available.</td>
</tr>
<tr>
<td>error</td>
<td>Collect only information about CAPF errors.</td>
</tr>
<tr>
<td>events</td>
<td>Collect only information about CAPF status events.</td>
</tr>
<tr>
<td>messages</td>
<td>Collect only CAPF system messages.</td>
</tr>
</tbody>
</table>

### Command Default

Collection of CAPF debug information is disabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(9)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(9)T.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

This command is used with Cisco Unified CallManager Express phone authentication.

### Examples

The following example shows debug messages for the CAPF server.

```
Router# debug capf-server all
001891: .Jul 21 18:17:07.014: %IPPHONE-6-UNREGISTER_NORMAL: ephone-1:SEP000E325C9A43
So cket:3 DeviceType:Phone has unregistered normally.
001897: .Jul 21 18:17:22.555: ephone_capf_send_auth_req:
```
Cisco IOS Debug Command Reference - Commands A through D
**debug cas**

To debug channel-associated signaling (CAS) messages and to debug the establishment of a time-division multiplexing (TDM) connection between a DS0 and a digital modem, use the `debug cas` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cas slot slot number port port number
no debug cas slot slot number port port number
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>slot</code></td>
<td>Slot and slot number. Valid values are 0 and 1.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>Port and port number. Valid values are 0 and 1.</td>
</tr>
</tbody>
</table>

**Command Default**

Disabled

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(7)T</td>
<td>This command was introduced for the Cisco AS5200 and AS5300 platforms.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(2)T and support was</td>
</tr>
<tr>
<td></td>
<td>added for the Cisco 2600 series and Cisco 3600 series platforms.</td>
</tr>
<tr>
<td>12.3(1)</td>
<td>This command was integrated into Cisco IOS Release 12.3(1) and support was</td>
</tr>
<tr>
<td></td>
<td>added for the Cisco 2600 XM series, Cisco 2691, and Cisco 3700 series</td>
</tr>
<tr>
<td></td>
<td>platforms.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

When the NM-xCE1T1PRI network module is used with an NM-xDM and a DS0-group is configured under the controller, you can use the `debug cas` command to debug CAS signaling messages and the establishment of a TDM connection between a DS0 and a digital modem. Use the `debug cas` command to identify and troubleshoot call connection problems on a T1/E1 interface. With this command, you can trace the complete sequence of incoming and outgoing calls.
The following shows an example session to enable debugging CAS and generate troubleshooting output:

Router# show debug

Router# debug cas slot 1 port 0

CAS debugging is on

Router# debug-cas is on at slot(1) dsx1(0)

Router# show debug

CAS debugging is on

The following example shows output for the first outgoing call:

Router# p 1.1.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

*Mar 2 00:17:45: dsx1_alloc_cas_channel: channel 0 dsx1_timeslot
1(0/0): TX SEIZURE (ABCD=0001)(0/0): RX SEIZURE ACK (ABCD=1101)(0/1):
RX_IDLE (ABCD=1001)(0/2): RX_IDLE (ABCD=1001)(0/3): RX_IDLE
(ABCD=1001)(0/4): RX_IDLE (ABCD=1001)(0/5): RX_IDLE (ABCD=1001)(0/6):
RX_IDLE (ABCD=1001)(0/7): RX_IDLE (ABCD=1001)(0/8): RX_IDLE
(ABCD=1001)(0/9): RX_IDLE (ABCD=1001)(0/10): RX_IDLE (ABCD=1001)(0/11):
RX_IDLE (ABCD=1001)(0/12): RX_IDLE (ABCD=1001)(0/13): RX_IDLE
(ABCD=1001)(0/14): RX_IDLE (ABCD=1001)(0/15): RX_IDLE (ABCD=1001)(0/16):
RX_IDLE (ABCD=1001)(0/17): RX_IDLE (ABCD=1001)(0/18): RX_IDLE
(ABCD=1001)(0/19): RX_IDLE (ABCD=1001)(0/20): RX_IDLE
(ABCD=1001)(0/21): RX_IDLE
(ABCD=1001)(0/22): RX_IDLE (ABCD=1001)(0/23): RX_IDLE
(ABCD=1001)(0/24): RX_IDLE (ABCD=1001)(0/25): RX_IDLE (ABCD=1001)(0/26):
RX_IDLE (ABCD=1001)(0/27): RX_IDLE (ABCD=1001)(0/28): RX_IDLE
(ABCD=1001)(0/29): RX_IDLE (ABCD=1001)(0/30): RX_IDLE
(ABCD=1001)...(0/0): RX ANSWERED (ABCD=0101).

Success rate is 0 percent (0/5)

Router# *Mar 2 00:18:13.333: %LINK-3-UPDOWN: Interface Async94, changed state to up
*Mar 2 00:18:13.333: %DIALER-6-BIND: Interface As94 bound to profile D11
*Mar 2 00:18:14.577: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async94, changed state to up

Router# p 1.1.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 160/180/236 ms

The following example shows that the call is cleared on the router:

Router# clear int dialer 1

Router# (0/0): TX IDLE (ABCD=1001)(0/0): RX IDLE (ABCD=1001)

*Mar 2 00:18:28:617: %LINK-5-CHANGED: Interface Async94, changed state to reset
*Mar 2 00:18:28:617: %DIALER-6-UNBIND: Interface As94 unbound from profile D11

*Mar 2 00:18:29:617: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async94, changed state to down

et2-c3745-1#

*Mar 2 00:18:33.617: %LINK-3-UPDOWN: Interface Async94, changed state to down

The following example shows a subsequent outbound CAS call:

Router# p 1.1.1.2

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:

*Mar 2 00:18:40: dsx1_alloc_cas_channel: channel 5 dsx1_timeslot
6(0/0): TX SEIZURE (ABCD=0001)(0/0): RX SEIZURE_ACK
(ABCD=1101)....(0/5): RX ANSWERED (ABCD=0101).

Success rate is 0 percent (0/5)

Router# *Mar 2 00:19:08.841: %LINK-3-UPDOWN: Interface Async93, changed state to up
*Mar 2 00:19:08.841: %DIALER-6-BIND: Interface As93 bound to profile D11
*Mar 2 00:19:10.033: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async93, changed state to up
Router# p 1.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 160/167/176 ms
The following example shows the call cleared by the switch:

Router# (0/5): TX IDLE (ABCD=1001)(0/5): RX IDLE (ABCD=1001)
*Mar 2 00:19:26.249: %LINK-5-CHANGED: Interface Async93, changed state to reset
*Mar 2 00:19:26.249: %DIALER-6-UNBIND: Interface As93 unbound from profile Di1
*Mar 2 00:19:27.249: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async93, changed state to down
Router#
*Mar 2 00:19:31.249: %LINK-3-UPDOWN: Interface Async93, changed state to down
The following example shows an incoming CAS call:

Router# (0/0): RX SEIZURE (ABCD=0001)
*Mar 2 00:22:40: dsx1_alloc_cas_channel: channel 0 dsx1_timeslot
1(0/0): TX SEIZURE_ACK (ABCD=1101)(0/0): TX ANSWERED (ABCD=0101)
Router#
*Mar 2 00:23:06.249: %LINK-3-UPDOWN: Interface Async83, changed state to up
*Mar 2 00:23:06.249: %DIALER-6-BIND: Interface As83 bound to profile Di1
*Mar 2 00:23:07.653: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async83, changed state to up

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>show debug</td>
<td>Displays information about the types of debugging that are enabled for your router.</td>
</tr>
</tbody>
</table>
debug ccaal2 session

To display the ccaal2 function calls during call setup and teardown, use the `debug ccaal2 session` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccaal2 session
no debug ccaal2 session
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

Debugging for ATM Adaptation Layer type 2 (AAL2) sessions is not enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)XA</td>
<td>This command was introduced for the Cisco MC3810 series.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated in Cisco IOS Release 12.1(2)T.</td>
</tr>
<tr>
<td>12.2(2)T</td>
<td>Support for this command was implemented on the Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

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

**Examples**

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

```
Router# debug ccaal2 session
CCAAL2 Session debugging is on
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# voice-port 2/0:0
Router(config)# shutdown
00:32:45:ccaal2_call_disconnect:peer tag 0
00:32:45:ccaal2_evhandle_call_disconnect:Entered
00:32:45:ccaal2_call_cleanup:freeccb 1, call_disconnected 1
00:32:45:starting incoming timer:Setting accept_incoming to FALSE and
00:32:45:timer 2:(0x622F6270)starts - delay (70000)
00:32:45:ccaal2_call_cleanup:Generating Call record
00:32:45:cause=81 tcause=81 cause_text=unspecified
00:32:45:ccaal2_call_cleanup:ccb 0x63FF1700, vdbPtr 0x62DFF2E0
freeccb_flag=1, call_disconnected_flag=1
00:32:45:%LINK-3-UPDOWN:Interface receive and transmit2/0:0(1),
    changed state to Administrative Shutdown
```
The following example shows sample output from the `debug ccaal2 session` command for a trunk setup on a voice port:

```
Router# debug ccaal2 session
Router(config-voiceport)# no shutdown
Router(config-voiceport)#
00:35:28:%LINK-3-UPDOWN:Interface receive and transmit[2/0:0(1)],
changed state to up
00:35:35:ccaal2_call_setup_request:Entered
00:35:35:ccaal2_evhandle_call_setup_request:Entered
00:35:35:ccaal2_initialize_ccb:preferred_codec set(-1)(0)
00:35:35:ccaal2_evhandle_call_setup_request:preferred_codec
set(5)(40). VAD is 1
00:35:35:ccaal2_call_setup_trunk:subchannel linking
successful ccaal2_receive:xmitFunc is NULL
00:35:35:ccaal2_caps_ind:PeerTag = 49
00:35:35: codec(preferred) = 1, fax_rate = 2, vad = 2
00:35:35: cid = 56, config_bitmask = 258, codec_bytes = 40,
signal_type=8
00:35:36:%HTSP-5-UPDOWN:Trunk port(channel) [2/0:0(1)] is up
Router(config-voiceport)#
```

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show debug</code></td>
<td>Shows which debug commands are enabled.</td>
</tr>
</tbody>
</table>
debug cce dp named-db urlfilter

To enable debug information of the Common Classification Engine Data-Plane (CCE DP) URL Filtering Classification module, use the `debug cce dp named-db urlfilter` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

ddebug cce dp named-db urlfilter

no debug cce dp named-db urlfilter

Syntax Description

This command has no keywords or arguments.

Command Default

No debugging information is generated for the the CCE DP URL Filtering Classification module.

Command Modes

Privileged EXEC (#)

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(15)XZ</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(20)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(20)T.</td>
</tr>
</tbody>
</table>

Examples

The following is sample output from the `debug cce dp named-db urlfilter` command at the time that a URL request to the untrusted domain www.example.com was made:

```
Router# debug cce dp named-db urlfilter
CCE DP Named DB URLF functionality debugging is on
Router# *Apr 4 10:38:08.043: CCE* FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't get token
*Apr 4 10:38:08.043: CCE* FUNC: cce_dp_urlf_truncate_url -- Truncating URL upto script before sending to the trend for classification
*Apr 4 10:38:08.043: CCE* FUNC: urlf_trend_find_cache_entry -- The host tree in bucket 1248 is empty
*Apr 4 10:38:08.043: CCE* FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't find in cache
*Apr 4 10:38:08.051: CCE FUNC: urlf_trend_store_response -- Host node with given domain name not found.
*Apr 4 10:38:08.051: CCE FUNC: urlf_trend_store_response -- Create domain type cache entry.
*Apr 4 10:38:08.051: CCE FUNC: cache_size_limit_check -- New cache size=73, existing cache size=0, cache size limit=131072000
*Apr 4 10:38:08.051: CCE FUNC: create_domain_cache_entry -- Domain cache entry 0x65EE0ED0 created.
*Apr 4 10:38:08.051: CCE FUNC: create_and_insert_domain_cache_entry --
*Apr 4 10:38:08.051: Domain cache entry 0x65EE0ED0 created and inserted into host tree with root=0x65EE0ED0, root left=0x0, root right=0x0; new node left=0x0, new node right=0x0
*Apr 4 10:38:08.051: CCE FUNC: cce_dp_named_db_urlf_gen_match_token -- pushing match-info token - class 0x0C000000E; filter 45; category 21
*Apr 4 10:38:08.051: CCE FUNC: cce_dp_named_db_urlf_non_pkt_classify -- Class 0x65C5D484 matched
*Apr 4 10:38:08.051: %URLF-4-URL_BLOCKED: Access denied URL 'http://www.example.com/', client 1.0.0.118:3056 server 192.168.0.30:8080
```
debug backhaul-session-manager session through debug channel packets

debug cce dp named-db urlfilter

*Apr 4 10:38:08.055: CCE* FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't get token
*Apr 4 10:38:08.055: CCE FUNC: cce_dp_named_db_urlf_pkt_classify -- Didn't get token
debug ccf11 session

To display the ccf11 function calls during call setup and teardown, use the **debug ccf11 session** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug ccf11 session**

**no debug ccf11 session**

### Syntax Description

This command has no keywords or arguments.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)XG</td>
<td>This command was introduced for the Cisco 2600 and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.0(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.0(4)T.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was first supported on the Cisco MC3810 series.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>Support for this command was implemented in Cisco MC3810 images.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

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

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

### Examples

The following is sample output from the **debug ccf11 session** command:

```plaintext
Router# debug ccf11 session
INCOMING CALL SETUP (port setup for answer-mode):
*Mar 6 18:04:07.693:ccfrf11_process_timers:scb (0x60EB6040) timer (0x60EB6098) expired
*Mar 6 18:04:07.693:Setting accept_incoming to TRUE
*Mar 6 18:04:11.213:ccfrf11_incoming_request:peer tag 800:callingNumber=+2602100,
calledNumber=+3622110
*Mar 6 18:04:11.213:ccfrf11_initialize_ccb:preffered_codec set(-1)(0)
*Mar 6 18:04:11.217: codec(preferred) = 4, fax_rate = 2, vad = 2
*Mar 6 18:04:11.217: cid = 30, config_bitmask = 0, codec_bytes = 20, signal_type=2
*Mar 6 18:04:11.217: required_bandwidth 8192
*Mar 6 18:04:11.221:ccfrf11_evhandle_call_connect:Entered
```
CALL SETUP (MASTER):
5d22h:ccfrf11_call_setup_request:Entered
5d22h:ccfrf11_evhandle_call_setup_request:Entered
5d22h:ccfrf11_initialize_ccb:preffered_codec set(-1)(0)
5d22h:ccfrf11_evhandle_call_setup_request:preffered_codec set(9)(24)
5d22h:ccfrf11_call_setup_trunk:subchannel linking successful
5d22h:ccfrf11_caps_ind:PeerTag = 810
5d22h: codec(preferred) = 512, fax_rate = 2, vad = 2
5d22h: cid = 30, config_bitmask = 1, codec_bytes = 24, signal_type=2
5d22h: required_bandwidth 6500
5d22h:ccfrf11_caps_ind:Bandwidth reservation of 6500 bytes succeeded.

CALL TEARDOWN:
*Mar 6 18:09:14.805:ccfrf11_call_cleanup:freeccb 1, call_disconnected 1
*Mar 6 18:09:14.809:timer 2:(0x60EB6098)starts - delay (70000)
*Mar 6 18:09:14.809:timer 1:(0x60F64104) stops
*Mar 6 18:09:14.809:ccfrf11_call_cleanup:ccb 0x60F6404C, vdbPtr 0x610DB7A4
freeccb_flag=1, call_disconnected_flag=1

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>debug call rsvp-sync events</strong></td>
<td>Displays the ccsvoice function calls during call setup and teardown.</td>
</tr>
<tr>
<td><strong>debug ccsvoice vofr-session</strong></td>
<td>Displays the ccsvoice function calls during call setup and teardown.</td>
</tr>
<tr>
<td><strong>debug vtsp session</strong></td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
debug cch323

To provide debugging 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

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

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>The debug cch323 command and the following keywords were introduced: h225, h245, and ras.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>The nxe keyword was added.</td>
</tr>
</tbody>
</table>
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.

This command was implemented on the Cisco AS5850.

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:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>Enables a CCH323 Service Provider Interface (SPI) trace.</td>
</tr>
<tr>
<td>h225</td>
<td>Enables an H225 state machine debugging trace.</td>
</tr>
<tr>
<td>h245</td>
<td>Enables an H245 state machine debugging trace.</td>
</tr>
<tr>
<td>nxe</td>
<td>Enables an Annex E debugging trace.</td>
</tr>
<tr>
<td>ras</td>
<td>Enables a RAS state machine debugging trace.</td>
</tr>
<tr>
<td>rawmsg</td>
<td>Enables a CCH323 RAWMSG debugging trace.</td>
</tr>
<tr>
<td>session</td>
<td>Enables a Session debugging trace.</td>
</tr>
</tbody>
</table>

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_UCC--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 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 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 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: 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: 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
```

The table below describes the significant fields shown in the display.

**Table 10: debug cch323 h225 Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H225_EVENT_SETUP</td>
<td>This event instructs the H.225 state machine to send a setup message to the peer.</td>
</tr>
<tr>
<td>H225_IDLE</td>
<td>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.</td>
</tr>
<tr>
<td>H225_SETUP</td>
<td>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.</td>
</tr>
<tr>
<td>SETUPCFM_CHOSEN</td>
<td>The H225 connect message that has been received from a remote H323 endpoint.</td>
</tr>
<tr>
<td>H225_EVENT_SETUP CFM_IND</td>
<td>This event indicates to the H.225 state machine that a connect message arrived from the peer.</td>
</tr>
<tr>
<td>H225_ACTIVE</td>
<td>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.</td>
</tr>
</tbody>
</table>
This event indicates to the H.225 state machine that the pending H.245 operation succeeded.

This event instructs the H.225 state machine to send a release complete message to the peer.

The state in which the H.225 state machine is waiting for the completion of the DRQ process from the RAS state machine.

This event indicates to the H.225 state machine that the pending RAS operation succeeded.

The finite state machine.

### Examples

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

```plaintext
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
```

The following is sample output from a typical `debug cch323 h245` request on a Cisco 3640 router:
The table below describes the significant fields shown in the display.

**Table 11: debug cch323 h245 Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H245_EVENT_MSD</td>
<td>Send MSD event message to the state machine.</td>
</tr>
<tr>
<td>H245_MS FSM</td>
<td>An H225 master slave determination finite state machine.</td>
</tr>
<tr>
<td>H245_MS_WAIT</td>
<td>The initial state of the MSD state machine.</td>
</tr>
<tr>
<td>H245_EVENT_CAP</td>
<td>Send CAP event message.</td>
</tr>
<tr>
<td>H245_CAP FSM</td>
<td>This is the H245 terminal CAP finite state machine.</td>
</tr>
<tr>
<td>H245_CAP_WAIT</td>
<td>The initial state of the CAP state machine.</td>
</tr>
<tr>
<td>H245_CAP_WAIT</td>
<td>In this state, a CAP message is sent, and the device is waiting for the reply.</td>
</tr>
<tr>
<td>M_H245_MS_DETERMINE_INDICATION</td>
<td>The MSD message that has been received by an H245 terminal from a remote H323 endpoint.</td>
</tr>
<tr>
<td>H245_EVENT_MS_IND</td>
<td>Received MSD event message.</td>
</tr>
<tr>
<td>M_H245_CAPTRANSFER_INDICATION</td>
<td>A CAP message that has been received by the H245 terminal from an H323 remote endpoint.</td>
</tr>
<tr>
<td>H245_EVENT_CAP_IND</td>
<td>Received CAP event message.</td>
</tr>
<tr>
<td>M_H245_MS_DETERMINE_CONFIRM</td>
<td>A confirmation message that the H245 master slave termination message was sent.</td>
</tr>
<tr>
<td>H245_EVENT_MS_CFM</td>
<td>Send MSD acknowledge event message.</td>
</tr>
<tr>
<td>H245_MS_DONE</td>
<td>The result is in.</td>
</tr>
<tr>
<td>M_H245_CAPTRANSFER_CONFIRM</td>
<td>An indication to the H245 terminal that the CAP message was sent.</td>
</tr>
<tr>
<td>H245_EVENT_CAP_CFM</td>
<td>Send CAP acknowledge event message.</td>
</tr>
<tr>
<td>H245_CAP_DONE</td>
<td>The result is in.</td>
</tr>
<tr>
<td>H245_EVENT_OLC</td>
<td>Send OLC event message.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
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 acknowledgment 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.

#### Examples

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 pdus
00:15:54:nxe_receive_ack:Ack received for seqnum=13 from IPAddr [0xA4D4A02], Port [2517]
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]
```

---

**Cisco IOS Debug Command Reference - Commands A through D**

82
Examples

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_ras_receiver: received msg of type RCF_CHOSEN
  cch323_run_ras_sm: changing to CCH323_RAS_STATE_IDLE state
  cch323_ras_receiver: received msg of type RCF_CHOSEN
20:58:59: cch323_percall_ras_sm: received event CCH323_RAS_EVENT_NEWCALL while at CCH323_RAS_STATE_IDLE state
  cch323_ras_receiver: received msg of type ACF_CHOSEN
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: 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
  cch323_ras_receiver: received msg of type DCF_CHOSEN
20:59:02: cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_DRQ
  cch323_ras_receiver: received msg of type DCF_CHOSEN
20:59:04: cch323_percall_ras_sm: changing to new state CCH323_RAS_STATE_ACTIVE
```

The table below describes the significant fields shown in the display.

Table 12: `debug cch323 ras` Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCH323_RAS_EVENT_SEND_RRQ</td>
<td>Send RRQ event message.</td>
</tr>
<tr>
<td>CCH323_RAS_STATE_IDLE</td>
<td>The global state machine is in the idle state.</td>
</tr>
<tr>
<td>CCH323_RAS_STATE_RRQ</td>
<td>The state machine is in the RRQ state. In this state, the gateway is registering with a gatekeeper.</td>
</tr>
<tr>
<td>RCF_CHOSEN</td>
<td>A registration confirm message that has been received from a gatekeeper.</td>
</tr>
<tr>
<td>CCH323_RAS_EVENT_RCF</td>
<td>Received RCF event message.</td>
</tr>
<tr>
<td>CCH323_RAS_EVENT_NEWCALL</td>
<td>New call event.</td>
</tr>
<tr>
<td>CCH323_RAS_STATE_ARQ</td>
<td>The per-call state machine is in the process of admitting a new call.</td>
</tr>
<tr>
<td>ACF_CHOSEN</td>
<td>ACF message that has been received from a gatekeeper.</td>
</tr>
<tr>
<td>CCH323_RAS_EVENT_ACF</td>
<td>Received ACF event message.</td>
</tr>
</tbody>
</table>
The per-call state machine is in the call active state.

CCH323_RAS_EVENT_CALLDISC

Call disconnect event message.

CCH323_RAS_STATE_DRQ

The per-call state machine is in the process of disengaging an active call.

DCF_CHOSEN

The disengage confirm message that has been received from a gatekeeper.

CCH323_RAS_EVENT_DCF

Received DCF event message.

CCH323_RAS_EVENT_IRR

Send IRR event message.

Examples

The following is sample output from a typical debugcch323rawmsg request on a Cisco 3640 router:

Router# debug cch323 rawmsg
00:32:04:cch323_h225_progress_ind: raw message is 4 bytes: 1E 02 81 88
00:32:22:cch323_h225_release_ind: raw message is 80 bytes: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:32:22:cch323_h225_release_notify: raw message is 80 bytes: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Examples

Following are two examples of output using the debugcch323session command and keyword combination. The first example is for a call setup on an originating gateway. The second example is for a call setup on a terminating gateway.

The following is sample output from a typical debugcch323session request for a call setup on an originating gateway:

Router# debug cch323 session
00:33:49:cch323_call_setup: gw_id=1, callID=16
00:33:49:cch323_get_new_cbb: ccb (0x81D12D2C) is in use
00:33:49:cch323_call_setup: inserted ccb
00:33:49:cch323_get_peer_info: preferred codec set to G729 IETF with Bytes = 20
00:33:49:cch323_get_peer_info: peer->voice_peer_tag: 12D, ccb: 81D12D2C
00:33:49: Call_setup Playout Mode: 0, Init 60, Min 40, Max 200
00:33:49: No account/pin number available
00:33:49:cch323_call_setup_normal: for callID 10
00:33:49: timer (0x81D1304) starts - delay (15000)
00:33:49:cch323_ct_main: SOCK 1 Event 0x1
00:33:49: timer (0x81D1304) stops
00:33:49: No account/pin number available
00:33:49: generic_open_logical_channel: codec is G.729 IETF
00:33:49: generic_open_logical_channel: codec is G.729
00:33:49: timer (0x81D1304) 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] owner_data=0x81D13C88, len=105, msgPtr=0x81D07608
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]towner_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_setup_ind
00:23:27:Not using Voice Class Codec
00:23:27:Near-end Pref Codecs = G.729 IETF
00:23:27:cch323_build_fastStart_cap_response:In Response Filling in qosCapability field to 0
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear h323 gateway</td>
<td>Clears the H.323 gateway counters.</td>
</tr>
<tr>
<td>debug h323-annexg</td>
<td>Displays all pertinent AnnexG messages that have been transmitted and received.</td>
</tr>
<tr>
<td>debug voip rawmsg</td>
<td>Displays the raw message owner, length, and pointer.</td>
</tr>
<tr>
<td>show h323 gateway</td>
<td>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.</td>
</tr>
</tbody>
</table>
**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.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug cch323 capacity` command to track the maximum and current call capacity values in the Registration, Admission, and Status (RAS) Protocol 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 is sample output from the `debug cch323 capacity` command:

```
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
    Active Voice Calls(In): 5, Active Voice Calls(Out): 7

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

The table below describes the significant fields shown in the display.
```
Table 13: debug cch323 capacity Update Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registered</td>
<td>Gateway registration:</td>
</tr>
<tr>
<td></td>
<td>• 0=Gateway is not registered to the gatekeeper</td>
</tr>
<tr>
<td></td>
<td>• 1=Gateway is registered to the gatekeeper at the time of the change</td>
</tr>
<tr>
<td>Event</td>
<td>Carriers updated:</td>
</tr>
<tr>
<td></td>
<td>• 0=All carriers updated</td>
</tr>
<tr>
<td></td>
<td>• 1=Single carrier updated</td>
</tr>
<tr>
<td>Reason</td>
<td>Reason for the update notification:</td>
</tr>
<tr>
<td></td>
<td>• 0=CURRENT_CAPACITY_UPDATE</td>
</tr>
<tr>
<td></td>
<td>• 1=MAX_CAPACITY_UPDATE</td>
</tr>
<tr>
<td></td>
<td>• 2=BOTH_CAPACITY_UPDATE</td>
</tr>
<tr>
<td>CarrierID</td>
<td>ID of the trunk group or carrier to which the change applies.</td>
</tr>
</tbody>
</table>

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
  Active Voice Calls(In): 5, Active Voice Calls(Out): 7
```

The table below describes the significant fields shown in the display.

Table 14: debug cch323 capacity Call Capacity Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupID</td>
<td>The circuit’s carrier identification (ID) or trunk group label.</td>
</tr>
<tr>
<td>Maximum Channels in Group</td>
<td>Maximum number of physical (or configured) circuits.</td>
</tr>
<tr>
<td>Max. Voice Calls(In)</td>
<td>Maximum number of allowed incoming voice and data calls.</td>
</tr>
<tr>
<td>Max. Voice Calls(Out)</td>
<td>Maximum number of allowed outgoing voice and data calls.</td>
</tr>
</tbody>
</table>
### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Voice Calls(In)</td>
<td>Current number of active incoming voice and data calls.</td>
</tr>
<tr>
<td>Active Voice Calls(Out)</td>
<td>Current number of active outgoing voice and data calls.</td>
</tr>
<tr>
<td>Max. Voice Calls(to GK)</td>
<td>Maximum call capacity value to be sent to the gatekeeper in the RAS message.</td>
</tr>
<tr>
<td>Avail. Voice Calls(to GK)</td>
<td>Available call capacity value to be sent to the gatekeeper in the RAS message.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endpoint circuit-id h323id</td>
<td>Associates a carrier with a non-Cisco endpoint.</td>
</tr>
</tbody>
</table>
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.

d debug cch323 h225
no debug cch323 h225

Syntax Description

This command has no keywords or arguments.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

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 Registration, Admission, and Status Protocol (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.
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:Changing from H225_ACTIVE state to H225_WAIT_FOR_DRQ state
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:H225 FSM:received event H225_EVENT_RAS_SUCCESS
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 arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines

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

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

State Definitions

The definitions are as follows:

- H245_MS_NONE-- This is the initial state of the master slave determination state machine.
- H245_MS_WAIT--In this state, a Master Slave Determination message is sent, 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 `debugcch323h245` 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_MS_DETERMINE_CONFIRM
20:58:23:Changing to new event H245_EVENT_MS_CF
20:58:23:H245 CAP FSM:received event H245_EVENT_MS_CF while at state H245_CAP_WAIT
20:58:23:cch323_h245_receiver:received msg of type M_H245_MS_CF
20:58:23:Changing to new event H245_EVENT_MS_CF_CONFIRM
20:58:23:H245 MS FSM:received event H245_EVENT_MS_CF_CONFIRM while at state H245_MS_WAIT
20:58:23:changing from H245_MS_WAIT state to H245_MS_DONE state
20:58:23:cch323_h245_receiver:received msg of type M_H245_MS_CF_CONFIRM
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_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_WAIT state
20:58:23:changing from H245_OLC_WAIT state to H245_OLC_DONE 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
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 debugging output, use the `no` form of this command.

```
debug cch323 preauth
no debug cch323 preauth
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**
The following is debugging output for a single H.323 call:

```
Router# debug cch323 preauth
cch323 preempt tracing is enabled
Jan 23 18:39:56.393: In cch323_send_preauth_req for preauth_id = -1
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 685258BC , 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: 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 685258BC ccb
```

The table below describes the significant fields shown in the display.
### Table 15: debug cch323 preauth Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>Request Type--0 for preauthentication, 1 for disconnect.</td>
</tr>
<tr>
<td>Preauth id</td>
<td>Identifier for the preauthentication request.</td>
</tr>
<tr>
<td>EndPt Type</td>
<td>Call Origin End Point Type--1 for IP address, 2 for IZCT value.</td>
</tr>
<tr>
<td>EndPt</td>
<td>Call Origin End Point Value--An IP address or IZCT value.</td>
</tr>
<tr>
<td>Resource Service</td>
<td>Resource Service Type--1 for Reservation, 2 for Query.</td>
</tr>
<tr>
<td>Call_origin</td>
<td>Answer.</td>
</tr>
<tr>
<td>Call_type</td>
<td>Voice over IP (VoIP).</td>
</tr>
<tr>
<td>Calling_num</td>
<td>Calling Party Number (CLID).</td>
</tr>
<tr>
<td>Called_num</td>
<td>Called Party Number (DNIS).</td>
</tr>
<tr>
<td>Protocol</td>
<td>0 for H.323, 1 for SIP.</td>
</tr>
<tr>
<td>function reports</td>
<td>Various identifiers and status reports for executed functions.</td>
</tr>
</tbody>
</table>
debug cch323 ras

To provide the trace of the state transition of the Registration, Admission, and Status (RAS) Protocol 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.

```
deped cch323 ras
no debug cch323 ras
```

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

**Command Default**
Disabled

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(6)NA2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T.</td>
</tr>
</tbody>
</table>

**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 Gatekeeper Request (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 Registration Request (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 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.

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 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 Rejection (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 URQ.
• CCH323_RAS_EVENT_URQ--Received URQ.
• CCH323_RAS_EVENT_UCF--Received Unregister Confirmation (UCF).
• CCH323_RAS_EVENT_SEND_UUCF--Send Unregister Confirmation (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 Disengage Confirm (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.

Examples
The following is sample output from the debugcch323preauth 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

20:58:59: cch323_percall_ras_sm: received event CCH323_RAS_EVENT_RCF while at CCH323_RAS_STATE_RRQ state
20:58:59: cch323_ras_receiver: received msg of type ACF_CHOSEN

20:59:02: cch323_percall_ras_sm: received event CCH323_RAS_EVENT_CALLDISC while at CCH323_RAS_STATE_ACTIVE state
20:59:02: cch323_ras_receiver: received msg of type DCF_CHOSEN

20:59:04: cch323_percall_ras_sm: received event CCH323_RAS_EVENT_IRR while at CCH323_RAS_STATE_ACTIVE state

Cisco IOS Debug Command Reference - Commands A through D
debug cch323 video

To provide debugging output for video components within the H.323 subsystem, use the \texttt{debug cch323 video} command in privileged EXEC mode. To disable debugging output, use the \texttt{no} form of this command.

\begin{verbatim}
debug cch323 video
no debug cch323 video
\end{verbatim}

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Cisco IOS Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(4)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(9)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(9)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command to enable a debugging trace for the video component in an H.323 network.

**Examples**

The following is sample output of the debugging log for an originating Cisco Unified CallManager Express (Cisco Unified CME) gateway after the \texttt{debug cch323 video} command was enabled:

\begin{verbatim}
Router# show log
Syslog logging: enabled (11 messages dropped, 487 messages rate-limited, 0 flushes, 0 overruns, xml disabled, filtering disabled)
   Console logging: disabled
   Monitor logging: level debugging, 0 messages logged, xml disabled, filtering disabled
   Buffer logging: level debugging, 1144 messages logged, xml disabled, filtering disabled
   Logging Exception size (4096 bytes)
   Count and timestamp logging messages: disabled
   Trap logging: level informational, 1084 message lines logged
Log Buffer (6000000 bytes):
Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_get_peer_info: Entry
Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_set_pref_codec_list: First preferred codec(bytes)=16(20)
Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_get_peer_info: Have peer
Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_set_h323_control_options_outgoing: h245 sm mode = 8463
\end{verbatim}
Jun 13 09:19:42.006: //103030/C7838B198002/H323/cch323_set_h323_control_options_outgoing: h323_ctl=0x20
Jun 13 09:19:42.010: //103030/C7838B198002/H323/cch323_rotary_validate: No peer_ccb available

Examples

The following is sample output of the debugging log for a terminating Cisco Unified Survivable Remote Site Telephony (Cisco Unified SRST) gateway after the `debug cch323 video` command was enabled:

Router# show log
Syslog logging: enabled (11 messages dropped, 466 messages rate-limited, 0 flushes, 0 overruns, xml disabled, filtering disabled)
Console logging: disabled
Monitor logging: level debugging, 0 messages logged, xml disabled, filtering disabled
Buffer logging: level debugging, 829 messages logged, xml disabled, filtering disabled
Logging Exception size (4096 bytes)
Count and timestamp logging messages: disabled
Trap logging: level informational, 771 message lines logged
Log Buffer (200000 bytes):
Jun 13 09:19:42.011: //103034/C7838B198002/H323/setup_ind: Receive bearer cap info XRate 24, rateMult 12
Jun 13 09:19:42.011: //103034/C7838B198002/H323/cch323_set_h245_state_mc_mode_incoming: h245 state m/c mode=0x10F, h323_ctl=0x2F
Jun 13 09:19:42.015: //-1/xxxxxxxxxxxx/H323/cch245_event_handler: callID=103034
Jun 13 09:19:42.019: //-1/xxxxxxxxxxxx/H323/cch323_set_mode: callID=103034, flow Mode=1 spi_mode=0x6
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_do_call_proceeding: set_mode NOT called yet...saved deferred CALL_PROC
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_h245_connection_sm: state=0, event=0, ccb=4461B518, listen state=0
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_process_set_mode: Setting inbound leg mode flags to flush, 0=mode to FLOW THROUGH
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_process_set_mode: Sending deferred CALL_PROC
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_process_set_mode: Sending deferred CALL_PROC
Jun 13 09:19:42.019: //103034/C7838B198002/H323/cch323_send_cap_request: Setting mode to VIDEO MODE
Jun 13 09:19:42.031: //103034/C7838B198002/H323/cch323_h245_cap_ind: Masks au=0xC data=0x2 uinp=0x32

Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ephone video</td>
<td>Sets video debugging for the Cisco Unified IP phone.</td>
</tr>
<tr>
<td>show call active video</td>
<td>Displays call information for SCCP video calls in progress.</td>
</tr>
<tr>
<td>show call history video</td>
<td>Displays call history information for SCCP video calls.</td>
</tr>
<tr>
<td>show debugging</td>
<td>Displays information about the types of debugging that are enabled for your router.</td>
</tr>
</tbody>
</table>
**debug ccm-manager**

To display debugging information about Cisco CallManager, 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 {errors | events | packets} | config-download {all | errors | events | packets | tone | xml} | errors | events | music-on-hold {errors | events | packets} | packets}
no debug ccm-manager
```

### Syntax Description

| backhaul | Enables debugging of the Cisco CallManager backhaul. The keywords are as follows:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• <code>errors</code> -- Displays Cisco CallManager backhaul errors.</td>
<td></td>
</tr>
<tr>
<td>• <code>events</code> -- Displays Cisco CallManager backhaul events.</td>
<td></td>
</tr>
<tr>
<td>• <code>packets</code> -- Displays Cisco CallManager packets.</td>
<td></td>
</tr>
</tbody>
</table>

| config-download | Enables debugging of the Cisco CallManager configuration download. The keywords are as follows:
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• <code>all</code> -- Displays all Cisco CallManager configuration parameters.</td>
<td></td>
</tr>
<tr>
<td>• <code>errors</code> -- Displays Cisco CallManager configuration errors.</td>
<td></td>
</tr>
<tr>
<td>• <code>events</code> -- Displays Cisco CallManager configuration events.</td>
<td></td>
</tr>
<tr>
<td>• <code>packets</code> -- Displays Cisco CallManager configuration packets.</td>
<td></td>
</tr>
<tr>
<td>• <code>tone</code> -- Displays Cisco CallManager downloaded custom tones.</td>
<td></td>
</tr>
<tr>
<td>• <code>xml</code> -- Displays the Cisco CallManager configuration XML parser.</td>
<td></td>
</tr>
</tbody>
</table>

| errors | Displays errors related to Cisco CallManager. |

| events | Displays Cisco CallManager events, such as when the primary Cisco CallManager server fails and control is switched to the backup Cisco CallManager server. |
### music-on-hold

Displays music on hold (MOH). The keywords are as follows:

- **errors** -- Displays MOH errors.
- **events** -- Displays MOH events.
- **packets** -- Displays MOH packets.

<table>
<thead>
<tr>
<th>packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Displays Cisco CallManager packets.</td>
</tr>
</tbody>
</table>

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)T</td>
<td>This command was introduced for Cisco CallManager Version 3.0 and the Cisco VG200.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>This command was implemented on Cisco 2600 series and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.2(2)XN</td>
<td>Support for enhanced MGCP voice gateway interoperability was added to Cisco CallManager Version 3.1 for the Cisco 2600 series, Cisco 3600 series, and Cisco VG200.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T and implemented on the Cisco IAD2420 series.</td>
</tr>
<tr>
<td>12.2(15)XJ</td>
<td>The <strong>tone</strong> keyword was added for the following platforms: Cisco 2610XM, Cisco 611XM, Cisco 2620XM, Cisco 2621XM, Cisco 2650XM, Cisco 2651XM, Cisco 2691, Cisco 3640A, Cisco 3660, Cisco 3725, and Cisco 3745.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>The <strong>tone</strong> keyword was added.</td>
</tr>
<tr>
<td>12.3(14)T</td>
<td>New output was added relating to the SCCP protocol.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

### Examples

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

```plaintext
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)
```
You can use the `debug ccm-manager config-download tone` command to verify the parameters assigned to each locale. The following sample output shows the locale name United Kingdom and lists all the dual-tone parameters for that region:

```
Router# debug ccm-manager config-download tone
00:09:07: cmapp_prefix_process_tag_tones:
00:09:07: cmapp_process_tag_trkLocaleName: region = United Kingdom
00:09:07: cmapp_process_tag_pulse_ratio: pulse ratio = 40
00:09:07: cmapp_process_tag_dtmf_llevel: low frequency level = 65438
00:09:07: cmapp_process_tag_dtmf_hlevel: high frequency level = 65463
00:09:07: cmapp_process_tag_special_oper: operation = uLaw
00:09:07: cmapp_prefix_process_tag_lpig:
00:09:07: cmapp_process_tag_fxs: ignore LPIG for fxs
00:09:07: cmapp_process_tag_fxo: ignore LPIG for fxo
00:09:07: cmapp_process_tag_digital: ignore LPIG for digital
00:09:07: cmapp_prefix_process_tag_lpog:
00:09:07: cmapp_process_tag_fxs: ignore LPOG for fxs Both ports are in service
00:09:07: cmapp_process_tag_fxo: ignore LPOG for fxo
00:09:07: cmapp_process_tag_digital: ignore LPOG for digital
00:09:07: cmapp_prefix_process_tag_dualtone: TID=[0:CPTONE_BUSY]
00:09:07: cmapp_process_tag_nf: number of frequencies = 1
00:09:07: cmapp_process_tag_dr: direction = 0
00:09:07: cmapp_process_tag_fof: frequency 1 = 400
00:09:07: cmapp_process_tag_fot: frequency 3 = 0
00:09:07: cmapp_process_tag_fos: frequency 2 = 0
00:09:07: cmapp_process_tag_ontf: frequency 3 on time = 375
00:09:07: cmapp_process_tag_oftf: frequency 3 off time = 375
00:09:07: cmapp_process_tag_fof2: frequency 1 cadence 2 = 0
00:09:07: cmapp_process_tag_oftf: frequency 1 off time = 375
00:09:07: cmapp_process_tag_onts: frequency 1 on time = 0
00:09:07: cmapp_process_tag_ofts: frequency 1 off time = 0
00:09:07: cmapp_process_tag_ontt: frequency 3 on time = 0
00:09:07: cmapp_process_tag_oftt: frequency 3 off time = 0
```
00:09:07: cmapp_process_tag_fos3: frequency 2 cadence 3 = 0
00:09:07: cmapp_process_tag_fof4: frequency 1 cadence 4 = 0
00:09:07: cmapp_process_tag_fos4: frequency 2 cadence 4 = 0
00:09:07: cmapp_process_tag_rct1: cadence 1 repeat count = 0
00:09:07: cmapp_process_tag_rct2: cadence 2 repeat count = 0
00:09:07: cmapp_process_tag_rct3: cadence 3 repeat count = 0
00:09:07: cmapp_process_tag_rct4: cadence 4 repeat count = 0
00:09:07: cmapp_prefix_process_tag_dualtone: TID=[1:CPTONE_RING_BACK]
00:09:07: cmapp_process_tag_nf: number of frequencies = 2
00:09:07: cmapp_process_tag_dr: direction = 0
00:09:07: cmapp_process_tag_fof: frequency 1 = 400
00:09:07: cmapp_process_tag_fos: frequency 2 = 450
00:09:07: cmapp_process_tag_fot: frequency 3 = 0
00:09:07: cmapp_process_tag_fo4: frequency 4 = 0
00:09:07: cmapp_prefix_process_tag_aof_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 1st = -190
00:09:07: cmapp_process_tag_fxo: amplitude of 1st = -190
00:09:07: cmapp_process_tag_digital: amplitude of 1st = -190
00:09:07: cmapp_prefix_process_tag_aos_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 2nd = -190
00:09:07: cmapp_process_tag_fxo: amplitude of 2nd = -190
00:09:07: cmapp_process_tag_digital: amplitude of 2nd = -190
00:09:07: cmapp_prefix_process_tag_aot_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_digital: amplitude of 3rd = 0
00:09:07: cmapp_prefix_process_tag_ao4_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 4th = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 4th = 0
00:09:07: cmapp_process_tag_digital: amplitude of 4th = 0
00:09:07: cmapp_process_tag_ontf: frequency 1 on time = 400
00:09:07: cmapp_process_tag_oftf: frequency 1 off time = 200
00:09:07: cmapp_process_tag_onts: frequency 2 on time = 400
00:09:07: cmapp_process_tag_ofts: frequency 2 off time = 2000
00:09:07: cmapp_process_tag_ontt: frequency 3 on time = 0
00:09:07: cmapp_process_tag_oftt: frequency 3 off time = 0
00:09:07: cmapp_process_tag_ont4: frequency 4 on time = 0
00:09:07: cmapp_process_tag_oft4: frequency 4 off time = 0
00:09:07: cmapp_process_tag_fof2: frequency 1 cadence 2 = 0
00:09:07: cmapp_process_tag_fos2: frequency 2 cadence 2 = 0
00:09:07: cmapp_process_tag_fof3: frequency 1 cadence 3 = 0
00:09:07: cmapp_process_tag_fos3: frequency 2 cadence 3 = 0
00:09:07: cmapp_process_tag_fof4: frequency 1 cadence 4 = 0
00:09:07: cmapp_process_tag_fos4: frequency 2 cadence 4 = 0
00:09:07: cmapp_process_tag_rct1: cadence 1 repeat count = 0
00:09:07: cmapp_process_tag_rct2: cadence 2 repeat count = 0
00:09:07: cmapp_process_tag_rct3: cadence 3 repeat count = 0
00:09:07: cmapp_process_tag_rct4: cadence 4 repeat count = 0
00:09:07: cmapp_prefix_process_tag_dualtone: TID=[2:CPTONE_CONGESTION]
00:09:07: cmapp_process_tag_nf: number of frequencies = 1
00:09:07: cmapp_process_tag_dr: direction = 0
00:09:07: cmapp_process_tag_fof: frequency 1 = 400
00:09:07: cmapp_process_tag_fos: frequency 2 = 0
00:09:07: cmapp_process_tag_fot: frequency 3 = 0
00:09:07: cmapp_process_tag_fo4: frequency 4 = 0
00:09:07: cmapp_prefix_process_tag_aof_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 1st = -200
00:09:07: cmapp_process_tag_fxo: amplitude of 1st = -200
00:09:07: cmapp_process_tag_digital: amplitude of 1st = -200
00:09:07: cmapp_prefix_process_tag_aos_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 2nd = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 2nd = 0
00:09:07: cmapp_process_tag_digital: amplitude of 2nd = 0
00:09:07: cmapp_prefix_process_tag_aot_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 3rd = 0
00:09:07: cmapp_process_tag_digital: amplitude of 3rd = 0
00:09:07: cmapp_prefix_process_tag_ao4_level:
00:09:07: cmapp_process_tag_fxs: amplitude of 4th = 0
00:09:07: cmapp_process_tag_fxo: amplitude of 4th = 0
00:09:07: cmapp_process_tag_digital: amplitude of 4th = 0
The following is sample output from the `debug ccm-manager config-download all` command for an error case in which the configuration file cannot be accessed for a Skinny Client Control Protocol (SCCP) download:

```
*Jan 9 07:28:33.499: cmapp_xml_process_Timer:
*Jan 9 07:28:33.499: cmapp_xml_find_ep_by_name: Checking for ep_name [*]
*Jan 9 07:28:33.499: cmapp_xml_exec_fsm: Endpoint is [*]
*Jan 9 07:28:33.499: cmapp_xml_exec_fsm: endpoint = * state = CMAPP_XML_FILE_DNLD, event
  = CMAPP_XML_EVT_FILE_DNLD TIMER
  = CMAPP_XML_EVT_FILE_DNLD TIMER
*Jan 9 07:29:14.499: cmapp_xml_tftp_download_file: Unable to read file
tftp://10.6.6.31/Router.cisco.com.cnf.xml, rc=-2
tftp://10.6.6.31/Router.cisco.com.cnf.xml, len = 0
*Jan 9 07:29:14.499: cmapp_xml_tftp_download_file: Unable to read file
tftp://10.6.6.31/Router.cisco.com.cnf.xml, rc=-2
tftp://10.6.6.31/Router.cisco.com.cnf.xml, len = 0
*Jan 9 07:29:14.499: cmapp_xml_tftp_download_file: Unable to read file
tftp://10.6.6.31/Router.cisco.com.cnf.xml, rc=-2
tftp://10.6.6.31/Router.cisco.com.cnf.xml, len = 0
```

The following is sample output from the `debug ccm-manager config-download all` command for a successful SCCP download:

```
*Jan 9 09:44:45.543: cmapp_sccp_config:
*Jan 9 09:44:45.543: cmapp_sccp_reset_curcfg:
*Jan 9 09:44:45.543: cmapp_sccp_init_curcfg:
*Jan 9 09:44:45.543: cmapp_sccp_download_gw_config_file:
*Jan 9 09:44:45.543: cmapp_sccp_get_gw_name:
*Jan 9 09:44:45.543: cmapp_sccp_get_gw_name: XML file name generated->SKIGW0C85226910.cnf.xml
*Jan 9 09:44:45.543: cmapp_sccp_get_xml_file_via_tftp:
*Jan 9 09:44:45.543: cmapp_sccp_tftp_download_file:
*Jan 9 09:44:45.543: cmapp_sccp_tftp_get_file_size:
*Jan 9 09:44:45.563: cmapp_sccp_get_buffer:
*Jan 9 09:44:45.575: cmapp_sccp_tftp_download_file: File
tftp://10.2.6.101/SKIGW0C85226910.cnf.xml read 8162 bytes
*Jan 9 09:44:45.575: cmapp_sccp_get_xml_file_via_tftp: Read file
tftp://10.2.6.101/SKIGW0C85226910.cnf.xml, len = 8162
*Jan 9 09:44:45.575: cmapp_parse_gw_xml_file:
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: ccm found, priority=0
```

The following lines show the conversion of XML data into router configuration information for the endpoint:

```
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Unit has been set to 1
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Subunit has been set to 0
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 0
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 1
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 2
*Jan 9 09:44:45.579: cmapp_sccp_gw_start_element_handler: Endpoint has been set to 3
```
The table below describes the significant fields shown in the displays.

**Table 16: debug ccm-manager Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>nn :nn :nn :</strong></td>
<td>Timestamp time in hours (military format), minutes, and seconds that indicates when the Cisco CallManager event occurred.</td>
</tr>
<tr>
<td><code>cmapp_ error message:</code></td>
<td>The Cisco CallManager routine in which the error event occurred.</td>
</tr>
<tr>
<td><code>LocaleName</code></td>
<td>Region name, such as United Kingdom.</td>
</tr>
<tr>
<td>low frequency level</td>
<td>DTMF low frequency.</td>
</tr>
<tr>
<td>high frequency level</td>
<td>DTMF high frequency.</td>
</tr>
<tr>
<td>operation</td>
<td>Special operations, such as uLaw.</td>
</tr>
</tbody>
</table>

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show ccm-manager</code></td>
<td>Displays a list of Cisco CallManager servers, their current status, and their availability.</td>
</tr>
</tbody>
</table>
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.

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

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

**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: 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: act_idle_connection_created
* Mar 6 14:10:42: act_idle_connection_created: Connid(1) created to 166.34.245.231:5060, local_port 54113
* Mar 6 14:10:42: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
* Mar 6 14:10:42: act_sentinvite_new_message
* 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_RECV_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
* Mar 6 14:10:42: 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=Call
t=0 0
c=IN IP4 166.34.245.230
m=audio 20208 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: 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
Cisco IOS Debug Command Reference - Commands A through D

debug ccsip all

debug backhaul-session-manager session through debug channel packets

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: Roundtrip delay 8 milliseconds for method INVITE
*Mar 6 14:10:46: HandleSIPmxxRinging: SDP MediaTypes negotiation successful!
Negotiated Codec : g711ulaw , bytes : 160
Inband Alerting : 0

*Mar 6 14:10:46: 0x624CFEF8 : State change from (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING) to (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
Negotiated Codec : g711ulaw , bytes : 160

*Mar 6 14:10:46: CCSIP-SPICONTROL: sipSPIReconnectConnection
*Mar 6 14:10:46: Queued event from SIP SPI : SIP_SPI_EV_RECONNECT_CONNECTION
*Mar 6 14:10:46: Queued event from SIP SPI : SIP_SPI_EV_SEND_MESSAGE
*Mar 6 14:10:46: 0x624CFEF8 : State change from (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING) to (STATE_ACTIVE, SUBSTATE_NONE)
*Mar 6 14:10:46: The Call Setup Information is :
  Call Control Block (CCB) : 0x624CFEF8
  State of The Call : STATE_ACTIVE
  TCP Sockets Used : NO
  Calling Number : 3660110
  Called Number : 3660210
  Negotiated Codec : g711ulaw
  Source IP Address (Media) : 166.34.245.230
  Source IP Port (Media) : 20028
  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_caps_ind: Load DSP with codec (5) g711ulaw, Bytes=160
*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: act_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 : SIP_SPI_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
CSeq: 101 BYE
Content-Length: 0

*Mar 6 14:10:50: Queued event from SIP SPI to CCAPI/DNS : SIP_SPI_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 : SIP_SPI_EV_CLOSE_CONNECTION
*Mar 6 14:10:50: CLOSE CONNECTION TO CONNID:1
*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) : 20039
  Destn SIP Addr (Control) : 166.34.245.231
  Destn SIP Port (Control) : 5060
  Destination Name : 166.34.245.231
The following example displays debug output from the other side of the call:

Router# debug ccsip all
All SIP call tracing enabled
3660-2 4
*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>
Call-ID: ABBAE7AF-8231000CE-0-1CCAA69C0172.18.192.194
Cisco-Guid: 2881152943-2184249548-0-443049712
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Max-Forwards: 6
Timestamp: 731427042
Contact: <sip:36601100166.34.245.230:5060;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: Queue event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*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
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0
*Mar 8 17:36:40: Queue event from SIP SPI : SIPSPI_EV_CC_CALL_PROCEEDING
*Mar 8 17:36:40: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_ALERTING
*Mar 8 17:36:40: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_PROCCEEDING
*Mar 8 17:36:40: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_ALERTING
*Mar 8 17:36:40: 180 Ringing with SDP - not likely
*Mar 8 17:36:40: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
*Mar 8 17:36:40: CCSIP-SPI-CONTROL: sip_stats_status_code
*Mar 8 17:36:40: 0x624D8CCC : State change from (STATE_RECD_INVITE, SUBSTATE_RECD_INVITE_CALL_SETUP) to (STATE_SENT_ALERTING, SUBSTATE_NONE)
*Mar 8 17:36:40: Sent:
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 166.34.245.230:54113
From: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660210@166.34.245.231;user=phone;phone-context=unknown>
Date: Mon, 08 Mar 1993 22:36:40 GMT
Call-ID: ABBAE7AF-823100CE-0-1CCAA69C@172.18.192.194
Timestamp: 731427042
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
Content-Length: 137
v=0
o=CiscoSystemsSIP-GW-UserAgent 969 7889 IN IP4 166.34.245.231
a=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
a=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
a=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: act_sentsucc_new_message
*Mar 8 17:36:44: CCSIP-SPI-CONTROL: act_sentsucc_new_message
*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

Cisco IOS Debug Command Reference - Commands A through D
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: 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-0-1CCAA69C0172.18.192.194
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-0-1CCAA69C0172.18.192.194
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Max-Forwards: 6
Timestamp: 731612207
CSeq: 101 BYE
Content-Length: 0
*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: sIP2IPCheckResponse
*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_DISCONNECT
*Mar 8 17:36:47: CLOSE CONNECTION TO CONNID:1
*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
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
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.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

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

**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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
debug ccsip dhcp

To display debugging related information on Session Initiation Protocol (SIP) and Dynamic Host Configuration Protocol (DHCP) interaction, when SIP parameters are provisioned by DHCP, use the `debug ccsip dhcp` command in privileged EXEC mode. To disable debugging output, use the `no debug ccsip dhcp` form of this command.

### Syntax Description

This command has no arguments or keywords.

### Command Modes

Privileged EXEC (#)

### Command History

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(22)YB</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>15.0(1)M</td>
<td>This command was integrated in Cisco IOS Release 15.0(1)M.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

The `debug ccsip dhcp` command can be enabled by executing the command itself or by issuing the `debug ccsip all` command.

### Examples

The following example displays debug output from the `debug ccsip dhcp` command:

```
Router# debug ccsip dhcp
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/config_credential_trigger_reg: Query DHCP for provisioned info upon credential dhcp config
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/sipua_query_dhcp_reg_info: DHCP provisioned option 125 available
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing data in option 125 of length 73
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: enterprise ID 210
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: total option data length 80
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 201 of length 6
Nov 18 17:21:00.965: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 202 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_macaddr: MAC addr 1234567890AB
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_contract_num: pilot # 777777
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 203 of length 6
Nov 18 17:21:00.969: //1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num: secondary # 222222 (index 0)
```
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 203 of length 6
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num: secondary # 333333 (index 1)
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 203 of length 6
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num: secondary # 444444 (index 2)
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 203 of length 6
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num: secondary # 555555 (index 3)
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: sub-option type 203 of length 6
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_subopt_addn_num: secondary # 666666 (index 4)
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/SIP-DHCP/ccsip_gw_parse_dhcp_opt125: parsing of DHCP option 125 succeeded
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/Info/ccsip_register_cred_user: Sending msg type 2 to register process for user 777777
Nov 18 17:21:00.969: //-1/xxxxxxxxxxxx/SIP/Info/ccsip_register_add_to_e164_table: ****Added to E164 Table
Nov 18 17:21:00.969: //1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (STATE_IDLE, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_SENT_DNS)
Nov 18 17:21:00.969: //1034/000000000000/SIP/Info/sip_dns_type_srv_query: TYPE SRV query for _sip._udp.cisco.com and type:1
Nov 18 17:21:00.977: //1034/000000000000/SIP/Info/sip_dns_type_a_aaaa_query: DNS query for cisco.com and type:1
Nov 18 17:21:00.977: //1034/000000000000/SIP/Info/sip_dns_type_a_query: TYPE A query successful for cisco.com
Nov 18 17:21:00.977: //1034/000000000000/SIP/Info/sip_dns_type_a_aaaa_query: IP Address of cisco.com is:
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/sip_dns_type_a_aaaa_query: 9.13.2.36
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsip_process_sipspi_queue_event: ccsip_api_get_msg_type returned: 2 for event 43
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsip_register_set_dns_resolved_address: CCSIP_REGISTER:: Primary registrar DNS resolved addr set to 0.0.0.1:15187567
Nov 18 17:21:00.977: //1/xxxxxxxxxxxx/SIP/Info/ccsipRegisterStartExpiresTimer: Starting timer for pattern for 3600 seconds
Nov 18 17:21:00.977: //1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_SENT_DNS) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:00.977: //1034/000000000000/SIP/Info/ccsipSetDateHeader: Clock Time Zone is UTC, same as GMT. Using GMT
Nov 18 17:21:00.981: //1034/000000000000/SIP/Info/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:00.981://1034/000000000000/SIP/Transport/sipTransportLogicSendMsg: switch transport is 0
Nov 18 17:21:00.981://1034/000000000000/SIP/Transport/sipTransportLogicSendMsg: Set to send the msg-0x4707F998
Nov 18 17:21:00.981://1034/000000000000/SIP/Transport/sipTransportPostSendMessage: Proceedable for sending msg immediately
Nov 18 17:21:00.981://1034/000000000000/SIP/State/sipSPIChangeState: 0x476FD758 : State change from (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE) to (SIP_STATE_OUTGOING_REGISTER, SUBSTATE_NONE)
Nov 18 17:21:00.981://1034/000000000000/SIP/Msg/ccsipDisplayMsg: Sent:
REGISTER sip:cisco.com:5060 SIP/2.0
Date: Tue, 18 Nov 2008 17:21:00 GMT
From: <sip:777777@cisco.com>;tag=34FBAED8-131
Supported: path
Supported: backhaul-session-manager session through debug channel packets
timestamp
User-Agent: Cisco-SIPGateway/IOS-12.x
To: <sip:777777@cisco.com>
Contact: <sip:7777779@9.13.8.183:5060>
Expires: 3600
Call-ID: 1AF6E28A-B4CC11DD-81078B9C-6E99E02B
Via: SIP/2.0/UDP 9.13.8.183:5060;branch=z9hG4bK3F522D9
CSeq: 2 REGISTER
Max-Forwards: 70
Nov 18 17:21:00.995: //1/xxxxxxxxxxxx/SIP/Info/ccsip_new_msg_preprocessor: Checking Invite Dialog
Nov 18 17:21:00.995: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg: Received:
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 9.13.8.183:5060;received=9.13.8.183;branch=z9hG4bK3F522D9
CSeq: 2 REGISTER
Content-Length: 0
}
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/Msg/ccsipDisplayMsg: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.13.8.183:5060;received=9.13.8.183;branch=z9hG4bK3F522D9
Call-ID: 1AF6E28A-B4CC11DD-81078B9C-6E99E02B
From: <sip:777777@cisco.com>;tag=34FBAED8-131
To: <sip:777777@cisco.com>
CSeq: 2 REGISTER
Contact: <sip:777777@9.13.8.183:5060>;expires=3600
Content-Length: 0
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/Info/ccsipRegisterStartExpiresTimer: Starting
timer for pattern 777777 for 2880 seconds
Nov 18 17:21:01.077: //1/xxxxxxxxxxxx/SIP/Info/sipSPIDeleteContextFromTable: Context for
deprecated key=[1061] removed.
Nov 18 17:21:01.077: //1034/000000000000/SIP/Info/sipSPIUdeleteCcbFromTable: Deleting from
table. ccb=0x476FD758 key=1AF6E28A-B4CC11DD-81078B9C-6E99E02B
Nov 18 17:21:01.077: //1034/000000000000/SIP/Info/sipSPIFlushEventBufferQueue: There are 0 events on the internal queue that are going to be free'd
Nov 18 17:21:01.077: //1034/000000000000/SIP/Info/ccsip_qos_cleanup: Entry
Nov 18 17:21:01.077: //1034/000000000000/SIP/Info/sipSPI_ipip_free_codec_profile: Codec Profiles Freed
Nov 18 17:21:02.761: %SYS-5-CONFIG_I: Configured from console by console
CUBE-DHCP-CLIENT1#

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Displays all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Displays SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Displays all SIP events tracing.</td>
</tr>
<tr>
<td>debug ccsip in</td>
<td>Displays all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Displays all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
debug ccsip error

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

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

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.

12.2(11)T This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.

**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 ccsip error
SIP Call error tracing is enabled
Router1#
*Mar 6 14:16:41: act_idle_call_setup:Not using Voice Class Codec
*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: HandleUdpSocketReads:Msg enqueued for SPI with IPaddr: 166.34.245.231:5060
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: sactIdleNewMessageInvite: Media Negotiation successful for an incoming call
  Preferred Codec : g711ulaw, bytes :160
* Mar 8 17:42:39: udpsock_close_connect: Socket fd: 1 closed for connid 1 with remote port: 5060
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
debug ccsip events

To enable tracing of events that are specific to service provider interface (SPI), use the `debug ccsip events` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
ddebug ccsip events
no debug ccsip events
```

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

**Command Default**
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 introduced on the Cisco AS5850 universal gateway.
- **12.2(11)T**: This command was integrated into Cisco IOS Release 12.2(11)T.
- **12.2(15)T**: Much of the information formerly found in the output of the `debug ccsip events` command is now reported in the output of the `debug ccsip info` and `debug ccsip media` commands. The `debug ccsip events` command now displays only the debugging information specifically related to SIP events.

**Usage Guidelines**
This command previously traced all events posted to Session Initiation Protocol (SIP) SPI from all interfaces and also provided general SIP SPI information. Beginning with Cisco IOS Release 12.2(15)T, the `debug ccsip events` command displays only debugging information specifically related to SIP SPI events. Media stream and SIP SPI information is now reported in the `debug ccsip media` and `debug ccsip info` command output.

**Note**
This command is intended for use by Cisco technicians only.

**Examples**
The following is sample output from the `debug ccsip events` command for a Cisco 3660:

```
Router# debug ccsip events
SIP Call events tracing is enabled
```
Router#
Nov 15 18:20:25.779: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_SETUP
Nov 15 18:20:25.779: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Nov 15 18:20:25.783: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:25.815: Queued event from SIP SPI : SIPSPI_EV_CREATE_CONNECTION
Nov 15 18:20:25.819: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:50.844: Queued event from SIP SPI : SIPSPI_EV_CLOSE_CONNECTION
Nov 15 18:20:50.844: Queued event from SIP SPI : SIPSPI_EV_SEND_MESSAGE
Nov 15 18:20:50.848: Queued event from SIP SPI : SIPSPI_EV_CC_CALL_DISCONNECT

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ccsip all</code></td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td><code>debug ccsip info</code></td>
<td>Enables tracing of general SIP SPI information.</td>
</tr>
<tr>
<td><code>debug ccsip media</code></td>
<td>Enables tracing of SIP call media streams.</td>
</tr>
</tbody>
</table>
debug ccsip info

To enable tracing of general service provider interface (SPI) information, use the `debug ccsip info` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccsip info
no debug ccsip info
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(15)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Beginning in Cisco IOS Release 12.2(15)T, the `debug ccsip info` command is a separate option that displays general SIP SPI information for debug purposes. In past releases, this output was part of the `debug ccsip events` command.

**Note**

This command is intended for use by Cisco technicians only.

**Examples**

The following is sample output from the `debug ccsip info` command for a Cisco 3660:

```
Router# debug ccsip info
SIP Call info tracing is enabled
Router# Nov 15 18:19:22.670: ****Adding to UAC table
Nov 15 18:19:22.670: adding call id E to table
Nov 15 18:19:22.670: sipSPICopyPeerDataToCCB: From CLI: Modem NSE payload = 100, Passthrough = 0, Modem relay = 0, Gw-Xid = 1
SPRT latency 200, SPRT Retries = 12, Dict Size = 1024
String Len = 32, Compress dir = 3
Nov 15 18:19:22.670: ****Deleting from UAC table
Nov 15 18:19:22.670: ****Adding to UAC table
Nov 15 18:19:22.670: sipSIPSetBillingProfile: sipCallld for billing records = 20A40C3B-D92C11D5-8015E1CC-C91F3F10@12.18.195.49
Nov 15 18:19:22.674: CCSIP-SPI-CONTROL: sipSPOutgoingCallSDP
Nov 15 18:19:22.674: convert_codec_bytes_to_ptime: Values :Codec: g729r8 codecbytes :20, ptime: 10
```
Nov 15 18:19:22.674: sipSPIAddLocalContact
Nov 15 18:19:22.674: sip_stats_method
Nov 15 18:19:22.690: sip_stats_status_code
Nov 15 18:19:22.690: Roundtrip delay 16 milliseconds for method INVITE
Nov 15 18:19:22.706: sip_stats_status_code
Nov 15 18:19:22.706: Roundtrip delay 32 milliseconds for method INVITE
Nov 15 18:19:22.706: sip_stats_status_code
Nov 15 18:19:22.706: Roundtrip delay 48 milliseconds for method PRACK
Nov 15 18:19:22.706: sip_stats_status_code
Nov 15 18:19:22.706: Roundtrip delay 48 milliseconds for method PRACK
Nov 15 18:19:22.706: CCSIP-SPI-CONTROL: sipSPICheckSocketConnection: Connid(2) created to 172.18.193.190:5060, local_port 51663
Nov 15 18:19:22.706: sip_stats_method
Nov 15 18:19:22.722: sip_stats_status_code
Nov 15 18:19:22.722: Roundtrip delay 48 milliseconds for method PRACK
Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Enables tracing of events that are specific to SIP SPI.</td>
</tr>
<tr>
<td>debug ccsip media</td>
<td>Enables tracing of SIP call media streams.</td>
</tr>
</tbody>
</table>
**debug ccsip media**

To enable tracing of Session Initiation Protocol (SIP) call media streams, use the `debug ccsip media` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(15)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Beginning in Cisco IOS Release 12.2(15)T, the `debug ccsip media` command is a separate option that displays debugging information specific to SIP media stream processing. In past releases, this output was part of the `debug ccsip events` command.

**Note**

This command is intended for use by Cisco technicians only.

**Examples**

The following is sample output from the `debug ccsip media` command for a Cisco 3660:

```
Router# debug ccsip media
SIP Call media tracing is enabled
Router#
Nov 15 18:19:53.835: sipSPIReserveRtpPort: reserved port 16500 for stream 1
Nov 15 18:19:53.867: sipSPIReplaceSDP
Nov 15 18:19:53.871: sipSPICopySdpInfo
Nov 15 18:19:53.871: sipSPIUpdCallWithSdpInfo:
Preferred Codec : g729r8, bytes :20
Preferred DTMF relay : inband-voice
Preferred NTE payload : 101
Early Media : No
Delayed Media : No
Bridge Done : No
New Media : No
DSP DNLD Req'd : No
Nov 15 18:19:53.871: sipSPIUpdCallWithSdpInfo:
M-line Index : 1
State : STREAM_ADDING (3)
Callid : -1
Negotiated Codec : g729r8, bytes :20
```
$\textbf{Negotiated DTMF relay : inband-voice}$

$\textbf{Negotiated NTE payload : 0}$

$\textbf{Media Srcce Addr/Port : 172.18.195.49:16500}$

$\textbf{Media Dest Addr/Port : 172.18.193.190:19148}$

Nov 15 18:19:53.871: $\text{sipSPIProcessRtpSessions}$

Nov 15 18:19:53.871: $\text{sipSPIAddStream: Adding stream 1 (callid 16) to the VOIP RTP library}$

Nov 15 18:19:53.871: $\text{sipSPISetMediaSrcAddr: media src addr for stream 1 = 172.18.195.49}$

Nov 15 18:19:53.871: $\text{sipSPIUpdateRtcpSession: for m-line 1}$

Nov 15 18:19:53.871: $\text{sipSPIUpdateRtcpSession: rtcp_session info}$

$\text{laddr = 172.18.195.49, lport = 16500, raddr = 172.18.193.190, rport=19148}$

Nov 15 18:19:53.871: $\text{sipSPIUpdateRtcpSession: No rtp session, creating a new one}$

Nov 15 18:19:53.871: $\text{sipSPIUpdateRtcpSession: num_streams = 1}$

Nov 15 18:19:53.871: $\text{sipSPIUpdateRtcpSession: adding stream type 0 from mline 1}$

Nov 15 18:19:53.871: $\text{sipSPIUpdateRtcpSession: caps.stream_count=1, caps.stream[0].stream_type=0x1,}$

$\text{caps.stream_list.xmitFunc=voip_rtp_xmit, caps.stream_list.context=0x634F1F2C (gccb)}$

Nov 15 18:19:55.555: $\text{sipSPICompareSDP}$

Nov 15 18:19:55.555: $\text{sipSPICompareStreams: stream 1 dest_port: old=19148 new=19148}$

Nov 15 18:19:55.555: $\text{sipSPICompareStreams: Flags set for stream 1: RTP\_CHANGE=No}$

$\text{CAPS\_CHANGE=No}$

Nov 15 18:19:55.555: $\text{sipSPICompareSDP: Flags set for call: NEW\_MEDIA=No DSPDNLD\_REQD=No}$

Nov 15 18:19:55.555: $\text{sipSPIReplaceSDP}$

Nov 15 18:19:55.555: $\text{sipSPICopySdpInfo}$

Nov 15 18:19:55.555: $\text{sipSPIUpdCallWithSdpInfo:}$

Preferred Codec : g729r8, bytes :20

Preferred DTMF relay : inband-voice

Preferred NTE payload : 101

Early Media : No

Delayed Media : No

Bridge Done : Yes

New Media : No

DSP DNLD Req’d : No

Nov 15 18:19:55.555: $\text{sipSPISetMediaSrcAddr: media src addr for stream 1 = 172.18.195.49}$

Nov 15 18:19:55.555: $\text{sipSPIUpdCallWithSdpInfo:}$

M-line Index : 1

State : STREAM\_ACTIVE (3)

Callid : 16

Negotiated Codec : g729r8, bytes :20

Negotiated DTMF relay : inband-voice

Negotiated NTE payload : 0

Media Srcce Addr/Port : 172.18.195.49:16500

Media Dest Addr/Port : 172.18.193.190:19148

---

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug csip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug csip events</td>
<td>Enables tracing of events that are specific to SIP SPI.</td>
</tr>
<tr>
<td>debug csip info</td>
<td>Enables tracing of general SIP SPI events.</td>
</tr>
</tbody>
</table>
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.

```
direct ccsip messages
no debug ccsip messages
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(3)T</td>
<td>The output of this command was changed.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
<tr>
<td>IOS Release XE 2.5</td>
<td>This command was integrated into Cisco IOS XE Release 2.5.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
This command traces the Session Initiation Protocol (SIP) messages exchanged between the SIP UA client (UAC) and the access server.
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:3660110@166.34.245.230;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-823100EZ-0-1CD274BC@172.18.192.194
Cisco-Guid: 2887152943-2184249568-0-483551624
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=7UserAgent 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-823100EZ-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-823100EZ-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=7UserAgent 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-823100EZ-0-1CD274BC@172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Type: application/sdp
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@166.34.245.230;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 SIP/2.0
Max-Forwards: 6
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731427554
CSeq: 101 INVITE
Content-Length: 0

CSeq: 101 BYE

Content-Length: 0
*Mar 8 17:45:19: Sent:
BYE sip:3660110@166.34.245.230;user=phone;phone-context=unknown SIP/2.0
Via: SIP/2.0/UDP 166.34.245.230:58520
From: <sip:3660110@166.34.245.230>
To: sip:3660210@166.34.245.231;user=phone;phone-context=unknown SIP/2.0
Max-Forwards: 6
User-Agent: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Timestamp: 731427554
CSeq: 101 BYE
Content-Length: 0

The following examples show debug output from the other side of the call:
debug ccsip messages
c-IN IP4 166.34.245.230
m=audio 20762 RTP/AVP 0
*M-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-823100E2-0-1CD274BC0172.18.192.194
Timestamp: 731427554
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
CSeq: 101 INVITE
Content-Length: 0
*M-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-823100E2-0-1CD274BC0172.18.192.194
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
*M-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=27DB6D8-1357
Date: Mon, 08 Mar 1993 22:45:12 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC0172.18.192.194
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
*M-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=27DB6D8-1357
Date: Sat, 06 Mar 1993 19:19:14 GMT
Call-ID: ABBAE7AF-823100E2-0-1CD274BC0172.18.192.194
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
*M-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: "3660110" <sip:3660110@166.34.245.230>
To: <sip:3660110@166.34.245.231;user=phone;phone-context=unknown>;tag=27DB6D8-1357
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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip states</td>
<td>Shows all SIP SPI state tracing.</td>
</tr>
</tbody>
</table>
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 debugging output, use the no form of this command.

debug ccsip preauth
no debug ccsip preauth

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(11)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Examples**

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

```sh
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: 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
```

The table below describes the significant fields shown in the display.
**Table 17: debug ccsip preauth Field Descriptions**

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

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(1)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(2)XA</td>
<td>Support was added for the Cisco AS5350 and Cisco AS5400 universal gateways.</td>
</tr>
<tr>
<td>12.2(2)XB1</td>
<td>This command was implemented on the Cisco AS5850 universal gateway.</td>
</tr>
<tr>
<td>12.2(8)T</td>
<td>This command was implemented on Cisco 7200 series routers.</td>
</tr>
<tr>
<td>12.2(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(11)T. Support for the Cisco AS5300 universal access server, Cisco AS5350, Cisco AS5400, and Cisco AS5850 universal gateway is not included in this release.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**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_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING)
*Jan 2 18:34:37.853:0x6220C634 :State change from (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_PROCEEDING) to (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING)
*Jan 2 18:34:38.261:0x6220C634 :State change from (STATE_RECD_PROCEEDING, SUBSTATE_PROCEEDING_ALERTING) to (STATE_ACTIVE, SUBSTATE_NONE)
*Jan 2 18:35:09.860:0x6220C634 :State change from (STATE_ACTIVE, SUBSTATE_NONE) to (STATE_DISCONNECTING, SUBSTATE_NONE)
*Jan 2 18:35:09.868:0x6220C634 :State change from (STATE_DISCONNECTING, SUBSTATE_NONE) to (STATE_DEAD, SUBSTATE_NONE)
*Jan 2 18:28:38.404: Queued event from SIP SPI :SIPSPI_EV_CLOSE_CONNECTION

<table>
<thead>
<tr>
<th>Related Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip calls</td>
<td>Shows all SIP SPI call tracing.</td>
</tr>
<tr>
<td>debug ccsip error</td>
<td>Shows SIP SPI errors.</td>
</tr>
<tr>
<td>debug ccsip events</td>
<td>Shows all SIP SPI events tracing.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Shows all SIP SPI message tracing.</td>
</tr>
</tbody>
</table>
**debug ccsip transport**

To enable tracing of the Session Initiation Protocol (SIP) transport handler and the TCP or User Datagram Protocol (UDP) process, use the `debug ccsip transport` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccsip transport
no debug ccsip transport
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)T</td>
<td>This command was introduced.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `debug ccsip transport` command to debug issues related to connection and transport usage and to see the flow of the messages being sent or received.

**Examples**

The following is sample output from the `debug ccsip transport` command for a Cisco 3660:

```
Router# debug ccsip transport
.
.
1w1d: //18/8E16980D800A/SIP/Transport/sipSPISendInvite: Sending Invite to the transport layer
1w1d: //18/8E16980D800A/SIP/Transport/sipSPIGetSwitchTransportFlag: Return the Global configuration, Switch Transport is TRUE
1w1d: //18/BE16980D800A/SIP/Transport/sipSPITransportSendMessage: msg=0x64082D50, addr=172.18.194.183, port=5060, sentBy_port=0, is_req=1, transport=1, switch=1, callBack=0x614FA8B58
1w1d: //18/BE16980D800A/SIP/Transport/sipSPITransportSendMessage: Proceedable for sending msg immediately
1w1d: //18/BE16980D800A/SIP/Transport/sipTransportLogicSendMsg: switch transport is 1
1w1d: //18/BE16980D800A/SIP/Transport/sipTransportGetInterfaceMtuSize: MTU size for remote address 172.18.194.183 is 500
1w1d: //18/BE16980D800A/SIP/Transport/sipTransportVerifyMsgForMTUThreshold: Interface MTU Size 500, Msg Size 1096
1w1d: //18/BE16980D800A/SIP/Transport/sipTransportLogicSendMsg: Switching msg=0x64082D50 transport UDP->TCP
1w1d: //18/BE16980D800A/SIP/Transport/sipTransportSetAgeingTimer: Aging timer initiated for holder=0x64084058,addr=172.18.194.183
1w1d: //18/BE16980D800A/SIP/Transport/sipCreateConnHolder: Created new holder=0x64084058, addr=172.18.194.183
```
The table below describes the significant fields shown in the display.

**Table 18: debug ccsip transport Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sending Invite to the transport layer</td>
<td>Indicates that the SIP signaling state machine has invoked transport layer operations such as transport arbitration logic and the connection management interface.</td>
</tr>
<tr>
<td>switch transport is 1</td>
<td>Indicates that the gateway has been provisioned to enable the transport switching functionality based on the message size. 1 is true and 0 is false.</td>
</tr>
<tr>
<td>MTU size for remote address</td>
<td>Indicates that the bound outgoing Ethernet interface that sends the message to the given remote address is configured for an MTU size of the indicated value.</td>
</tr>
<tr>
<td>Interface MTU Size 500, Msg Size 1096</td>
<td>Indicates that the size of the message is larger than the size of the MTU; thus transport switching (from UDP to TCP) should be enabled.</td>
</tr>
<tr>
<td>Switching msg=... transport UDP-&gt;TCP</td>
<td>Indicates that transport switching from UDP to TCP is occurring for the handled message because of the large size of the message.</td>
</tr>
<tr>
<td>Aging timer initiated for holder</td>
<td>Indicates that the connection algorithm is started; that is, the counter begins to age out the TCP or UDP connection if inactivity occurs.</td>
</tr>
<tr>
<td>Posting TCP conn create request</td>
<td>Indicates a request for a TCP connection from a lower TCP process.</td>
</tr>
</tbody>
</table>
Indicates all the transport related attributes that the SIP signaling state machine originally gives to the transport layer to send out the message. The attributes are:

- transport: 1 for UDP; 2 for TCP.
- switch (switching transport enabled or disabled for large messages): 1 for enabled; 0 for disabled.

Indicates that all transport and connection related operations are complete. The message is sent out on the network targeted to the given address, port, and transport.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ccsip all</td>
<td>Enables all SIP-related debugging.</td>
</tr>
<tr>
<td>debug ccsip info</td>
<td>Enables tracing of general SIP SPI information.</td>
</tr>
<tr>
<td>transport switch</td>
<td>Enables switching between UDP and TCP transport mechanisms globally for large SIP messages.</td>
</tr>
<tr>
<td>voice-class sip transport switch</td>
<td>Enables switching between UDP and TCP transport mechanisms for large SIP messages for a specific dial peer.</td>
</tr>
</tbody>
</table>
debug ccsvoice vo-debug

To display detailed debugging information related to ccsvoice function calls during call setup and teardown, use the `debug ccsvoice vo-debug` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug ccsvoice vo-debug
no debug ccsvoice vo-debug
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(1)MA</td>
<td>This command was introduced on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was implemented on the Cisco 3600 series router.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(2)T.</td>
</tr>
</tbody>
</table>

**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 ccsvoicevo-session` command, but includes additional debugging information relating to the calls.

**Examples**

The following shows sample output from the `debug ccsvoicevo-debug` command:

```
Router# debug ccsvoice vo-debug
2w2d: ccsvoice: callID 529927 pvcid -1 cid -1 state NULL event O/G SETUP
2w2d: ccsvoice_out_callinit_setup: callID 529927 using pvcid 1 cid 15
2w2d: ccsvoice: callID 529927 pvcid 1 cid 15 state O/G INIT event I/C PROC
2w2d: ccsvoice: callID 529927 pvcid 1 cid 15 state O/G PROC event I/C ALERTccfrf11_caps_ind: codec(preferred) = 1
2w2d: ccsvoice: callID 529927 pvcid 1 cid 15 state O/G ALERT event I/C CONN
2w2d: ccsvoice_bridge_drop: dropping bridge calls src 529927 dst 529926 pvcid 1 cid 15 state ACTIVE
2w2d: ccsvoice: callID 529927 pvcid 1 cid 15 state ACTIVE event O/G REL
2w2d: ccsvoice: callID 529927 pvcid 1 cid 15 state RELEASE event I/C RELCOMP
2w2d: ccswo_store_call_history_entry: cause=10 tcause=10 cause_text=normal call clearing.
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug ccsvoice vo-session</code></td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

depug ccswvoice vo-debug
**debug ccswvoice vofr-debug**

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

**debug ccswvoice vofr-debug**

**no debug ccswvoice vofr-debug**

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)XG</td>
<td>This command was introduced on the Cisco 2600 and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.0(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.0(4)T.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was implemented on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(2)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

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

**Examples**

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

```
Router# debug ccswvoice vofr-debug  
CALL TEARDOWN:
3640_vofr(config-voiceport)#  

*Mar 1 03:02:08.719:ccswvofr_bridge_drop:dropping bridge calls src 17 dst 16 dlci 100 cid 9 state ACTIVE  
*Mar 1 03:02:08.727:ccswvofr:callID 17 dlci 100 cid 9 state 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 CONNECTED  
ccfref11_caps_ind:codec(preferred) = 0
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug cch323</code></td>
<td>Displays the ccf11 function calls during call setup and teardown.</td>
</tr>
<tr>
<td><code>debug ccsw voice vo-debug</code></td>
<td>Displays the ccswvoice function calls during call setup and teardown.</td>
</tr>
<tr>
<td><code>debug vtsp session</code></td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
**debug ccsvoice vofr-session**

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

```
debug ccswvoice vofr-session
no debug ccswvoice vofr-session
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)XG</td>
<td>This command was introduced on the Cisco 2600 and Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.0(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.0(4)T.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was implemented on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(2)T.</td>
</tr>
</tbody>
</table>

**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 shows sample output from the `debug ccswvoice vofr-session` command:

```
Router# debug ccswvoice vofr-session
CALL TEARDOWN:
3640_vofr(config-voiceport)#
*Mar 1 02:58:13.203:ccswvofr:callID 14 dlci 100 cid 8 state ACTIVE event O/G REL
*Mar 1 02:58:13.215:ccswvofr:callID 14 dlci 100 cid 8 state RELEASE event I/C RELCOMP
3640_vofr(config-voiceport)#
CALL SETUP (outgoing):
*Mar 1 02:59:46.551:ccswvofr:callID 17 dlci -1 cid -1 state NULL event O/G SETUP
*Mar 1 02:59:46.559:ccswvofr:callID 17 dlci 100 cid 9 state O/G INIT event I/C PROC
*Mar 1 02:59:46.567:ccswvofr:callID 17 dlci 100 cid 9 state O/G PROC event I/C CONN
3640_vofr(config-voiceport)#
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cch323</td>
<td>Displays the ccfrrt11 function calls during call setup and teardown.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>debug call rsvp-sync events</code></td>
<td>Displays events that occur during RSVP setup.</td>
</tr>
<tr>
<td><code>debug vtsp session</code></td>
<td>Displays the first 10 bytes (including header) of selected VoFR subframes for the interface.</td>
</tr>
</tbody>
</table>
**debug ccswvoice vo-session**

To display the first 10 bytes (including header) of selected VoFR subframes for the interface, use the `debug ccswvoice vo-session` command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```
ddebug ccswvoice vo-session
no debug ccswvoice vo-session
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3(1)MA</td>
<td>This command was introduced on the Cisco MC3810 networking device.</td>
</tr>
<tr>
<td>12.0(7)XK</td>
<td>This command was implemented on the Cisco 3600 series router.</td>
</tr>
<tr>
<td>12.1(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.1(2)T.</td>
</tr>
</tbody>
</table>

**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 shows sample output from the `debug ccswvoice vo-session` command:

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>debug ccswvoice vo-debug</strong></td>
<td>Displays detailed debugging information related to ccswvoice function calls during call setup and teardown.</td>
</tr>
</tbody>
</table>
debug cdapi

To display information about the Call Distributor Application Programming Interface (CDAPI), use the debug cdapi command in privileged EXEC mode. To disable debugging output, use the no form of this command.

```
debug cdapi {detail| events}
no debug cdapi {detail| events}
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detail</td>
<td>Displays when applications register or become unregistered with CDAPI, when calls are added or deleted from the CDAPI routing table, and when CDAPI messages are created and freed.</td>
</tr>
<tr>
<td>events</td>
<td>Displays the events passing between CDAPI and an application or signalling stack.</td>
</tr>
</tbody>
</table>

### Command Default

Debugging output is disabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(6)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.1(5)XM2</td>
<td>This command was implemented on the Cisco AS5350 and Cisco AS5400.</td>
</tr>
<tr>
<td>12.3(2)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(2)T. This command was enhanced to show V.110 call types.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was enhanced to show V.120 call types.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

### Usage Guidelines

The **detail** keyword is useful for determining if messages are being lost (or not freed). It is also useful for determining the size of the raw messages passed between CDAPI and other applications to ensure that the correct number of bytes is being passed.

The **events** keyword 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.

The following bandwidths are supported:
The following Media Gateway Control Protocol (MGCP) packet received example shows V.110 call debugging output for the `debug cdapi detail` command. In this example, the modem is not yet in STEADY_STATE.

```
Router# debug cdapi detail
Sep 26 19:12:25.327: MGCP Packet received from 10.0.44.109:2427-
CRCX 6318 s7/ds1-0/24 MGCP 1.0
C:111
M: nas/data
L: b:64, nas/bt:v.110, nas/cdn:234567
R: nas/au, nas/ax,nas/of, nas/crq
X:101
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Router# 
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
Sep 26 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
```

The following partial example shows V.120 call debugging output for the `debug cdapi detail` command:

```
Router# debug cdapi detail
May 14 19:12:25.327: MGCP Packet received from 10.0.44.109:2427-
CRCX 6318 s7/ds1-0/24 MGCP 1.0
C:111
M: nas/data
L: b:64, nas/bt:v.120, nas/cdn:234567
R: nas/au, nas/ax,nas/of, nas/crq
X:101
May 14 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
May 14 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
May 14 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
May 14 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
May 14 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:959, Raw Length = 0
May 14 19:12:25.327: CDAPI: cdapi_create_msg(): CDAPI Pool Count:958, Raw Length = 0
```
The following MGCP packet received example shows V.120 call debugging output for the `debug cdapi events` command:

```
Router# debug cdapi events
Sep 26 19:14:39.027: MGCP Packet received from 10.0.44.109:2427-
C:123
Sep 26 19:14:39.027: Se7/0:23 CDAPI: TX -> CDAPI_MSG_CONNECT_IND to CSM call = 0x7017
Sep 26 19:14:39.027: From Appl/Stack = CSM
Sep 26 19:14:39.027: Call Type = MODEM
Sep 26 19:14:39.027: B Channel = 23
Sep 26 19:14:39.027: ds1Id = 0
Sep 26 19:14:39.027: RPA = 0
Sep 26 19:14:39.027: Cause = 0
Sep 26 19:14:39.027: ApplCause = 0
Sep 26 19:14:39.027: ApplSpecData = 0
Sep 26 19:14:39.027: Overlap = 0
Sep 26 19:14:39.031: MGCP Packet received from 10.0.44.109:2427-
DLCX 5623 s7/ds7-0/24 MGCP 1.0
```

```
Router# debug cdapi events
Sep 26 19:14:39.027: MGCP Packet received from 10.0.44.109:2427-
C:123
Sep 26 19:14:39.027: Se7/0:23 CDAPI: TX -> CDAPI_MSG_CONN_ACT_REQ to XCS call = 0x7017
Sep 26 19:14:39.027: From Appl/Stack = CSM
Sep 26 19:14:39.027: Call Type = V.120
Sep 26 19:14:39.027: B Channel = 23
Sep 26 19:14:39.027: ds1Id = 0
Sep 26 19:14:39.027: RPA = 0
Sep 26 19:14:39.027: Cause = 0
Sep 26 19:14:39.027: ApplCause = 0
Sep 26 19:14:39.027: ApplSpecData = 0
Sep 26 19:14:39.027: Overlap = 0
```

```
Router# debug cdapi events
Sep 26 19:14:48.959: MGCP Packet received from 10.0.44.109:2427-
DLCX 5623 s7/ds7-0/24 MGCP 1.0
Sep 26 19:14:48.963: Se7/0:23 CDAPI: TX -> CDAPI_MSG_DISCONNECT_IND to CSM call = 0x7017
Sep 26 19:14:48.963: From Appl/Stack = XCS
Sep 26 19:14:48.963: Call Type = V.120
Sep 26 19:14:48.963: B Channel = 23
```
The table below describes the significant fields shown in the displays.

### Table 19: debug cdapi Field Descriptions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L:b:64, nas/bt</td>
<td>The bearer type parameter includes v.110 and v.120 for V.110 and V.120 calls.</td>
</tr>
<tr>
<td>Call Type</td>
<td>Call types are V.110, V.120, and modem.</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug mgcp packet</td>
<td>Displays the MGCP signaling message received and sent to the called agent.</td>
</tr>
<tr>
<td>debug voip rawmsg</td>
<td>Displays the raw message owner, length, and pointer.</td>
</tr>
</tbody>
</table>
debug cdma pdsn a10 ahdlc

To display debug messages for Asynchronous High-Level Data Link Control (AHDLC), use the `debug cdma pdsn a10 ahdlc` command in privileged EXEC mode. To disable debugging output, use the `no debug cdma pdsn a10 ahdlc [errors| events]` command.

### Syntax Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>errors</code></td>
<td>(Optional) Displays details of AHDLC packets in error.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>(Optional) Displays AHDLC events.</td>
</tr>
</tbody>
</table>

### Command Default

If the command is entered without any optional keywords, all of the types of debug information are enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>Keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4).</td>
</tr>
</tbody>
</table>

### Examples

The following is sample output from the `debug cdma pdsn a10 ahdlc [errors| events]` command:

```
Router# debug cdma pdsn a10 ahdlc errors
ahdlc error packet display debugging is on
Router# debug cdma pdsn a10 ahdlc events
ahdlc events display debugging is on
Router#
*Jan 1 00:18:30:*LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
*Jan 1 00:18:30:******OPEN AHDLC******
*Jan 1 00:18:30: ahdlc_mgr_channel_create
*Jan 1 00:18:30: ahdlc_mgr_allocate_available_channel:
*Jan 1 00:18:30:ahdlc:tell h/w open channel 9 from engine 0
```
debug cdma pdsn a10 gre

To display debug messages for A10 Generic Routing Encapsulation (GRE) interface errors, events, and packets, use the **debug cdma pdsn a10 gre** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

**debug cdma pdsn a10 gre [errors| events| packets] [tunnel-key key]**

**no debug cdma pdsn a10 gre [errors| events| packets]**

### Syntax Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>errors</strong></td>
<td>(Optional) Displays A10 GRE errors.</td>
</tr>
<tr>
<td><strong>events</strong></td>
<td>(Optional) Displays A10 GRE events.</td>
</tr>
<tr>
<td><strong>packets</strong></td>
<td>(Optional) Displays transmitted or received A10 GRE packets.</td>
</tr>
<tr>
<td><strong>tunnel-key</strong> key</td>
<td>(Optional) Specifies the GRE key.</td>
</tr>
</tbody>
</table>

### Command Default

If the command is entered without any optional keywords, all of the types of debug information are enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>The <strong>tunnel-key</strong> keyword was added and the existing keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

### Examples

The following is sample output from the **debug cdmapdsna10greeventstunnel-key** command:

```
Router# debug cdma pdsn a10 gre events tunnel-key 1

Router# show debug
CDMA:
  CDMA PDSN A10 GRE events debugging is on for tunnel key 1
PDSN#
  *Mar 1 04:00:57.847:CDMA-GRE:CDMA-IX1 (GRE/CDMA) created with src 5.0.0.2 dst 0.0.0.0
  *Mar 1 04:00:57.847:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
  *Mar 1 04:00:59.863:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
  *Mar 1 04:00:59.863:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
  *Mar 1 04:01:01.879:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
```
Mar 1 04:01:01.879:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
Mar 1 04:01:03.899:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
Mar 1 04:01:03.899:CDMA-GRE:(in) found session 5.0.0.2-4.0.0.1-1
debug cdma pdsn a10 ppp

To display debug messages for A10 Point-to-Point protocol (PPP) interface errors, events, and packets, use the `debug cdma pdsn a10 ppp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
ddebug cdma pdsn a10 ppp [errors| events| packets]
nno debug cdma pdsn a10 ppp [errors| events| packets]
```

**Syntax Description**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>(Optional) Displays A10 PPP errors.</td>
</tr>
<tr>
<td>events</td>
<td>(Optional) Displays A10 PPP events.</td>
</tr>
<tr>
<td>packets</td>
<td>(Optional) Displays transmitted or received A10 PPP packets.</td>
</tr>
</tbody>
</table>

**Command Default**

If the command is entered without any optional keywords, all of the types of debug information are enabled.

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>Keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn a10 ppp` command:

```
Router# debug cdma pdsn a10 ppp errors
CDMA PDSN A10 errors debugging is on
Router# debug cdma pdsn a10 ppp events
CDMA PDSN A10 events debugging is on
Router# debug cdma pdsn a10 ppp packets
CDMA PDSN A10 packet debugging is on
Router# show debug
*Jan 1 00:13:09:CDMA-PPP:create_va tunnel=CDMA-Ix1 virtual-template template=Virtual-Template2 ip_enabled=1
*Jan 1 00:13:09:CDMA-PPP:create_va va=Virtual-Access1
*Jan 1 00:13:09:CDMA-PPP:clone va=Virtual-Access1 subif_state=1 hwidb->state=0
*Jan 1 00:13:09:linestate=1 ppp_lineup=0
```
debug backhaul-session-manager session through debug channel packets

debug cdma pdsn a10 ppp

*Jan 1 00:13:09:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
*Jan 1 00:13:09:CDMA-PPP:clone va=Virtual-Access1 subif_state=1 hwidb->state=4
*Jan 1 00:13:09: linestate=0 ppp_lineup=0
*Jan 1 00:13:09:*****OPEN AHDLC*****
**debug cdma pdsn a11**

To display debug messages for A11 interface errors, events, and packets, use the `debug cdma pdsn a11` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
dump cdma pdsn a11 [errors|events|packets] [mnid]
no dump cdma pdsn a11 [errors|events|packets]
```

### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>(Optional) Displays A11 protocol errors.</td>
</tr>
<tr>
<td>events</td>
<td>(Optional) Displays A11 events.</td>
</tr>
<tr>
<td>packets</td>
<td>(Optional) Displays transmitted or received packets.</td>
</tr>
<tr>
<td>mnid</td>
<td>(Optional) Specifies the ID of the mobile station.</td>
</tr>
</tbody>
</table>

### Command Default

If the command is entered without any optional keywords, all of the types of debug information are enabled.

### Command Modes

Privileged EXEC

### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>The <code>mnid</code> argument was added and the existing keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

### Examples

The following is sample output from the `debug cdma pdsn a11` commands:

```
Router# debug cdma pdsn a11 all errors
CDMA PDSN A11 all errors debugging is on
Router# debug channel packets
1d21h:CDMA-RP:(in) rp_msgs, code=1, status=0
1d21h:CDMA-RP:(enqueue req) type=1 homeagent=5.0.0.2 coaddr=4.0.0.1
id=0xBEF750F0-0xBA53E0F lifetime=65535
1d21h:CDMA-RP:len=8, 00-00-00-00-00-00-00-F1 convert to 00000000000001 (14 digits), type=IMSI
1d21h:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
id=65535 id=BEF750F0-BA53E0F
imsi=00000000000001
1d21h:CDMA-RP:(req) rp_req_create, 5.0.0.2-4.0.0.1-1 imsi=00000000000001
1d21h:CDMA-RP:(out) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=65535
1d21h:CDMA-RP:(out) setup_dp_out_msg, ha=5.0.0.2 coa=4.0.0.1 key=1
```
ld2h:%LINK-3-UPDOWN:Interface Virtual-Access2000, changed state to up
ld2h:CDMA-RP:ipmobile_visitor add/delete=1, mn=8.0.2.132, ha=7.0.0.2
ld2h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access2000, changed state to up
Router# debug cdma pdsn a11 packets events
Router# show debug
CDMA:
CDMA PDSN A11 packet debugging is on for mnid 0000000000000001
CDMA PDSN A11 events debugging is on for mnid 0000000000000001
Router# *
 Mar 1 03:15:32.507:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 0000000000000001 (15 digits), type=IMSI
 Mar 1 03:15:32.507:CDMA-RP:extension type=38, len=0
 Mar 1 03:15:32.507:CDMA-RP:extension type=38, len=0
 Mar 1 03:15:32.507:CDMA-RP:extension type=38, len=0
 Mar 1 03:15:32.507:CDMA-RP:extension type=32, len=20
 Mar 1 03:15:32.507:SA 64 D5 9C
 Mar 1 03:15:32.507:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
 Mar 1 03:15:32.507: lifetime=1800 id=AF3BFE55-69A109D IMSI=0000000000000001
 Mar 1 03:15:32.507:CDMA-RP:(req) rp_req_create, ha=5.0.0.2, coa=4.0.0.1, key=1
 IMSI=0000000000000001
 Mar 1 03:15:32.507:CDMA-RP:(req) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=1800
 Mar 1 03:15:32.507:CDMA-RP:(req) SetUp RP out message, ha=5.0.0.2 coa=4.0.0.1 key=1
 Mar 1 03:15:38.555:CDMA-RP:(out) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=0
 Mar 1 03:15:38.555:CDMA-RP:(out) Setup RP out message, ha=5.0.0.2 coa=4.0.0.1 key=1
 Router# debug cdma pdsn a11 event mnid 0000000000000001
Router# show debug
CDMA:
CDMA PDSN A11 events debugging is on for mnid 0000000000000001
Router# *
 Mar 1 03:15:34.339:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 0000000000000001 (15 digits), type=IMSI
 Mar 1 03:15:34.339:CDMA-RP:extension type=38, len=0
 Mar 1 03:15:34.339:CDMA-RP:extension type=38, len=0
 Mar 1 03:15:34.339:CDMA-RP:extension type=38, len=0
 Mar 1 03:15:34.339:CDMA-RP:extension type=32, len=20
 Mar 1 03:15:34.339:SA 5A 56 45
 Mar 1 03:15:34.339:CDMA-RP:(req) process_rp_req, homeagent=5.0.0.2 coaddr=4.0.0.1
 Mar 1 03:15:34.339: lifetime=1800 id=AF3BFE6B-4616E475 IMSI=0000000000000001
 Mar 1 03:15:34.339:CDMA-RP:(req) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=0
 Mar 1 03:15:34.339:CDMA-RP:(req) rp_reply session=5.0.0.2-4.0.0.1-1, lifetime=0
 Router# close the session
Router# *
 Mar 1 03:10:00.575:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 0000000000000001 (15 digits), type=IMSI
 Mar 1 03:10:00.575:CDMA-RP:extension type=38, len=0
 Mar 1 03:10:00.575:CDMA-RP:extension type=38, len=0
 Mar 1 03:10:00.575:CDMA-RP:extension type=38, len=0
 Mar 1 03:10:00.575:CDMA-RP:extension type=32, len=20
 Router# close the session
Router# debug cdma pdsn a11 packet mnid 0000000000000001
Router# show debug
CDMA:
CDMA PDSN A11 packet debugging is on for mnid 0000000000000001
Router# *
 Mar 1 03:13:37.803:CDMA-RP:len=8, 01-00-00-00-00-00-00-10 convert to 0000000000000001 (15 digits), type=IMSI
 Mar 1 03:13:37.803:CDMA-RP:extension type=38, len=0
 Mar 1 03:13:37.803:CDMA-RP:extension type=38, len=0
 Mar 1 03:13:37.803:CDMA-RP:extension type=38, len=0
 Cisco IOS Debug Command Reference - Commands A through D
*Mar 1 03:13:37.803: 00 00 01 00 A8 5B 30 0D 4E 2B 83 FE 18 C6 9D C2
*Mar 1 03:13:37.803: 15 BF 5B 57
*Mar 1 03:13:51.575:CDMA-RP:extension type=38, len=0
*Mar 1 03:13:51.575: 00 00 01 00 58 77 E5 59 67 B5 62 15 17 52 83 6D
*Mar 1 03:13:51.579: DC 0A B0 5B
debug cdma pdsn accounting

To display debug messages for accounting events, use the `debug cdma pdsn accounting` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```plaintext
debug cdma pdsn accounting
no cdma pdsn accounting
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn accounting` command:

```plaintext
Router# debug cdma pdsn accounting
CDMA PDSN accounting debugging is on
Router#
*Jan 1 00:15:32:CDMA/ACCT: Call VAccess in session, start
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9] 01 CDMA/ACCT: Processing Y1
*Jan 1 00:15:32:CDMA/ACCT: Setup airlink record received
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[12] 00 00 00 02 CDMA/ACCT: Processing Y2
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9] 12 CDMA/ACCT: Processing Y3
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[12] 00 00 00 04 04 04 04 04 04 04 04 04 04 04 04 04 CDMA/ACCT: Processing D3
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[14] 00 00 00 02 CDMA/ACCT: Processing D4
*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[9] 00 00 00 02 CDMA/ACCT: Processing E1
```

Cisco IOS Debug Command Reference - Commands A through D
debug backhaul-session-manager session through debug channel packets

debug cdma pdsn accounting

*Jan 1 00:15:32:CDMA/ACCT: Current Attribute type:0x[1A] len:[10]
*Jan 1 00:15:32:CDMA/ACCT: VSA Vid:5535 type:[12] len:[4] 00 F1 Processing F1
**debug cdma pdsn accounting flow**

To display debug messages for accounting flow, use the `debug cdma pdsn accounting flow` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn accounting flow
no debug cdma pdsn accounting flow
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(2)XC</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn accounting flow` command:

```
Router# debug cdma pdsn accounting flow
CDMA PDSN flow based accounting debugging is on
pdsn-6500#
01:59:40:CDMA-SM:cdma_pdsn_flow_acct_upstream sess id 1 flow type 0 bytes 100 addr 20.20.20.1
01:59:40:CDMA-SM:cdma_pdsn_flow_acct_downstream sess id 1 flow type 0 bytes 100 addr 20.20.20.1
```
**debug cdma pdsn accounting time-of-day**

To display the timer value, use the `debug cdma pdsn accounting time-of-day` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```bash
debug cdma pdsn accounting time-of-day
no debug cdma pdsn accounting time-of-day
```

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

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**
The following is sample output from the `debug cdma pdsn accounting time-of-day` command:

```
Router# debug cdma pdsn accounting time-of-day
CDMA PDSN accounting time-of-day debugging is on
Feb 15 19:13:24.194:%SYS-5-CONFIG_I:Configured from console by console
Router#
Feb 15 19:13:45.635:CDMA-TOD:Timer expired...Rearming timer
Feb 15 19:13:45.635:CDMA-TOD:Gathering session info
Feb 15 19:13:45.635:CDMA-TOD:Found 0 sessions
```
debug cdma pdsn cluster

To display the error messages, event messages, and packets received, use the `debug cdma pdsn cluster` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn cluster message [error| events| packets] redundancy [error| events| packets]
no debug cdma pdsn cluster message [error| events| packets] redundancy [error| events| packets]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>Displays cluster messages for errors, events and packets received.</td>
</tr>
<tr>
<td>redundancy</td>
<td>Displays redundancy information for errors, events, and sent or received packets.</td>
</tr>
<tr>
<td>error</td>
<td>Displays either cluster or redundancy error messages.</td>
</tr>
<tr>
<td>events</td>
<td>Displays either all cluster or all redundancy events.</td>
</tr>
<tr>
<td>packets</td>
<td>Displays all transmitted or received cluster or redundancy packets.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This debug is *only* allowed on PDSN c6-mz images, and helps to monitor prepaid information.

**Examples**

The following is sample output from the `debug cdma pdsn cluster` command:

```
Router# debug cdma pdsn cluster ?
message  Debug PDSN cluster controller messages
redundancy  Debug PDSN cluster controller redundancy
```
debug cdma pdsn ipv6

To display IPV6 error or event messages, use the debug cdma pdsn IPV6 command in privileged EXEC mode. To disable debug messages, use the no form of this command.

ddebug cdma pdsn ipv6
ndon debug cdma pdsn ipv6

Syntax Description There are no arguments or keywords for this command.

Command Default No default behavior or values.

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(14)YX</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines The following example illustrates the debug cdma pdsn ipv6 command:

Router# debug cdma pdsn ipv6
**debug cdma pdsn prepaid**

To display debug messages about prepaid flow, use the `debug cdma pdsn prepaid` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

**debug cdma pdsn prepaid**

**no debug cdma pdsn prepaid**

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)BY</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

This debug is *only* allowed on PDSN c6-mz images, and helps to monitor prepaid information.

**Examples**

The following is sample output from the `debug cdma pdsn prepaid` command:

```
Router# debug cdma pdsn prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Initialized the authorization request
*Mar 1 00:09:38.391: CDMA-PREPAID: Added username into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added CLID into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added session id for prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Added correlation id into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added auth reason for prepaid into A-V list
*Mar 1 00:09:38.391: CDMA-PREPAID: Added USER_ID for prepaid
*Mar 1 00:09:38.391: CDMA-PREPAID: Built prepaid VSAs
*Mar 1 00:09:38.391: CDMA-PREPAID: Sent the request to AAA
*Mar 1 00:09:38.391: CDMA-PREPAID: Auth_reason: CRB_RSP_PEND INITIAL QUOTA
*Mar 1 00:09:38.395: CDMA-PREPAID: Received prepaid response: status 2
*Mar 1 00:09:38.395: CDMA-PREPAID: AAA authorised parms being processed
*Mar 1 00:09:38.395: CDMA-PREPAID: Attr in Grp Prof: crb-entity-type
*Mar 1 00:09:38.395: (0x4B000000) CDMA-PREPAID: AAA_AT_CRB_ENTITY_TYPE
*Mar 1 00:09:38.395: (0x4B000000) CDMA-PREPAID: entity type returns 1
*Mar 1 00:09:38.395: CDMA-PREPAID: Attr in Grp Prof: crb-duration
*Mar 1 00:09:38.395: (0x4B000000) CDMA-PREPAID: AAA_AT_CRB_DURATION
*Mar 1 00:09:38.395: (0x4B000000) CDMA-PREPAID: duration returns 120
*Mar 1 00:09:38.395: CDMA-PREPAID: Retrieved attributes successfully
*Mar 1 00:09:38.395: CDMA-PREPAID: Reset duration to 120, mn 9.3.0.1
*Mar 1 00:09:38.395: CDMA-PREPAID: Started duration timer for 120 sec
```
debug cdma pdsn qos

To display debug messages about quality of service features, use the debug cdma pdsn qos command in privileged EXEC mode. To disable debug messages, use the no form of this command.

debug cdma pdsn qos
no debug cdma pdsn qos

Syntax Description
There are no arguments or keywords for this command.

Command Default
There are no default values for this command.

Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3(8)XW</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(11)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(11)T.</td>
</tr>
</tbody>
</table>

Examples
There are currently no sample outputs for this command.
**debug cdma pdns resource-manager**

To display debug messages that help you monitor the resource-manager information, use the `debug cdma pdns resource-manager` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdns resource-manager [error| events]
no debug cdma pdns resource-manager [error| events]
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>errors</code></td>
<td>Displays Packet Data Service node (PDSN) resource manager errors.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>Displays PDSN resource manager events.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(8)BY</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdns resource-manager` command:

```
Router# debug cdma pdns resource-manager

errors CDMA PDSN resource manager errors
events CDMA PDSN resource manager events
```
debug cdma pdsn selection

To display debug messages for the intelligent Packet Data Serving Node (PDSN) selection feature, use the `debug cdma pdsn selection` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default cdma pdsn selection {errors| events| packets}
no default cdma pdsn selection {errors| events| packets}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>Displays PDSN selection errors.</td>
</tr>
<tr>
<td>events</td>
<td>Displays PDSN selection events.</td>
</tr>
<tr>
<td>packets</td>
<td>Displays transmitted or received packets.</td>
</tr>
</tbody>
</table>

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn selection` command with the keyword `events` specified:

```
Router# debug cdma pdsn selection events

CDMA PDSN selection events debugging is on
Router# 00:27:46: CDMA-PSL: Message(IN) pdsn 51.4.2.40 interface 70.4.2.40
00:27:46: Keepalive 10
00:27:46: Count 0
00:27:46: Capacity 16000
00:27:46: Weight 0
00:27:46: Hostname 11 7206-PDSN-2
00:27:46: CDMA-PSL: Reset keepalive, pdsn 51.4.2.40 current 10 new 10
00:27:46: CDMA-PSL: Message processed, pdsn 51.4.2.40 tsize 0 pending 0
00:27:47: CDMA-PSL: Send KEEPALIVE, len 32
00:27:47: CDMA-PSL: Message(OUT) dest 224.0.0.11
00:27:47: Keepalive 10
00:27:47: Count 1
00:27:47: Capacity 16000
00:27:47: Weight 0
```
debug cdma pdsn selection

00:27:47: Hostname 11 7206-PDSN-1
00:27:47: CDMA-PSL: RRQ sent, s=70.4.1.40 (FastEthernet0/1), d=224.0.0.11
**debug cdma pdsn service-selection**

To display debug messages for service selection, use the `debug cdma pdsn service-selection` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn service-selection
no debug cdma pdsn service-selection
```

**Syntax Description**

This command has no arguments or keywords.

**Command Default**

No default behavior or values

**Command Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following is sample output from the `debug cdma pdsn service-selection` command:

```
Router# debug cdma pdsn service-selection
CDMA PDSN service provisioning debugging is on
1d02h:%LINK-3-UPDOWN:Interface Virtual-Access3, changed state to up
1d02h:Vi3 CDMA-SP:user_class=1, ms_ipaddr_req=1, apply_acl=0
1d02h:Vi3 CDMA-SP:Adding simple ip flow, user=bsip, mn=6.0.0.2,
1d02h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access3, changed state to up
```
### debug cdma pdsn session

To display debug messages for Session Manager errors, events, and packets, use the `debug cdma pdsn session` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdma pdsn session [errors| events]
no debug cdma pdsn session [errors| events]
```

#### Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>errors</code></td>
<td>(Optional) Displays session protocol errors.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>(Optional) Displays session events.</td>
</tr>
</tbody>
</table>

#### Command Default

If the command is entered without any optional keywords, all of the types of debug information are enabled.

#### Command Modes

Privileged EXEC

#### Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(3)XS</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(8)BY</td>
<td>Keywords were made optional.</td>
</tr>
<tr>
<td>12.3(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.3(4)T.</td>
</tr>
</tbody>
</table>

#### Examples

The following is sample output from the `debug cdma pdsn session` command:

```
Router# debug cdma pdsn session events
CDMA PDSN session events debugging is on
Router# debug cdma pdsn session errors
CDMA PDSN session errors debugging is on
Router# show debug
CDMA:
    CDMA PDSN session events debugging is on
    CDMA PDSN session errors debugging is on
Router# *Jan 1 00:22:27:CDMA-SM:create_session 5.5.5.5-4.4.4.5-2
*Jan 1 00:22:27:CDMA-SM:create_tunnel 5.5.5.5-4.4.4.5
*Jan 1 00:22:27:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
*Jan 1 00:22:29:CDMA-SM:create_flow mn=0.0.0.0, ha=8.8.8.8 nai=l2tp2@cisco.com
*Jan 1 00:22:30:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access1, changed state to up
```
debug cdp

To enable debugging of the Cisco Discovery Protocol (CDP), use the `debug cdp` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
debug cdp {packets| adjacency| events}
no debug cdp {packets| adjacency| events}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Packets</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packets</td>
<td>Enables packet-related debugging output.</td>
</tr>
<tr>
<td>adjacency</td>
<td>Enables adjacency-related debugging output.</td>
</tr>
<tr>
<td>events</td>
<td>Enables output related to error messages, such as detecting a bad checksum.</td>
</tr>
</tbody>
</table>

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(2)T</td>
<td>This command was introduced in a release earlier than Cisco IOS Release 12.4(2)T.</td>
</tr>
<tr>
<td>12.2(55)SE</td>
<td>This command was modified. The debug output was enhanced to display location Type-Length-Values (TLVs), location-server TLVs, and application TLV-related debugs.</td>
</tr>
</tbody>
</table>

**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 the `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-FA: Packet sent out on Ethernet0
CDP-FA: 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
```
### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdp tlv</td>
<td>Configures location support in CDP.</td>
</tr>
<tr>
<td>show cdp tlv</td>
<td>Displays information about CDP TLVs.</td>
</tr>
</tbody>
</table>
**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` command in privileged EXEC mode. 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.

**Command Modes**

Privileged EXEC

**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 `debugcdppacket` and `debugcdpadjacency` 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug ip routing</td>
<td>Displays information on RIP routing table updates and route cache updates.</td>
</tr>
</tbody>
</table>
debug cef

To enable the display of information about Cisco Express Forwarding events, use the `debug cef` command in privileged EXEC mode. To disable the display of Cisco Express Forwarding events, use the `no` form of this command.

```
dump cef {all| assert| background| broker| consistency-check| elog| epoch| fib [attached export| subblock]| hardware [notification| queries]| hash| high-availability| interest| interface| iprm| issu| loadinfo| memory| non-ip| path [extension| list| scope]| subtree context| switching background| table| xdr}
```

Syntax Description

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Displays debug messages for all Cisco Express Forwarding events.</td>
</tr>
<tr>
<td>assert</td>
<td>Displays debug messages for Cisco Express Forwarding assert events.</td>
</tr>
<tr>
<td>background</td>
<td>Displays debug messages for Cisco Express Forwarding background events.</td>
</tr>
<tr>
<td>broker</td>
<td>Displays debug messages for Cisco Express Forwarding broker events.</td>
</tr>
<tr>
<td>consistency-check</td>
<td>Displays debug messages for Cisco Express Forwarding consistency checker events.</td>
</tr>
<tr>
<td>elog</td>
<td>Displays debug messages for Cisco Express Forwarding elog events.</td>
</tr>
<tr>
<td>epoch</td>
<td>Displays debug messages for Cisco Express Forwarding epoch events.</td>
</tr>
<tr>
<td>fib [attached export</td>
<td>Displays debug messages for Cisco Express Forwarding Forwarding Information Base entry events.</td>
</tr>
<tr>
<td>subblock]</td>
<td>path [extension</td>
</tr>
<tr>
<td>hash</td>
<td>Displays debug messages for Cisco Express Forwarding load-balancing hash algorithms.</td>
</tr>
</tbody>
</table>
**Command Default**  
Debugging information about Cisco Express Forwarding events is not displayed.

**Command Modes**  
Privileged EXEC (#)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>high-availability</td>
<td>Displays debug messages for Cisco Express Forwarding high availability events.</td>
</tr>
<tr>
<td>interest</td>
<td>Displays debug messages for Cisco Express Forwarding interest list events.</td>
</tr>
<tr>
<td>interface</td>
<td>Displays debug messages for Cisco Express Forwarding interface events.</td>
</tr>
<tr>
<td>iprm</td>
<td>Displays debug messages for Cisco Express Forwarding IP rewrite manager events. (This keyword is not available in Cisco IOS Release 12.2(33)SRA.)</td>
</tr>
<tr>
<td>issu</td>
<td>Displays debug messages for Cisco Express Forwarding In Service Software Upgrade (ISSU) events.</td>
</tr>
<tr>
<td>loadinfo</td>
<td>Displays debug messages for Cisco Express Forwarding loadinfo events.</td>
</tr>
<tr>
<td>memory</td>
<td>Displays debug messages for Cisco Express Forwarding memory events.</td>
</tr>
<tr>
<td>non-ip</td>
<td>Displays debug messages for Cisco Express Forwarding non-IP entry events.</td>
</tr>
<tr>
<td>path</td>
<td>Displays debug messages for Cisco Express Forwarding path events.</td>
</tr>
<tr>
<td>subtree</td>
<td>Displays debug messages for Cisco Express Forwarding subtree context events.</td>
</tr>
<tr>
<td>switching</td>
<td>Displays debug messages for Cisco Express Forwarding switching background events.</td>
</tr>
<tr>
<td>table</td>
<td>Displays debug messages for Cisco Express Forwarding table events.</td>
</tr>
<tr>
<td>xdr</td>
<td>Displays debug messages for Cisco Express Forwarding External Data Representation (XDR) events.</td>
</tr>
</tbody>
</table>
Command History

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(25)S</td>
<td>This command was introduced. The <code>debugceffibattachedexport</code> command replaces the <code>debugipcefadjfib</code> command.</td>
</tr>
<tr>
<td>12.2(28)SB</td>
<td>This command was integrated into Cisco IOS Release 12.2(28)SB and implemented on the Cisco 10000 series routers.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
<tr>
<td>12.2(33)SXH</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SXH.</td>
</tr>
<tr>
<td>12.4(20)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(20)T.</td>
</tr>
</tbody>
</table>

Usage Guidelines

Because debugging output is assigned high priority in the CPU process, you should use debug commands only to troubleshoot specific problems or during troubleshooting sessions with Cisco technical support staff. Moreover, you should use debug commands during periods of lower network traffic and fewer users. Debugging during these periods decreases the likelihood that increased debug command processing overhead will affect system use.

Examples

The following is sample output from the `debugcefall` command:

```
Router# debug cef all
06:23:38: HW-API: Counter poll: Label[label=implicit-null]
06:23:38: HW-API: Counter poll: Label[label=implicit-null]
06:23:38: HW-API: Counter poll: Label[label=implicit-null]
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-rib-ios scanner' expired, calling 0x40FA03FC, context 0x00010003
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-rib-ios scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-rios-rib scanner' expired, calling 0x40FA03FC, context 0x00010004
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-rios-rib scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-ios-rib scanner' expired, calling 0x40FA03FC, context 0x00010002
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-ios-rib scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv4 scan-rp-lc scanner' expired, calling 0x40FA03FC, context 0x00010002
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-rp-lc scanner' with delay 60000
06:23:43: FIBbg: Timer 'FIB checkers: IPv6 scan-rp-lc scanner' expired, calling 0x40FA03FC, context 0x00020002
06:23:43: FIBbg: Restarting timer 'FIB checkers: IPv6 scan-rp-lc scanner' with delay 60000
06:24:06: FIBtable: IPv4: Event modified, 0.0.0.0/0, vrf Default, 1 path, flags 00420005
06:24:06: FIBpath: Configuring IPv4 path 444B2AB0 from rib (idb=NULL, gw=9.1.41.1, gw_table=0, rr=1) and host prefix 0.0.0.0
```
06:24:06: FIBpath: Configured recursive-nexthop 9.1.41.1[0] 444B2AB0 path
06:24:06: FIBfib: [v4-0.0.0.0/0 (44AAC750)] Mod type - null
06:24:06: FIBtable: IPv4: Event up, default, 0.0.0.0/0, vrf Default, 1 path, flags 00420005
06:24:06: FIBtable: IPv4: Adding route for 0.0.0.0/0 but route already exists. Trying modify.
06:24:06: FIBpath: Configuring IPv4 path 444B2AA0 from rib (idb=NULL, gw=9.1.41.1, gw_table=0, rr=1) and host prefix 0.0.0.0/sh ip
06:24:06: FIBpath: Configured recursive-nexthop 9.1.41.1[0] 444B2AA0 path
06:24:06: FIBfib: [v4-0.0.0.0/0 (44AAC750)] Mod type - null vrf
06:24:06: FIBbg: Timer 'FIB checkers: IPv4 scan-hw-sw scanner' expired, calling O±40FA03FC, context 0x00010005)
06:24:07: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-hw-sw scanner' with delay 60000
06:24:07: FIBbg: Timer 'FIB checkers: IPv4 scan-sw-hw scanner' expired, calling O±40FA03FC, context 0x00010006)
06:24:07: FIBbg: Restarting timer 'FIB checkers: IPv4 scan-sw-hw scanner' with delay 60000

<table>
<thead>
<tr>
<th>Name</th>
<th>Default RD</th>
<th>Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1:1</td>
<td>Ethernet4/0/5</td>
</tr>
</tbody>
</table>

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cef table consistency-check</td>
<td>Enables Cisco Express Forwarding consistency checker table values by type and parameter.</td>
</tr>
<tr>
<td>clear cef table</td>
<td>Clears the Cisco Express Forwarding tables.</td>
</tr>
<tr>
<td>clear ip cef inconsistency</td>
<td>Clears Cisco Express Forwarding inconsistency statistics and records found by the Cisco Express Forwarding consistency checkers.</td>
</tr>
<tr>
<td>debug ip cef table</td>
<td>Enables the collection of events that affect entries in the Cisco Express Forwarding tables.</td>
</tr>
<tr>
<td>show cef table consistency-check</td>
<td>Displays Cisco Express Forwarding consistency checker table values.</td>
</tr>
<tr>
<td>show ip cef inconsistency</td>
<td>Displays Cisco Express Forwarding IP prefix inconsistencies.</td>
</tr>
</tbody>
</table>
debug cell-hwic driver

To debug the Cisco IOS driver for the cellular interface, use the `debug cell-hwic driver` command in EXEC mode.

```
dump cell-hwic slot/wic_slot/port driver {crcdump|errdump|errors}
```

**Syntax Description**

<table>
<thead>
<tr>
<th>slot/wic_slot/port</th>
<th>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</th>
</tr>
</thead>
<tbody>
<tr>
<td>crcdump</td>
<td>CRC error details.</td>
</tr>
<tr>
<td>errdump</td>
<td>Other error details.</td>
</tr>
<tr>
<td>errors</td>
<td>Errors debugging.</td>
</tr>
</tbody>
</table>

**Command Default**

None

**Command Modes**

EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cellular firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>debug cellular messages management</code></td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td><code>debug cellular messages dm</code></td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td><code>debug cell-hwic virt-con</code></td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cell-hwic firmware

To see the Cisco IOS firmware information, use the `debug cell-hwic firmware` command in EXEC mode.

```
debug cellular slot/wic_slot/port firmware
```

**Syntax Description**

| `slot/wic_slot/port` | Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port. |

**Command Default**

None

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.4(22)YB1</td>
<td>This command was integrated into Cisco IOS Release 12.4(22)YB1.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug cellular messages async</code></td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td><code>debug cellular messages data</code></td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td><code>debug cellular firmware</code></td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td><code>debug cellular messages management</code></td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td><code>debug cellular messages dm</code></td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td><code>debug cell-hwic virt-con</code></td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cellular messages all

To print all Cisco IOS driver debug messages, use the `debugcellularmessagesall` command in EXEC mode.

**debug cellular slotwic_slotport messages all**

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>slot/wic_slot/port</th>
<th>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</th>
</tr>
</thead>
</table>

**Command Default**
None

**Command Modes**
EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

d debug cellular messages all
**debug cellular messages async**

To debug cellular async, use the `debugcellularmessagesasync` command in EXEC mode.

```plaintext
debug cellular slot/wic_slot/port messages async
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>slot/wic_slot/port</code></td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

**Command Default**

None

**Command Modes**

EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug cellular messages all</code></td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td><code>debug cellular messages data</code></td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td><code>debug cellular driver</code></td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td><code>debug cellular firmware</code></td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td><code>debug cellular messages management</code></td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td><code>debug cellular messages dm</code></td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug cellular messages data

To print Cisco IOS data path debug messages, use the `debugcellularmessagesdata` command in EXEC mode.

```
show cellular slot/wic_slot/port messages data
```

**Syntax Description**

| slot/wic_slot/port | Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port. |

**Command Default**

None

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

ddebug cellular messages data
debug cellular messages dm

To print Diagnostics Monitor (DM) messages from the Qualcomm CDMA chipset, use the `debugcellularmessagesdm` command in EXEC mode.

```
debug cellular slot/wic_slot/port messages dm
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

**Command Default**

There is no default for this command.

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
debug backhaul-session-manager session through debug channel packets

d debug cellular messages dm
debug cellular messages management

To print management path messages, such as CnS, use the `debugcellularmessagesmanagement` command in EXEC mode.

```plaintext
debug cellular slot/wic_slot/port messages management
```

**Syntax Description**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
</tbody>
</table>

**Command Default**

None

**Command Modes**

EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use this command for debugging purposes only.

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages virt-con</td>
<td>Redirects the Nios II console driver messages to display them in the Cisco IOS router console environment.</td>
</tr>
</tbody>
</table>
**debug cell-hwic virt-con**

To redirect the Nios II console driver messages to display them in the Cisco IOS router console environment, use the `debug cell-hwic virt-con` command in EXEC mode.

```plaintext
debug cell-hwic slot/wic_slot/port virt-con {clear| disable| dump-data-structs| log| monitor| wrapper-on| wrapper-off}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slot/wic_slot/port</td>
<td>Numeric values that indicate the router slot, WAN interface card (WIC) slot, and port.</td>
</tr>
<tr>
<td>clear</td>
<td>(Optional) Clears all virtual console debug log messages.</td>
</tr>
<tr>
<td>disable</td>
<td>(Optional) Disables virtual console real-time debug monitoring.</td>
</tr>
<tr>
<td>dump-data-structs</td>
<td>(Optional) Dumps virtual console data structures.</td>
</tr>
<tr>
<td>log</td>
<td>(Optional) Displays virtual console messages from the debug log.</td>
</tr>
<tr>
<td>monitor</td>
<td>(Optional) Enables monitoring of real-time virtual console debug messages.</td>
</tr>
<tr>
<td>wrapper-on</td>
<td>(Optional) Disables wraparound for virtual console log messages.</td>
</tr>
<tr>
<td>wrapper-off</td>
<td>(Optional) Enables wraparound for virtual console log messages.</td>
</tr>
</tbody>
</table>

**Command Default**

There is no default for this command.

**Command Modes**

EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4(11)XV</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.4(15)T</td>
<td>This command was integrated into Cisco IOS Release 12.4(15)T.</td>
</tr>
<tr>
<td>12.2SX</td>
<td>This command is supported in the Cisco IOS Release 12.2SX train. Support in a specific 12.2SX release of this train depends on your feature set, platform, and platform hardware.</td>
</tr>
</tbody>
</table>
### Usage Guidelines

Use this command for debugging purposes only.

### Related Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cellular messages all</td>
<td>Prints all Cisco IOS driver debug messages.</td>
</tr>
<tr>
<td>debug cellular messages async</td>
<td>Debugs cellular async.</td>
</tr>
<tr>
<td>debug cellular messages data</td>
<td>Prints Cisco IOS data path debug messages.</td>
</tr>
<tr>
<td>debug cell-hwic driver</td>
<td>Debugs the Cisco IOS driver.</td>
</tr>
<tr>
<td>debug cell-hwic firmware</td>
<td>Displays Cisco IOS firmware information.</td>
</tr>
<tr>
<td>debug cellular messages management</td>
<td>Prints management path messages, such as CnS.</td>
</tr>
<tr>
<td>debug cellular messages dm</td>
<td>Prints diagnostics monitor (DM) messages from the Qualcomm CDMA chipset.</td>
</tr>
</tbody>
</table>
**debug cem ls errors**

To debug connection errors or null data structures, use the `debug cem ls errors` command in privileged EXEC mode. To disable this form of debugging, use the `no` form of this command.

```
debug cem ls errors
no debug cem ls errors
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRC</td>
<td>This command was introduced on the Cisco 7600 series routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the `show debug` command to see debug information.

**Examples**

The following command turns on CEM local switching error debugging:

```
Router# debug cem ls errors
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cem ls events</td>
<td>Enables debugging of events relating to CEM local switching.</td>
</tr>
</tbody>
</table>
**debug cem ls events**

To debug CEM local switching events, use the debug cem ls events command in privileged EXEC mode. To disable this form of debugging, use the no form of this command.

```plaintext
debug cem ls events
no debug cem ls events
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SRC</td>
<td>This command was introduced on the Cisco 7600 series routers.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

Use the show debug command to see debug information.

**Examples**

The following command turns on debugging for CEM local switching events.

```plaintext
Router# debug cem ls events
```

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug cem ls errors</td>
<td>Enables debugging of connection errors or null data structures.</td>
</tr>
</tbody>
</table>
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 this command.

```
debug ces-conn [all| errors| events]
no debug ces-conn
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all</code></td>
<td>(Optional) Displays all error and event information.</td>
</tr>
<tr>
<td><code>errors</code></td>
<td>(Optional) Displays only error information.</td>
</tr>
<tr>
<td><code>events</code></td>
<td>(Optional) Displays only event information.</td>
</tr>
</tbody>
</table>

**Command Default**
No default behavior or values

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1(5)XM</td>
<td>This command is supported on Cisco 3600 series routers.</td>
</tr>
<tr>
<td>12.2(4)T</td>
<td>This command was integrated into Cisco IOS Release 12.2(4)T.</td>
</tr>
</tbody>
</table>

**Examples**

The following example shows debug output for a CES connection:

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

To enable debugging of the data path of Ethernet connectivity fault management (CFM) on Cisco Catalyst 6500 series switches, use the **debug cfm** command in privileged EXEC mode. To disable the debugging function, use the **no** form of this command.

```
debug cfm {all| api| cfmpal| common| db| isr}
no debug cfm {all| api| cfmpal| common| db| isr}
```

<table>
<thead>
<tr>
<th>Syntax Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Specifies all Catalyst 6500 switch-specific route processor and switch processor (RP/SP) events.</td>
</tr>
<tr>
<td>api</td>
<td>Specifies Catalyst 6500 switch-specific application program interface (API) events.</td>
</tr>
<tr>
<td>cfmpal</td>
<td>Specifies general Catalyst 6500 switch debugging.</td>
</tr>
<tr>
<td>common</td>
<td>Specifies common Catalyst 6500 switch RP/SP components.</td>
</tr>
<tr>
<td>db</td>
<td>Specifies Catalyst 6500 switch CFM database debugging.</td>
</tr>
<tr>
<td>isr</td>
<td>Specifies Catalyst 6500 switch-specific ingress CFM packet debugging.</td>
</tr>
</tbody>
</table>

**Command Default**

Debugging is disabled.

**Command Modes**

Privileged EXEC (#)

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.2(33)SXI2</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRE</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRE.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**

The output from this command is a log of activity.

Use this command to troubleshoot Ethernet CFM on Cisco Catalyst 6500 series switches.
Examples

The following example shows output of the **debug cfm all** command:

```
Device# debug cfm all
CFM DB events debugging is on
CFM Ingress ISR events debugging is on
CFMPAL events debugging is on
CFM API events debugging is on
CFM RP/SP COMMON events debugging is on
CFM packets debugging is on
```
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** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

```plaintext
debug channel events
no debug channel events
```

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

**Command Modes**
Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0(3)T</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**Usage Guidelines**
This command displays CMCC adapter events that occur on the Channel Interface Processor (CIP) or Channel Port Adapter (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 extendedchannels 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 Common Link Access for Workstations (CLAW) subchannel command is being sent from the Route Processor (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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug channel ilan</code></td>
<td>Displays CIP love letter events.</td>
</tr>
<tr>
<td><code>debug channel packets</code></td>
<td>Displays per-packet debugging output.</td>
</tr>
</tbody>
</table>
debug channel ilan

To display messages relating to configuration and bridging using Cisco Mainframe Channel Connection (CMCC) internal LANs and to help debug source-route bridging (SRB) problems related to CMCC internal LANs, use the `debug channel ilan` command in privileged EXEC mode. 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 Modes**

Privileged EXEC

**Command History**

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.0(3)</td>
<td>This command was introduced.</td>
</tr>
<tr>
<td>12.2(33)SRA</td>
<td>This command was integrated into Cisco IOS Release 12.2(33)SRA.</td>
</tr>
</tbody>
</table>

**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 is sample output 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 Logical Link Control (LLC) end station in Cisco IOS software did not exist:

CIP ILAN(Channel3/2-Token): Packet dropped - NULL LLC
The following line indicates that a packet destined for the CMCC via a configured internal MAC adapter configured on an internal LAN was dropped because the CMCC had not yet acknowledged the internal MAC adapter configuration command:

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>debug channel events</code></td>
<td>Displays processing that occurs on the channel adapter interfaces of all installed adapters.</td>
</tr>
<tr>
<td><code>debug source bridge</code></td>
<td>Displays information about packets and frames transferred across a source-route bridge.</td>
</tr>
</tbody>
</table>
debug channel love

To display Channel Interface Processor (CIP) love letter events, use the `debug channel love` command in privileged EXEC mode. 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.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

This command displays CIP love letter events (an operating status or configuration message) that occur on the CIP interface processor and is useful for diagnosing problems in an IBM channel attach network. It provides an overall picture of the stability of the network. In a stable network, the `debug channel love` command returns a statistic message (cip_love_letter) that is sent every 10 seconds. This command is valid for the Cisco 7000 series routers only.

**Examples**

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

```
Router# debug channel love
Channel3/1: love letter received, bytes 3308
Channel3/0: love letter received, bytes 3336

cip_love_letter: received ll, but no cip_info
```

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

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

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

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

**Related Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug channel events</td>
<td>Displays processing that occurs on the channel adapter interfaces of all installed adapters.</td>
</tr>
<tr>
<td>debug channel packets</td>
<td>Displays per-packet debugging output.</td>
</tr>
</tbody>
</table>
debug channel packets

To display per-packet debugging output, use the `debug channel packets` command in privileged EXEC mode. To disable debugging output, use the `no` form of this command.

```
default channel packets
no default channel packets
```

**Syntax Description**

This command has no arguments or keywords.

**Command Modes**

Privileged EXEC

**Usage Guidelines**

The `debug channel packets` command displays all process-level Channel Interface Processor (CIP) packets for both outbound and inbound packets. The output reports information when a packet is received or a transmission is attempted. 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 172.24.15.197, dst 172.24.1.58
(Channel3/0)-in size = 44, vc = 0000, type = 0800, src 172.24.1.58, dst 172.24.15.197
```

The table below describes the significant fields shown in the display.

**Table 20: debug channel packets Field Descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Channel3/0)</td>
<td>Interface slot and port.</td>
</tr>
<tr>
<td>in/out</td>
<td>&quot;In&quot; is a packet from the mainframe to the router. &quot;Out&quot; is a packet from the router to the mainframe.</td>
</tr>
<tr>
<td>size =</td>
<td>Number of bytes in the packet, including internal overhead.</td>
</tr>
<tr>
<td>vc =</td>
<td>Value from 0 to 511 that maps to the claw interface configuration command. This information is from the MAC layer.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>type =</td>
<td>Encapsulation type in the MAC layer. The value 0800 indicates an IP datagram.</td>
</tr>
<tr>
<td>src</td>
<td>Origin, or source, of the packet, as opposed to the previous hop address.</td>
</tr>
<tr>
<td>dst</td>
<td>Destination of the packet, as opposed to the next-hop address.</td>
</tr>
</tbody>
</table>