debug voip avlist through debug vpm signaling

debug voip application lpcor

To enable debugging of the logical partitioning class of restriction (LPCOR) application system, use the **debug voip application lpcor** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip application lpcor

no debug voip application lpcor

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.0(1)XA	This command was introduced.
15.1(1)T	This command was integrated into Cisco IOS Release 15.1(1)T.

Examples

The following is sample output from the **debug voip application lpcor**command for a call between two phones that was blocked by LPCOR policy validation:

```
Router# debug voip application lpcor
voip application AFW lpcor debugging is on CME#
*Jun 24 11:24:58.115: //44//Dest:/DestOutboundCallUsingPeer: Save Lpcor Index 1 to
Interworking Leg
*Jun 24 11:24:58.119: //44//Dest:/DestProcessLPCOR: Peer 20002 Source Callid 44 CallType 0
 *Jun 24 11:24:58.119: //44//Dest:/DestProcessLPCOR: lpcor source index(1) target index (2)
 ret_cause=63
*Jun 24 11:24:58.119: //44//Dest:/DestSetup: lpcor block with peerTag 20002
```

Related Commands

Command	Description
debug ephone lpcor	Displays debugging information for calls using the LPCOR feature.
debug voip lpcor	Displays debugging information for the LPCOR feature.
show voice lpcor policy	Displays the LPCOR policy for the specified resource group.
voice lpcor enable	Enables LPCOR functionality on the Cisco Unified CME router.

Command	Description
voice lpcor policy	Creates a LPCOR policy for a resource group.

debug voip avlist

To troubleshoot the attribute value list (AVLIST) contents, use the **debug voip avlist** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip avlist [**all**| **default**| **detail**| **error** [**call [informational]**| **software [informational]**]| **inout**]

no debug voip avlist

Syntax Description

all	(Optional) Displays all AVLIST debugging messages.
default	(Optional) Displays AVLIST error and inout information. This option also runs if no keywords are added.
detail	(Optional) Displays AVLIST background messages.
error	(Optional) Displays AVLIST error messages.
call	(Optional) Displays call processing errors.
informational	(Optional) Displays minor errors and major errors. Without the **informational** keyword, only major errors are displayed.
software	(Optional) Displays software errors.
inout	(Optional) Displays AVLIST in/out functions.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.

Usage Guidelines

The **debug voip avlist** command does not support call debug filtering.

Examples

Output is primarily used by TAC.

debug voip ccapi

To troubleshoot the call control application programming interface (CCAPI) contents, use the **debug voip ccapi** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip ccapi [**all**| **default**| **detail**| **error** [**call [informational]**| **software [informational]**]| **individual** *range*| **inout**| **function**| **protoheaders**| **service**]

no debug voip ccapi

Syntax Description

all	(Optional) Displays all CCAPI debugging messages.
default	(Optional) Displays CCAPI error and inout information. This option also runs if no keywords are added.
detail	(Optional) Displays CCAPI background messages.
error	(Optional) Displays CCAPI error messages. The **debug voip ccapi error** command traces the error logs in the call control API. Error logs are generated during normal call processing, if there are insufficient resources, or if there are problems in the underlying network-specific code, the higher call session application, or the call control API itself. This debug command shows error events or unexpected behavior in system software. Usually no events will be generated.
call	(Optional) Displays call processing errors.
informational	(Optional) Displays minor errors and major errors. Without the **informational** keyword, only major errors are displayed.
software	(Optional) Displays software errors.
individual	(Optional) Enables individual CCAPI debug outputs.
range	For the **individual** keyword, the range is an integer value from 1 to 146. For specific range values, see the table below.

inout	(Optional) Displays CCAPI in/out functions. The **debug voip ccapi inout** command traces the execution path through the call control API, which serves as the interface between the call session application and the underlying network-specific software. You can use the output from this command to understand how calls are being handled by the router. This command shows how a call flows through the system. Using this debug level, you can see the call setup and teardown operations performed on both the telephony and network call legs.
function	(Optional) Displays CCAPI function tracing.
protoheaders	(Optional) Displays CCAPI protocol headers passing information.
service	(Optional) Logs debug messages that are not call related.

Table 1: CCAPI Individual Debug Values

Value	CCAPI Debug Function
1	CC_IDMSG_API_DISPLAY_IES
2	CC_IDMSG_SETUP_IND_COMM_2
3	CC_IDMSG_SETUP_IND_COMM_3
4	CC_IDMSG_SETUP_IND_COMM_4
5	CC_IDMSG_ALERT_IND_5
6	CC_IDMSG_ALERT_IND_6
7	CC_IDMSG_CONNECT_IND_7
8	CC_IDMSG_CONNECT_IND_8
9	CC_IDMSG_RECONNECT_IND_9
10	CC_IDMSG_DISCONNECTED_IND_10
11	CC_IDMSG_DISCONNECTED_IND_11
12	CC_IDMSG_DISCONNECTED_IND_12

Value	CCAPI Debug Function
13	CC_IDMSG_DISCONNECT_DONE_IND_13
14	CC_IDMSG_DISCONNECT_DONE_IND_14
15	CC_IDMSG_DISCONNECT_DONE_IND_15
16	CC_IDMSG_PRE_DISC_CAUSE_16
17	CC_IDMSG_PRE_DISC_CAUSE_17
18	CC_IDMSG_DIGIT_BEGIN_IND_18
19	CC_IDMSG_DIGIT_END_IND_19
20	CC_IDMSG_DIGIT_END_IND_20
21	CC_IDMSG_DIGIT_END_NO_TERM_21
22	CC_IDMSG_TONE_IND_22
23	CC_IDMSG_FEATURE_IND_23
24	CC_IDMSG_MODIFY_DONE_IND_24
25	CC_IDMSG_MODIFY_MODE_DONE_IND_25
26	CC_IDMSG_INBAND_MSG_RCVD_IND_26
27	CC_IDMSG_INBAND_MSG_DONE_IND_27
28	CC_IDMSG_UPD_CALL_INFO_IND_28
29	CC_IDMSG_GEN_NTK_ALERT_EVENT_29
30	CC_IDMSG_VOICE_MODE_EVENT_30
31	CC_IDMSG_VOICE_MODE_EVENT_31
32	CC_IDMSG_DIALING_COMPLETE_IND_32
33	CC_IDMSG_DIGITS_DONE_IND_33
34	CC_IDMSG_DIGITS_DONE_IND_34
35	CC_IDMSG_VBD_XMIT_DONE_IND_35
36	CC_IDMSG_FWD_SETUP_IND_36

Value	CCAPI Debug Function
37	CC_IDMSG_RSVP_DONE_IND_37
38	CC_IDMSG_AUDIT_RSP_IND_38
39	CC_IDMSG_XFR_STATUS_IND_39
40	CC_IDMSG_XFR_STATUS_IND_40
41	CC_IDMSG_XFR_DONE_IND_41
42	CC_IDMSG_XFR_DONE_IND_42
43	CC_IDMSG_XFR_DONE_IND_43
44	CC_IDMSG_TGT_CID_ACTIVE_RCD_44
45	CC_IDMSG_MODIFY_MEDIA_IND_45
46	CC_IDMSG_MODIFY_MEDIA_ACK_IND_46
47	CC_IDMSG_MODIFY_MEDIA_REJ_IND_47
48	CC_IDMSG_MODEM_CALL_START_IND_48
49	CC_IDMSG_MODEM_CALL_DONE_IND_49
50	CC_IDMSG_ACCT_STATUS_IND_50
51	CC_IDMSG_NW_STATUS_IND_51
52	CC_IDMSG_DESTINFO_IND_52
53	CC_IDMSG_LOOPBACK_DONE_IND_53
54	CC_IDMSG_RT_PACKET_STATS_IND_54
55	CC_IDMSG_CUT_PROGRESS_IND_55
56	CC_IDMSG_CUT_PROGRESS_IND_56
57	CC_IDMSG_PROCEEDING_IND_57
58	CC_IDMSG_FACILITY_IND_58
59	CC_IDMSG_INFO_IND_59
60	CC_IDMSG_PROGRESS_IND_60

Value	CCAPI Debug Function
61	CC_IDMSG_USERINFO_IND_61
62	CC_IDMSG_DISC_PROG_IND_62
63	CC_IDMSG_DISC_PROG_IND_63
64	CC_IDMSG_PING_DONE_IND_64
65	CC_IDMSG_COT_TEST_DONE_IND_65
66	CC_IDMSG_PROCESS_DONE_IND_66
67	CC_IDMSG_ASSOCIATED_IND_67
68	CC_IDMSG_SUSPEND_IND_68
69	CC_IDMSG_SUSPEND_ACK_IND_69
70	CC_IDMSG_SUSPEND_REJ_IND_70
71	CC_IDMSG_RESUME_IND_71
72	CC_IDMSG_RESUME_ACK_IND_72
73	CC_IDMSG_RESUME_REJ_IND_73
74	CC_IDMSG_IF_SETUP_REQ_PRIV_74
75	CC_IDMSG_IF_SETUP_REQ_PRIV_75
76	CC_IDMSG_IF_ALLOCATE_DSP_76
77	CC_IDMSG_CONNECT_77
78	CC_IDMSG_CONNECT_78
79	CC_IDMSG_PING_79
80	CC_IDMSG_DISCONNECT_80
81	CC_IDMSG_DISCONNECT_81
82	CC_IDMSG_DISCONNECT_82
83	CC_IDMSG_ALERT_83
84	CC_IDMSG_ALERT_84

Value	CCAPI Debug Function
85	CC_IDMSG_CUT_PROGRESS_85
86	CC_IDMSG_CUT_PROGRESS_86
87	CC_IDMSG_CUT_PROGRESS_87
88	CC_IDMSG_DISC_PROG_88
89	CC_IDMSG_DISC_PROG_89
90	CC_IDMSG_SET_PEER_90
91	CC_IDMSG_SET_PEER_91
92	CC_IDMSG_PROCEEDING_92
93	CC_IDMSG_SETUP_REQ_93
94	CC_IDMSG_SETUP_REQ_94
95	CC_IDMSG_SETUP_REQ_95
96	CC_IDMSG_SETUP_REQ_96
97	CC_IDMSG_SETUP_REQ_97
98	CC_IDMSG_SETUP_REQ_98
99	CC_IDMSG_SETUP_REQ_99
100	CC_IDMSG_SETUP_REQ_100
101	CC_IDMSG_SETUP_REQ_101
102	CC_IDMSG_SETUP_ACK_102
103	CC_IDMSG_FACILITY_103
104	CC_IDMSG_TRANSFER_REQ_104
105	CC_IDMSG_GET_CONSULT_ID_105
106	CC_IDMSG_FORWARD_TO_106
107	CC_IDMSG_INFO_107
108	CC_IDMSG_NOTIFY_108

Value	CCAPI Debug Function
109	CC_IDMSG_PROGRESS_109
110	CC_IDMSG_PRE_DISC_110
111	CC_IDMSG_PRE_DISC_111
112	CC_IDMSG_USER_INFO_112
113	CC_IDMSG_MODIFY_113
114	CC_IDMSG_DIGIT_114
115	CC_IDMSG_DIGIT_DIAL_115
116	CC_IDMSG_DIGIT_DIAL_STOP_116
117	CC_IDMSG_FEATURE_117
118	CC_IDMSG_FEATURE_ENABLE_118
119	CC_IDMSG_ASSOCIATE_STREAM_119
120	CC_IDMSG_ASSOCIATE_STREAM_120
121	CC_IDMSG_DISASSOCIATE_STREAM_121
122	CC_IDMSG_DISASSOCIATE_STREAM_122
123	CC_IDMSG_GENERATE_TONE_INFO_123
124	CC_IDMSG_SET_DIGIT_TIMEOUTS_124
125	CC_IDMSG_SET_DIGIT_TIMEOUTS_125
126	CC_IDMSG_SUSPEND_126
127	CC_IDMSG_SUSPEND_ACK_127
128	CC_IDMSG_SUSPEND_REJ_128
129	CC_IDMSG_RESUME_129
130	CC_IDMSG_RESUME_ACK_130
131	CC_IDMSG_RESUME_REJ_131
132	CC_IDMSG_UPDATE_REDIRECT_NUM_132

Value	CCAPI Debug Function
133	CC_IDMSG_BABBLER_AUDIT_133
134	CC_IDMSG_CONFERENCE_CREATE_134
135	CC_IDMSG_CONFERENCE_CREATE_135
136	CC_IDMSG_CONFERENCE_CREATE_136
137	CC_IDMSG_CONFERENCE_DESTROY_137
138	CC_IDMSG_CONFERENCE_DESTROY_138
139	CC_IDMSG_CONFERENCE_DESTROY_139
140	CC_IDMSG_LOOPBACK_140
141	CC_IDMSG_COT_TEST_141
142	CC_IDMSG_HANDOFF_142
143	CC_IDMSG_APP_RETURN_143
144	CC_IDMSG_T38_FAX_START_144
145	CC_IDMSG_T38_FAX_DONE_145
146	CC_IDMSG_CALL_PREEMPT_IND_146

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(6)NA2	This command was introduced.
12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.
12.3(8)T	The **all**, **default**, **detail**, **call**, **informational**, **software**, **individual**, **function**, **protoheaders**, and **service** keywords were added.

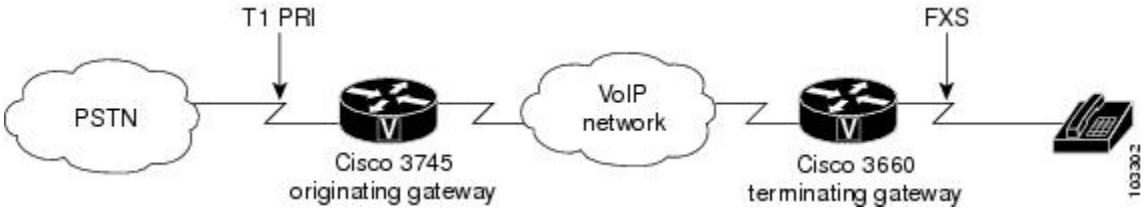
Release	Modification
12.4(4)XC	The range for the **individual** keyword was extended to 146, to include logs for call preemption indication information.
12.4(9)T	This command was integrated into Cisco IOS Release 12.4(9)T.

Examples

The following examples show output for variations of the **debug voip ccapi** command:

For these examples, the topology shown in the figure below is used.

Figure 1: Network Topology for debug voip ccapi Output Examples



Examples

```
Router# debug voip ccapi detail
voip ccapi detail debugging is on
Router#
*Apr 18 20:35:35.779: //-1/ABCE697D8005/CCAPI/cc_api_call_setup_ind_common:
   Interface Type=13, Protocol=0
*Apr 18 20:35:35.779: //-1/ABCE697D8005/CCAPI/ccCheckClipClir:
   Calling Party Number Is User Provided
*Apr 18 20:35:35.779: //11/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry:
   Total Call Count=0, Call Entry(Call Count On=FALSE, Incoming Call=TRUE)
*Apr 18 20:35:35.779: //11/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry:
   Total Call Count=1
```

The following event shows that the CallEntry ID 11 is used for the incoming call leg.

```
*Apr 18 20:35:35.779: //11/ABCE697D8005/CCAPI/cc_insert_guid_pod_entry:
   Incoming=TRUE, Call Id=11
*Apr 18 20:35:35.779: //11/ABCE697D8005/CCAPI/cc_setupind_registration_lookup:
   Matching Parameters; Called Number=83103, Call Transfer Consult Id=
*Apr 18 20:35:35.779: //11/ABCE697D8005/CCAPI/cc_setupind_registration_lookup:
   No Matching Node
*Apr 18 20:35:35.779: //11/ABCE697D8005/CCAPI/ccCheckClipClir:
   Calling Party Number Is User Provided
*Apr 18 20:35:35.779: //12/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry:
   Total Call Count=1, Call Entry(Call Count On=FALSE, Incoming Call=FALSE)
```

The following event shows that the incoming call leg with CallEntry ID 11 is bound to the outgoing call leg with CallEntry ID 12.

```
*Apr 18 20:35:35.779: //11/ABCE697D8005/CCAPI/cc_peer_bind:
   Bind=TRUE, Binder Call Id=11, Bindee Call Id=12
```

The next event shows that CallEntry ID 12 is used for the outgoing call leg.

```
*Apr 18 20:35:35.779: //12/ABCE697D8005/CCAPI/cc_insert_guid_pod_entry:
   Incoming=FALSE, Call Id=12
*Apr 18 20:35:35.779: //-1/xxxxxxxxxxxx/CCAPI/cc_api_supported_data:
   data_mode=0x10082
```

The next event shows an IP address for a remote device on the outgoing call leg, which indicates that this is the VoIP call leg.

```
*Apr 18 20:35:35.779: //12/ABCE697D8005/CCAPI/cc_incr_if_call_volume:
   Remote IP Address=172.16.13.81, Hwidb=FastEthernet0/0
*Apr 18 20:35:35.779: //12/ABCE697D8005/CCAPI/cc_incr_if_call_volume:
   Total Call Count=1, Voip Call Count=1, MMoip Call Count=0
*Apr 18 20:35:35.795: //11/ABCE697D8005/CCAPI/ccCallGetContext:
   Context=0x652C0168, Call Id=11
*Apr 18 20:36:31.419: //11/ABCE697D8005/CCAPI/ccCallDisconnect:
   Start Calling Accounting;
   Call Entry(Incoming=TRUE)
*Apr 18 20:36:31.419: //11/ABCE697D8005/CCAPI/ccCallDisconnect:
   Cause Value=16, Call Entry(Disconnect Cause=16)
*Apr 18 20:36:31.419: //11/ABCE697D8005/CCAPI/ccCallDisconnect:
   Call Entry(Disconnect Cause=16)
```

At this point, the CallEntry ID changes as the call accounting process begins. The accounting data is sent over the outgoing call leg. The GUID, which identifies the unique call, remains the same.

```
*Apr 18 20:36:31.419: //12/ABCE697D8005/CCAPI/ccCallDisconnect:
   Start Calling Accounting;
   Call Entry(Incoming=FALSE)
*Apr 18 20:36:31.419: //12/ABCE697D8005/CCAPI/ccCallDisconnect:
   Cause Value=16, Call Entry(Disconnect Cause=0)
```

The change of the CallEntry ID indicates that the call is using the incoming call leg, which is the POTS call leg in this case.

```
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Data Bitmask=0x1, Call Id=11
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Flag=FALSE
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/cc_delete_guid_pod_entry:
   Incoming=TRUE
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/cc_delete_call_entry:
   ccFreeRawMsgInfo=0x63FF8198
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Total Call Count=1, Call Entry(Call Count On=FALSE, Incoming Call=TRUE)
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Total Call Count=0
*Apr 18 20:36:31.423: //11/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Deleting profileTable[0x64F44700]
```

The next line shows the impairment calculation. This is the only CCAPI debug command that shows impairment.

```
*Apr 18 20:36:31.423: //-1/ABCE697D8005/CCAPI/g113_calculate_impairment:
   (delay=91(ms), loss=0%), Io=0 Iq=0 Idte=0 Idd=2 Ie=10 Itot=12
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Data Bitmask=0x1, Call Id=12
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Flag=FALSE
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/cc_decr_if_call_volume:
   Remote IP Address=172.16.13.81, Hwidb=FastEthernet0/0
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/cc_decr_if_call_volume:
   Total Call Count=0, Voip Call Count=0, MMoip Call Count=0
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/cc_delete_guid_pod_entry:
   Incoming=FALSE
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Total Call Count=0, Call Entry(Call Count On=FALSE, Incoming Call=FALSE)
*Apr 18 20:36:31.423: //12/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Deleting profileTable[0x652E3310]
*Apr 18 20:36:31.427: //12/xxxxxxxxxxxx/CCAPI/cc_get_call_entry:
   Call Entry Is Not Found
```

Examples

```
Router# debug voip ccapi detail
voip ccapi detail debugging is on
Router#
```

```
*May  1 18:58:26.251: //-1/xxxxxxxxxxxx/CCAPI/cc_api_supported_data:
   data_mode=0x10082
*May  1 18:58:26.255: //8/xxxxxxxxxxxx/CCAPI/cc_get_call_entry:
   Call Entry Is Not Found
*May  1 18:58:26.255: //-1/ABCE697D8005/CCAPI/cc_api_call_setup_ind_common:
   Interface Type=0, Protocol=1
*May  1 18:58:26.255: //-1/ABCE697D8005/CCAPI/ccCheckClipClir:
   Calling Party Number Is User Provided
```

The following line shows the attributes of the calling number:

```
*May  1 18:58:26.255: //-1/ABCE697D8005/CCAPI/cc_api_call_setup_ind_common:
   After Number Translation Checking:
   Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed),
   Called Number=3600(TON=Unknown, NPI=Unknown)
*May  1 18:58:26.255: //8/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry:
   Total Call Count=0, Call Entry(Call Count On=FALSE, Incoming Call=TRUE)
*May  1 18:58:26.255: //8/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry:
   Total Call Count=1
*May  1 18:58:26.255: //8/ABCE697D8005/CCAPI/cc_insert_guid_pod_entry:
   Incoming=TRUE, Call Id=8
```

The following line shows the IP address of the originating gateway:

```
*May  1 18:58:26.255: //8/ABCE697D8005/CCAPI/cc_incr_if_call_volume:
   Remote IP Address=172.16.13.175, Hwidb=FastEthernet0/0
*May  1 18:58:26.255: //8/ABCE697D8005/CCAPI/cc_incr_if_call_volume:
   Total Call Count=1, Voip Call Count=1, MMoip Call Count=0
*May  1 18:58:26.255: //8/ABCE697D8005/CCAPI/cc_setupind_registration_lookup:
   Matching Parameters; Called Number=3600, Call Transfer Consult Id=
*May  1 18:58:26.255: //8/ABCE697D8005/CCAPI/cc_setupind_registration_lookup:
   No Matching Node
*May  1 18:58:26.255: //8/ABCE697D8005/CCAPI/ccCheckClipClir:
   Calling Party Number Is User Provided
*May  1 18:58:26.259: //9/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry:
   Total Call Count=1, Call Entry(Call Count On=FALSE, Incoming Call=FALSE)
*May  1 18:58:26.259: //8/ABCE697D8005/CCAPI/cc_peer_bind:
   Bind=TRUE, Binder Call Id=8, Bindee Call Id=9
*May  1 18:58:26.259: //9/ABCE697D8005/CCAPI/cc_insert_guid_pod_entry:
   Incoming=FALSE, Call Id=9
*May  1 18:58:26.259: //9/ABCE697D8005/CCAPI/cc_set_voice_port_value:
   CC_IF_TELEPHONY: Echo=0, Playout=0
*May  1 18:58:26.263: //9/ABCE697D8005/CCAPI/ccCallGetContext:
   Context=0x64B6BB5C, Call Id=9
*May  1 18:59:21.871: //8/ABCE697D8005/CCAPI/ccCallDisconnect:
   Start Calling Accounting;
   Call Entry(Incoming=TRUE)
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/ccCallDisconnect:
   Cause Value=16, Call Entry(Disconnect Cause=16)
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/ccCallDisconnect:
   Call Entry(Disconnect Cause=16)
*May  1 18:59:21.875: //9/ABCE697D8005/CCAPI/ccCallDisconnect:
   Start Calling Accounting;
   Call Entry(Incoming=FALSE)
*May  1 18:59:21.875: //9/ABCE697D8005/CCAPI/ccCallDisconnect:
   Cause Value=16, Call Entry(Disconnect Cause=0)
```

The next line shows the impairment calculation. This is the only CCAPI debug command that shows impairment.

```
*May  1 18:59:21.875: //-1/ABCE697D8005/CCAPI/g113_calculate_impairment:
   (delay=99(ms), loss=0%), Io=0 Iq=0 Idte=0 Idd=2 Ie=10 Itot=12
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Data Bitmask=0x1, Call Id=8
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Flag=FALSE
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/cc_decr_if_call_volume:
   Remote IP Address=172.16.13.175, Hwidb=FastEthernet0/0
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/cc_decr_if_call_volume:
   Total Call Count=0, Voip Call Count=0, MMoip Call Count=0
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/cc_delete_guid_pod_entry:
   Incoming=TRUE
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/cc_delete_call_entry:
```

```
   ccFreeRawMsgInfo=0x644EB850
Router#
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Total Call Count=1, Call Entry(Call Count On=FALSE, Incoming Call=TRUE)
*May  1 18:59:21.875: //8/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Total Call Count=0
*May  1 18:59:21.879: //8/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Deleting profileTable[0x64B78600]
*May  1 18:59:21.879: //8/xxxxxxxxxxxx/CCAPI/cc_get_call_entry:
   Call Entry Is Not Found
*May  1 18:59:21.879: //8/xxxxxxxxxxxx/CCAPI/cc_get_call_entry:
   Call Entry Is Not Found
Router#
*May  1 18:59:24.587: //9/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Data Bitmask=0x1, Call Id=9
*May  1 18:59:24.587: //9/ABCE697D8005/CCAPI/ccCallGetVoipFlag:
   Flag=FALSE
*May  1 18:59:24.587: //9/ABCE697D8005/CCAPI/cc_api_call_disconnect_done:
   Prefix Is Not Defined From Peer; Peer=3600, Called Number=3600
*May  1 18:59:24.587: //9/ABCE697D8005/CCAPI/cc_delete_guid_pod_entry:
   Incoming=FALSE
*May  1 18:59:24.587: //9/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Total Call Count=0, Call Entry(Call Count On=FALSE, Incoming Call=FALSE)
*May  1 18:59:24.587: //9/ABCE697D8005/CCAPI/cc_delete_call_entry:
   Deleting profileTable[0x6453F228]
```

Examples

```
Router# debug voip ccapi inout
voip ccapi inout debugging is on
Router#
*Apr 18 20:42:19.347: //-1/9C5A9CA88009/CCAPI/cc_api_display_ie_subfields:
   cc_api_call_setup_ind_common:
   acme-username=
   ----- ccCallInfo IE subfields -----
   acme-ani=4085550111
   acme-anitype=2
   acme-aniplan=1
   acme-anipi=0
   acme-anisi=1
   dest=83103
   acme-desttype=0
   acme-destplan=0
   acme-rdn=
   acme-rdntype=-1
   acme-rdnplan=-1
   acme-rdnpi=-1
   acme-rdnsi=-1
   acme-redirectreason=-1
```

The following lines show information about the calling and called numbers. The network presentation indicator (NPI) shows the type of transmission. The Incoming Dial-Peer field shows that the incoming dial peer has been matched.

```
*Apr 18 20:42:19.347: //-1/9C5A9CA88009/CCAPI/cc_api_call_setup_ind_common:
   Interface=0x64F26F10, Call Info(
   Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed),
   Called Number=83103(TON=Unknown, NPI=Unknown),
   Calling Translated=FALSE, Subsriber Type Str=RegularLine, FinalDestinationFlag=TRUE,
   Incoming Dial-peer=1, Progress Indication=NULL(0), Calling IE Present=TRUE,
   Source Trkgrp Route Label=, Target Trkgrp Route Label=, CLID Transparent=FALSE), Call
Id=-1
*Apr 18 20:42:19.347: //-1/9C5A9CA88009/CCAPI/ccCheckClipClir:
   In: Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed)
*Apr 18 20:42:19.347: //-1/9C5A9CA88009/CCAPI/ccCheckClipClir:
   Out: Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed)
```

In the following event, the call leg is established. The CallEntry ID field changes from-1 to 19.

```
*Apr 18 20:42:19.347: //19/9C5A9CA88009/CCAPI/cc_api_call_setup_ind_common:
   Set Up Event Sent;
   Call Info(Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed),
   Called Number=83103(TON=Unknown, NPI=Unknown))
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/cc_process_call_setup_ind:
   Event=0x63FF4730
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCallSetContext:
   Context=0x652A9858
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/cc_process_call_setup_ind:
   >>>>CCAPI handed cid 19 with tag 1 to app "Default"
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCallProceeding:
   Progress Indication=NULL(0)
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCallSetupRequest:
   Destination=, Calling IE Present=TRUE, Mode=0,
   Outgoing Dial-peer=3600, Params=0x652AA4A8, Progress Indication=NULL(0)
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCheckClipClir:
   In: Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed)
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCheckClipClir:
   Out: Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed)
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCallSetupRequest:
   Destination Pattern=360., Called Number=3600, Digit Strip=FALSE
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccCallSetupRequest:
   Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed),
   Called Number=3600(TON=Unknown, NPI=Unknown),
   Redirect Number=, Display Info=
   Account Number=, Final Destination Flag=TRUE,
   Guid=9C5A9CA8-5243-11D6-8009-00059A3A15A0, Outgoing Dial-peer=3600
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/cc_api_display_ie_subfields:
   ccCallSetupRequest:
   cisco-username=
   ----- ccCallInfo IE subfields -----
   cisco-ani=4085550111
   cisco-anitype=2
   cisco-aniplan=1
   cisco-anipi=0
   cisco-anisi=1
   dest=3600
   cisco-desttype=0
   cisco-destplan=0
   cisco-rdn=
   cisco-rdntype=-1
   cisco-rdnplan=-1
   cisco-rdnpi=-1
   cisco-rdnsi=-1
   cisco-redirectreason=-1
```

In the following lines, the outgoing dial peer is matched:

```
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccIFCallSetupRequestPrivate:
   Interface=0x63EAF24C, Interface Type=1, Destination=, Mode=0x0,
   Call Params(Calling Number=4085550111(TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed),
   Called Number=3600(TON=Unknown, NPI=Unknown), Calling Translated=FALSE,
  Subsriber Type Str=RegularLine, FinalDestinationFlag=TRUE, Outgoing Dial-peer=3600, Call
 Count On=FALSE,
   Source Trkgrp Route Label=, Target Trkgrp Route Label=, tg_label_flag=0, Application
Call Id=)
*Apr 18 20:42:19.351: //20/9C5A9CA88009/CCAPI/ccIFCallSetupRequestPrivate:
   SPI Call Setup Request Is Success; Interface Type=1, FlowMode=1
*Apr 18 20:42:19.351: //20/9C5A9CA88009/CCAPI/ccCallSetContext:
   Context=0x652AA458
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccSaveDialpeerTag:
   Outgoing Dial-peer=3600
*Apr 18 20:42:19.351: //19/9C5A9CA88009/CCAPI/ccSaveDialpeerTag:
   Outgoing Dial-peer=3600
```

```
*Apr 18 20:42:19.367: //20/9C5A9CA88009/CCAPI/cc_api_call_proceeding:
   Interface=0x652F6388, Progress Indication=NULL(0)
```

The following lines show call progress. The progress and signal indications are shown.

```
*Apr 18 20:42:19.371: //20/9C5A9CA88009/CCAPI/cc_api_call_cut_progress:
   Interface=0x652F6388, Progress Indication=INBAND(8), Signal Indication=SIGNAL RINGBACK(1),

   Cause Value=0
*Apr 18 20:42:19.371: //20/9C5A9CA88009/CCAPI/cc_api_call_cut_progress:
   Call Entry(Responsed=TRUE)
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/ccCallCutProgress:
   Progress Indication=INBAND(8), Signal Indication=SIGNAL RINGBACK(1), Cause Value=0
   Voice Call Send Alert=FALSE, Call Entry(AlertSent=FALSE)
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/ccCallCutProgress:
   Call Entry(Responsed=TRUE)
```

The following lines show the tone generation information:

```
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/ccGenerateToneInfo:
   Stop Tone On Digit=FALSE, Tone=Null,
   Tone Direction=Network, Params=0x0, Call Id=19
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/ccConferenceCreate:
   Conference Id=0x652F723C, Call Id1=19, Call Id2=20, Tag=0x0
*Apr 18 20:42:19.371: //20/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done:
   Conference Id=0x6, Source Interface=0x63EAF24C, Source Call Id=20,
   Destination Call Id=19, Disposition=0x0, Tag=0x0
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/ccConferenceCreate:
   Call Entry(Conference Id=0x6, Destination Call Id=20)
*Apr 18 20:42:19.371: //20/9C5A9CA88009/CCAPI/ccConferenceCreate:
   Call Entry(Conference Id=0x6, Destination Call Id=19)
*Apr 18 20:42:19.371: //19/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done:
   Conference Id=0x6, Source Interface=0x64F26F10, Source Call Id=19,
   Destination Call Id=20, Disposition=0x0, Tag=0x0
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/cc_generic_bridge_done:
   Conference Id=0x6, Source Interface=0x64F26F10, Source Call Id=19,
   Destination Call Id=20, Disposition=0x0, Tag=0x0
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/cc_api_caps_ind:
   Destination Interface=0x63EAF24C, Destination Call Id=20, Source Call Id=19,
   Caps(Codec=0x2887F, Fax Rate=0xBF, Vad=0x3,
   Modem=0x2, Codec Bytes=0, Signal Type=3)
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/cc_api_caps_ind:
   Caps(Playout Mode=1, Playout Initial=60(ms), Playout Min=40(ms),
   Playout Max=300(ms), Fax Nom=300(ms))
*Apr 18 20:42:19.371: //19/9C5A9CA88009/CCAPI/cc_process_notify_bridge_done:
   Conference Id=0x6, Call Id1=19, Call Id2=20
*Apr 18 20:42:19.375: //20/9C5A9CA88009/CCAPI/cc_api_caps_ind:
   Destination Interface=0x64F26F10, Destination Call Id=19, Source Call Id=20,
   Caps(Codec=0x4, Fax Rate=0x1, Vad=0x2,
   Modem=0x2, Codec Bytes=20, Signal Type=2)
*Apr 18 20:42:19.375: //20/9C5A9CA88009/CCAPI/cc_api_caps_ind:
   Caps(Playout Mode=1, Playout Initial=60(ms), Playout Min=40(ms),
   Playout Max=300(ms), Fax Nom=300(ms))
```

The following lines show codec information:

```
*Apr 18 20:42:19.375: //20/9C5A9CA88009/CCAPI/cc_api_caps_ack:
   Destination Interface=0x64F26F10, Destination Call Id=19, Source Call Id=20,
   Caps(Codec=g729r8(0x4), Fax Rate=FAX_RATE_NONE(0x1), Vad=ON(0x2),
   Modem=ON(0x2), Codec Bytes=20, Signal Type=2, Seq Num Start=6872)
*Apr 18 20:42:19.375: //19/9C5A9CA88009/CCAPI/cc_api_caps_ack:
   Destination Interface=0x63EAF24C, Destination Call Id=20, Source Call Id=19,
   Caps(Codec=g729r8(0x4), Fax Rate=FAX_RATE_NONE(0x1), Vad=ON(0x2),
   Modem=ON(0x2), Codec Bytes=20, Signal Type=2, Seq Num Start=6872)
*Apr 18 20:42:19.375: //19/9C5A9CA88009/CCAPI/cc_api_voice_mode_event:
   Call Id=19
*Apr 18 20:42:19.375: //19/9C5A9CA88009/CCAPI/cc_api_voice_mode_event:
   Call Entry(Context=0x652A9858)
```

The following lines show progress indication information. In this case, the event shows that the destination is not ISDN.

```
*Apr 18 20:42:26.855: //20/9C5A9CA88009/CCAPI/cc_api_call_connected:
   Interface=0x652F6388, Data Bitmask=0x0, Progress Indication=DESTINATION IS NON ISDN(2),
   Connection Handle=0
*Apr 18 20:42:26.855: //20/9C5A9CA88009/CCAPI/cc_api_call_connected:
   Call Entry(Connected=TRUE, Responsed=TRUE, Retry Count=0)
*Apr 18 20:42:26.855: //19/9C5A9CA88009/CCAPI/ccCallConnect:
   Progress Indication=DESTINATION IS NON ISDN(2), Data Bitmask=0x0
*Apr 18 20:42:26.855: //19/9C5A9CA88009/CCAPI/ccCallConnect:
   Call Entry(Connected=TRUE, Responsed=TRUE)
*Apr 18 20:42:26.855: //20/9C5A9CA88009/CCAPI/ccSaveDialpeerTag:
   Incoming Dial-peer=1
*Apr 18 20:42:26.859: //19/9C5A9CA88009/CCAPI/ccSaveDialpeerTag:
   Outgoing Dial-peer=3600
*Apr 18 20:42:26.859: //20/9C5A9CA88009/CCAPI/ccCallFeature:
   Feature Type=24, Call Id=20
```

This event shows that the call is disconnected.

```
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/cc_api_call_disconnected:
   Cause Value=16, Interface=0x64F26F10, Call Id=19
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/cc_api_call_disconnected:
   Call Entry(Responsed=TRUE, Cause Value=16, Retry Count=0)
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/ccConferenceDestroy:
   Conference Id=0x6, Tag=0x0
*Apr 18 20:43:16.795: //20/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done:
   Conference Id=0x6, Source Interface=0x63EAF24C, Source Call Id=20,
   Destination Call Id=19, Disposition=0x0, Tag=0x0
*Apr 18 20:43:16.795: //19/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done:
   Conference Id=0x6, Source Interface=0x64F26F10, Source Call Id=19,
   Destination Call Id=20, Disposition=0x0, Tag=0x0
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/cc_generic_bridge_done:
   Conference Id=0x6, Source Interface=0x64F26F10, Source Call Id=19,
   Destination Call Id=20, Disposition=0x0, Tag=0x0
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/ccCallDisconnect:
   Cause Value=16, Tag=0x0, Call Entry(Previous Disconnect Cause=0, Disconnect Cause=16)
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/ccCallDisconnect:
   Cause Value=16, Call Entry(Responsed=TRUE, Cause Value=16)
*Apr 18 20:43:16.795: //19/9C5A9CA88009/CCAPI/cc_api_get_transfer_info:
   Transfer Number Is Null
*Apr 18 20:43:16.795: //20/9C5A9CA88009/CCAPI/ccCallDisconnect:
   Cause Value=16, Tag=0x0, Call Entry(Previous Disconnect Cause=0, Disconnect Cause=0)
*Apr 18 20:43:16.795: //20/9C5A9CA88009/CCAPI/ccCallDisconnect:
   Cause Value=16, Call Entry(Responsed=TRUE, Cause Value=16)
*Apr 18 20:43:16.795: //20/9C5A9CA88009/CCAPI/cc_api_get_transfer_info:
   Transfer Number Is Null
*Apr 18 20:43:16.803: //20/9C5A9CA88009/CCAPI/cc_api_call_disconnect_done:
   Disposition=0, Interface=0x652F6388, Tag=0x0, Call Id=20,
   Call Entry(Disconnect Cause=16, Voice Class Cause Code=0, Retry Count=0)
*Apr 18 20:43:16.803: //20/9C5A9CA88009/CCAPI/cc_api_call_disconnect_done:
   Call Disconnect Event Sent
*Apr 18 20:43:16.803: //19/9C5A9CA88009/CCAPI/cc_api_call_disconnect_done:
   Disposition=0, Interface=0x64F26F10, Tag=0x0, Call Id=19,
   Call Entry(Disconnect Cause=16, Voice Class Cause Code=0, Retry Count=0)
*Apr 18 20:43:16.803: //19/9C5A9CA88009/CCAPI/cc_api_call_disconnect_done:
   Call Disconnect Event Sent
```

Examples

```
Router# debug voip ccapi service

voip ccapi service debugging is on
*May  1 19:08:41.803: //-1/xxxxxxxxxxxx/CCAPI/cc_setupind_match_search:
   Searching Node;
   Called Number=3600, Call Transfer Consult Id=
```

This debug shows noncall related events. In this case, information about the timer is shown.

```
*May  1 19:08:48.027: //-1/xxxxxxxxxxxx/CCAPI/cc_handle_periodic_timer:
```

```
   Calling The Callback, ccTimerctx=0x63B368C0
*May  1 19:08:48.027: //-1/xxxxxxxxxxxx/CCAPI/ccTimerStart:
   ccTimerctx=0x63B368C0
*May  1 19:10:08.615: //-1/xxxxxxxxxxxx/CCAPI/cc_api_icpif:
   ExpectFactor=0xA
```

Examples

This debug shows the preemption tone timer getting initiated.

```
*Aug 24 18:28:16.919: //18958/B37648B6AF48/CCAPI/cc_api_call_preempt_ind:
   PreemptionToneTimer=10(s)
```

debug voip ccapi error

To trace error logs in the call control application programming interface (CCAPI), use the **debug voip ccapi error**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip ccapi error

no debug voip ccapi error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

Usage Guidelines

The **debug voip ccapi error** command traces the error logs in the call control API. Error logs are generated during normal call processing, when there are insufficient resources, or when there are problems in the underlying network-specific code, the higher call session application, or the call control API itself.

This **debug** command shows error events or unexpected behavior in system software. In most cases, no events will be generated.

Note We recommend that you log output from the **debug voip ccapi error**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.

Related Commands

Command	Description
debug voip ccapi inout	Traces the execution path through the CCAPI.
voice call debug	Allows configuration of the voice call debug output.

debug voip ccapi inout

To trace the execution path through the call control application programming interface (CCAPI), use the **debug voip ccapi inout command**in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip ccapi inout

no debug voip ccapi inout

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(11)T	This command was implemented on the following platforms: Cisco 2600 series, Cisco 3620, Cisco 3660, Cisco AS5350, Cisco AS5400, Cisco AS5850, Cisco AS5300, Cisco AS5800, and Cisco MC3810.

Usage Guidelines

The **debug voip ccapi inout** command traces the execution path through the call control API, which serves as the interface between the call session application and the underlying network-specific software. You can use the output from this command to understand how calls are being handled by the voice gateway.

This command shows how a call flows through the system. Using this debug level, you can see the call setup and teardown operations performed on both the telephony and network call legs.

Note We recommend that you log output from the **debug voip ccapi inout**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 the call setup indicated and accepted by the voice gateway:

```
Router# debug voip ccapi inout
*Mar  1 15:35:53.588: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructTDUsrContainer:
usrContainer[0x638C1BF0], magic[FACE0FFF]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDUtilAddDataToUsrContainer:
container=0x638C1BF0, tagID=6, dataSize=16, instID=-1,modifier=1
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructInstanceTDObject:
tdObject[0x638BC1AC], nxtElem[0x0], magic[0xFACE0FFF] tagID[6], dataLen[16],
modif[1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer:
Adding tdObject[0x638BC1AC] instID[-1] into container[0x638C1BF0]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDUtilAddDataToUsrContainer:
container=0x638C1BF0, tagID=5, dataSize=276, instID=-1,modifier=1
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructInstanceTDObject:
```

```
tdObject[0x63401148], nxtElem[0x0], magic[0xFACE0FFF] tagID[5], dataLen[276],
modif[1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer:
Adding tdObject[0x63401148] instID[-1] into container[0x638C1BF0]
```

In the following lines, the call control API (CCAPI) receives the call setup. The called number is 34999, and the calling number is 55555. The calling number matches dial peer 10002.

```
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/cc_api_display_ie_subfields:
*Mar  1 15:35:53.592: cc_api_call_setup_ind:
*Mar  1 15:35:53.592:  cisco-username=
*Mar  1 15:35:53.596: ----- ccCallInfo IE subfields -----
*Mar  1 15:35:53.596:  cisco-ani=55555
*Mar  1 15:35:53.596:  cisco-anitype=0
*Mar  1 15:35:53.596:  cisco-aniplan=0
*Mar  1 15:35:53.596:  cisco-anipi=0
*Mar  1 15:35:53.596:  cisco-anisi=0
*Mar  1 15:35:53.596:  dest=34999
*Mar  1 15:35:53.596:  cisco-desttype=0
*Mar  1 15:35:53.596:  cisco-destplan=0
*Mar  1 15:35:53.596:  cisco-rdn=
*Mar  1 15:35:53.596:  cisco-rdntype=-1
*Mar  1 15:35:53.596:  cisco-rdnplan=-1
*Mar  1 15:35:53.596:  cisco-rdnpi=-1
*Mar  1 15:35:53.596:  cisco-rdnsi=-1
*Mar  1 15:35:53.596:  cisco-redirectreason=-1
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind:
(vdbPtr=0x637EC1E0,
callInfo={called=34999,called_oct3=0x80,calling=55555,calling_oct3=0x80,calling_oct3a=0x0,calling_xlated=false,subscriber_type_str=RegularLine,fdest=1,
peer_tag=10002, prog_ind=0,callingIE_present 1, src_route_label=, tgt_route_label=
clid_transparent=0},callID=0x637B4278)
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind:
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind: type 13 , prot 0
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: ccCheckClipClir: calling number is: "55555", calling oct3a is: 0x0
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: Calling Party number is User Provided
*Mar  1 15:35:53.596: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.596: Leaving ccCheckClipClir
  calling number is: "55555"
  calling oct3 is:  0x80
  calling oct3a is: 0x0
```

In the next line, 44 is the CallEntry ID.

```
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: Increment call volume:
 0
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: current call volume: 1
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: entry's incoming TRUE.
*Mar  1 15:35:53.600: //44/xxxxxxxxxxxx/CCAPI/cc_insert_call_entry: is_incoming is TRUE
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDConstructHashProfileTab:
profileTable[0x6380E11C], numBuckets[11], numEntries[0]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager: Invoking
necessary profileTable updaters...
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer: Updating
 profileTable[0x6380E11C] with objects in container[0x638C1BF0]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer: obtained
 key[5] for the tag[6]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket:
profileTable[0x6380E11C], tdObject[0x638BC1AC]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer: obtained
 key[0] for the tag[5]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket:
profileTable[0x6380E11C], tdObject[0x63401148]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager:
*Mar  1 15:35:53.600: ccTDUtilDumpAllElemInProfileTab: profileTable[0x6380E11C],
 numBuckets[11], numEntries[2]
*Mar  1 15:35:53.600: Bucket { 0 } ------>0x63401148[0x0,t-5,l-276,d-0x63401168,
m-1,u-56153,g-FACE0FFF]
*Mar  1 15:35:53.604:
*Mar  1 15:35:53.604: Bucket { 5 }
------>0x638BC1AC[0x0,t-6,l-16,d-0x638BC1CC,m-1,u-56153,g-FACE0FFF]
```

```
*Mar  1 15:35:53.604:
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructTDUsrContainer:
Container[0x638C1BF0]
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: not the VoIP or MMoIP
*Mar  1 15:35:53.608: //-1/xxxxxxxxxxxx/CCAPI/cc_process_call_setup_ind: (event=
0x63073AA0)
```

In the next line, 45F2AAE28044 is the GUID. The tag 10002 entry shows that the incoming dial peer matched the CallEntry ID.

```
*Mar  1 15:35:53.608: //44/45F2AAE28044/CCAPI/cc_process_call_setup_ind: >>>>CCAPI handed
cid 44 with tag 10002 to app "DEFAULT"
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(24=CC_EV_CALL_SETUP_IND),
 cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(SSA_EV_CALL_SETUP_IND),
cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/SSAPP:-1:-1/ssaCallSetupInd:
```

The next line shows CallEntry ID in hexadecimal form, 0x2C (44 in decimal). The CallID and GUID numbers have been identified. The incoming dial-peer is 10002.

```
*Mar  1 15:35:53.608: //44/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2C,
context=0x634A430C)
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: cid(44),
st(SSA_CS_MAPPING),oldst(0), ev(24)ev->e.evCallSetupInd.nCallInfo.finalDestFlag
 = 1
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: src route label=,
tgt route label= tg_label_flag 0x0
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: finalDest
cllng(55555), clled(34999) tgt_route_label()tg_label_flag 0x0
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: cid(44),
 st(SSA_CS_CALL_SETTING),oldst(0), ev(24)dpMatchPeersMoreArg result= 0
```

For CallEntry ID 44, two dial-peer tags (10001 and 20002) were matched with called number 34999.

```
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaDebugPeers: ssaSetupPe
er cid(44) peer list: tag(10001) called number (34999) tag(20002) called number
(34999)
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaSetupPeer: dialpeer tags in rotary=
 10001  20002
```

The next line shows that 5 digits were matched for this dial peer and no prefix was added. The encapType (2) entry indicates a VoIP call.

```
*Mar  1 15:35:53.612: //44/45F2AAE28044/SSAPP:10002:-1/ssaSetupPeer: cid(44), de
stPat(34999), matched(5), prefix(), peer(637B0984), peer->encapType (2)
*Mar  1 15:35:53.612: //-1/xxxxxxxxxxxx/CCAPI/cc_can_gateway: Call legs: In=6, O
ut=1
```

The next line shows the voice gateway sending out a call-proceeding message to the incoming call leg with progress indicator of 0x0.

```
*Mar  1 15:35:53.612: //44/xxxxxxxxxxxx/CCAPI/ccCallProceeding: (callID=0x2C, pr
og_ind=0x0)
```

The next line shows the voice gateway sending out the call-setup request to the outgoing call leg. The dial-peer is 10001 with the incoming CallEntry ID being 0x2C.

```
*Mar  1 15:35:53.612: //44/xxxxxxxxxxxx/CCAPI/ccCallSetupRequest: (Inbound call
= 0x2C, outbound peer =10001, dest=,
        params=0x63085D80 mode=0, *callID=0x63086314, prog_ind = 0callingIE_pres
ent 1)
*Mar  1 15:35:53.612: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.612: ccCallSetupRequest numbering_type 0x80
*Mar  1 15:35:53.612: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.616: ccCallSetupRequest: calling number is:55555
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: calling oct3a
is:0x0
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: ccCheckClipClir: calling number is: "55555", calling oct3a
 is: 0x0
```

```
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: Calling Party number is User Provided
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/ccCheckClipClir:
*Mar  1 15:35:53.616: Leaving ccCheckClipClir
  calling number is: "55555"
  calling oct3 is:  0x80
  calling oct3a is: 0x0
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: after ccCheckC
lipClir - calling oct3a is:0x0
```

The next line shows that all digits are passed.

```
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest: dest pattern 3
4999, called 34999, digit_strip 0
*Mar  1 15:35:53.616: //44/45F2AAE28044/CCAPI/ccCallSetupRequest:
*Mar  1 15:35:53.616: callingNumber=55555, calledNumber=34999, redirectNumber= d
isplay_info= calling_oct3a=0
*Mar  1 15:35:53.616: accountNumber=, finalDestFlag=1,
guid=45f2.aae2.1571.11cc.8044.95f5.fabb.6b0f
*Mar  1 15:35:53.616: peer_tag=10001
*Mar  1 15:35:53.616: //-1/xxxxxxxxxxxx/CCAPI/cc_api_display_ie_subfields:
*Mar  1 15:35:53.616: ccCallSetupRequest:
*Mar  1 15:35:53.616:  cisco-username=
*Mar  1 15:35:53.616: ----- ccCallInfo IE subfields -----
*Mar  1 15:35:53.616:  cisco-ani=55555
*Mar  1 15:35:53.616:  cisco-anitype=0
*Mar  1 15:35:53.616:  cisco-aniplan=0
*Mar  1 15:35:53.616:  cisco-anipi=0
*Mar  1 15:35:53.616:  cisco-anisi=0
*Mar  1 15:35:53.620:  dest=34999
*Mar  1 15:35:53.620:  cisco-desttype=0
*Mar  1 15:35:53.620:  cisco-destplan=0
*Mar  1 15:35:53.620:  cisco-rdn=
*Mar  1 15:35:53.620:  cisco-rdntype=-1
*Mar  1 15:35:53.620:  cisco-rdnplan=-1
*Mar  1 15:35:53.620:  cisco-rdnpi=-1
*Mar  1 15:35:53.620:  cisco-rdnsi=-1
*Mar  1 15:35:53.620:  cisco-redirectreason=-1
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate: (vdbP
tr=0x62EC61A4, dest=, callParams={called=34999,called_oct3=0x80,
calling=55555,calling_oct3=0x80, calling_oct3a= 0x0, calling_xlated=false,
subscriber_type_str
=RegularLine, fdest=1, voice_peer_tag=10001},mode=0x0)
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 15:35:53.620: ccIFCallSetupRequestPrivate: src route label  tgt route label
tg_label_flag 0x0
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:  vdbPtr type =
1
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate: (vdbPtr=0x62EC61A4,
 dest=, callParams={called=34999, called_oct3 0x80,  calling=55555,calling_oct3 0x80,
calling_oct3a 0x0, calling_xlated=false,  fdest=1, voice_pee
r_tag=10001}, mode=0x0, xltrc=-5)
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
In the next line, outgoing CallEntry ID 45 is bound to the same GUID 45F2AAE28044.
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: not incoming
 entry
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: entry's incoming FALSE.
*Mar  1 15:35:53.620: //45/45F2AAE28044/CCAPI/cc_insert_call_entry: is_incoming
is FALSE
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/ccSaveDialpeerTag: (callID=0x2C,
dialpeer_tag=10001)
*Mar  1 15:35:53.624: //45/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2D,
context=0x634A537C) 0x2D (decimal 45 is the second call leg ID).
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/ccCallReportDigits: (callID=0x2C,
enable=0x0)
```

The voice gateway informs the incoming call leg that digits were forwarded.

```
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/CCAPI/cc_api_call_report_digits_done:
(vdbPtr=0x637EC1E0, callID=0x2C, disp=0)
*Mar  1 15:35:53.624: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(54=CC_EV_CALL_
REPORT_DIGITS_DONE), cid(44), disp(0)
```

```
*Mar  1 15:35:53.624: //44/45F2AAE28044/SS
Router#APP:10002:-1/ssaTraceSct: cid(44)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_RE
PORT_DIGITS_DONE)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(1)fDest(1)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaTraceSct: -cid2(45)st2
(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaDebugPeers: ssaReportDigitsDone
cid(44) peer list: tag(20002) called number (34999)
*Mar  1 15:35:53.624: //44/45F2AAE28044/SSAPP:10002:-1/ssaReportDigitsDone: call
id=44 Reporting disabled.
*Mar  1 15:35:53.628: //-1/xxxxxxxxxxxx/CCAPI/cc_api_supported_data: data_mode=0x10082
*Mar  1 15:35:53.628: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_ic_leg_obtained_numbers: callID=0x2D
```

The next two lines shows the IP address of the terminating gateway and that the terminating gateway is reached through Ethernet port 0/0.

```
*Mar  1 15:35:53.628: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: remote IP
is 171.69.85.111
*Mar  1 15:35:53.632: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: hwidb is Ethernet0/0

*Mar  1 15:35:53.632: //-1/xxxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: create entry in list:
 1
*Mar  1 15:35:53.636: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetInstanceCount: For tagI
D[1] of callID[45]
*Mar  1 15:35:53.636: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessManager: No
profileTable set for callID[45]
*Mar  1 15:35:53.636: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetInstanceCount: For tagID[2] of
callID[45]
*Mar  1 15:35:53.636: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessManager: No
profileTable set for callID[45]
```

The next line shows that the voice gateway received a call proceeding message from the terminating gateway, and then the following line shows that the voice gateway received a call alert from the terminating gateway.

```
*Mar  1 15:35:53.740: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_proceeding: (vdbPtr=0x62EC61A4,
callID=0x2D,
      prog_ind=0x0)
*Mar  1 15:35:53.740: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_alert: (vdbPtr=0x62EC61A4,
callID=0x2D, prog_ind=0x0, sig_ind=0x1)
*Mar  1 15:35:53.744: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(21=CC_EV_CALL_PROCEEDING),
 cid(45), disp(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_CALL_SETTING)ev(SSA_EV_CALL_PROCEEDING)
oldst(SSA_CS_MAPPING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct:
-cid2(44)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaCallProc:
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaIgnore: cid(45), st(SSA_CS
_CALL_SETTING),oldst(1), ev(21)
*Mar  1 15:35:53.744: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(7=CC_EV_CALL_ALERT),
cid(45), disp(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_CALL_SETTING)ev(SSA_EV_CALL_ALERT)
oldst(SSA_CS_CALL_SETTING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:35:53.744: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:35:53.744: //44/45F2AAE28044/SSAPP:10002:-1/ssaAlert:
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
Router#
```

The voice gateway forwarded a call alert to the originating gateway.

```
*Mar  1 15:35:53.744: //44/xxxxxxxxxxxx/CCAPI/ccCallAlert: (callID=0x2C, prog_ind=0x0,
sig_ind=0x1)
Router#
```

The phone is answered at the called number.

```
Router#!call answered
Router#
```

The voice gateway receives a connect message from the terminating gateway.

```
*Mar  1 15:36:05.016: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_connected: (vdbPtr=0x6
2EC61A4, callID=0x2D), prog_ind = 0
*Mar  1 15:36:05.016: //45/45F2AAE28044/CCAPI/cc_api_call_connected: setting
callEntry->connected to TRUE
```

The next line shows that the call accounting starts. The leg_type=False message means this is for an outgoing call. The line that follows shows that AAA accounting is not configured.

```
*Mar  1 15:36:05.016: //45/45F2AAE28044/CCAPI/cc_api_call_connected: calling accounting
start for callID=45 leg_type=0
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/CCAPI/ccCallSetAAA_Accounting: callID=0x
2D, accounting=0
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(8=CC_EV_CALL_CONNECTED),
 cid(45), disp(0)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_ALERT_RCVD)ev(SSA_EV_CALL_CONNECTED)
oldst(SSA_CS_CALL_SETTING)cfid(-1)csize(0)in(0)fDest(0)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_ALERT_RCVD)oldst2(SSA_CS_CALL_SETTING)
*Mar  1 15:36:05.020: //45/45F2AAE28044/SSAPP:0:-1/ssaConnect:
*Mar  1 15:36:05.020: //44/xxxxxxxxxxxx/CCAPI/ccGetDialpeerTag: (callID=0x2C)
```

The next lines show a conference being set up between the two call legs 0x2C and 0x2D. Bridge complete messages are sent to both the terminating and originating gateways.

```
*Mar  1 15:36:05.020: //44/xxxxxxxxxxxx/CCAPI/ccConferenceCreate: (confID=0x6308
6424, callID1=0x2C, callID2=0x2D, tag=0x0)
*Mar  1 15:36:05.020: //45/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done: (confID=0x15,
srcIF=0x62EC61A4, srcCallID=0x2D, dstCallID=0x2C, disposition=0, tag=0x0)
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_bridge_done: (confID=0x15,
srcIF=0x637EC1E0, srcCallID=0x2C, dstCallID=0x2D, disposition=0, tag=0x0)
```

Here, the voice gateway sets up negotiating capability with the originating telephony leg.

```
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (dstVdbPtr=0x62EC
61A4, dstCallId=0x2D, srcCallId=0x2C,
     caps={codec=0x2887F, fax_rate=0xBF, vad=0x3, modem=0x2
           codec_bytes=0, signal_type=3})
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (Playout: mode 0,
 initial 60,min 40, max 300)
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(29=CC_EV_CONF_
CREATE_DONE), cid(44), disp(0)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
cid(44)st(SSA_CS_CONFERENCING)ev(SSA_EV_CONF_CREATE_DONE)
oldst(SSA_CS_CALL_SETTING)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_CONFERENCING)oldst2(SSA_CS_ALERT_RCVD)
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaConfCreateDone:
*Mar  1 15:36:05.024: //44/xxxxxxxxxxxx/CCAPI/ccCallConnect: (callID=0x2C), prog
_ind = 0
*Mar  1 15:36:05.024: //44/45F2AAE28044/CCAPI/ccCallConnect: setting callEntry->
connected to TRUE
*Mar  1 15:36:05.024: //44/45F2AAE28044/SSAPP:10002:21/ssaDebugPeers: ssaFlushPeerTagQueue
 cid(44) peer list: tag(20002) called number (34999)
*Mar  1 15:36:05.028: //-1/xxxxxxxxxxxx/CCAPI/cc_process_notify_bridge_done:
(event=0x63067FC0)
```

The voice gateway sets up negotiating capability with the terminating VoIP leg.

```
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ind: (dstVdbPtr=0x637E
C1E0, dstCallId=0x2C, srcCallId=0x2D,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2})
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/
cc_api_caps_ind:
 (Playout: mode 0,
 initial 60,min 40, max 300)
```

The capabilities are acknowledged for both call legs.

```
*Mar  1 15:36:05.028: //45/xxxxxxxxxxxx/CCAPI/cc_api_caps_ack: (dstVdbPtr=0x637E
C1E0, dstCallId=0x2C, srcCallId=0x2D,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2, seq_num_start=2944})
*Mar  1 15:36:05.028: //44/xxxxxxxxxxxx/CCAPI/cc_api_caps_ack: (dstVdbPtr=0x62EC
61A4, dstCallId=0x2D, srcCallId=0x2C,
     caps={codec=0x4, fax_rate=0x2, vad=0x2, modem=0x0
           codec_bytes=20, signal_type=2, seq_num_start=2944})
*Mar  1 15:36:05.032: //44/xxxxxxxxxxxx/CCAPI/cc_api_voice_mode_event: callID=0x2C
*Mar  1 15:36:05.032: //44/45F2AAE28044/CCAPI/cc_api_voice_mode_event: Call Pointer =634A430C
*Mar  1 15:36:05.032: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(52=CC_EV_VOICE
_MODE_DONE), cid(44), disp(0)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
Router#
Router# cid(44)st(SSA_CS_ACTIVE)ev(SSA_EV_VOICE_MODE_DONE)
oldst(SSA_CS_CONFERENCING)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_ACTIVE)oldst2(SSA_CS_ALERT_RCVD)
*Mar  1 15:36:05.032: //44/45F2AAE28044/SSAPP:10002:21/ssaIgnore: cid(44), st(SS
A_CS_ACTIVE),oldst(5), ev(52)
Router#
Router#! digit punched
Router#
```

The phone at the terminating gateway enters digit1.

```
*Mar  1 15:36:11.204: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPtr=0x637EC1E0,
 dstCallId=0x2C, srcCallId=0x2D,
    digit=1, digit_begin_flags=0x0, rtp_timestamp=0x0
    rtp_expiration=0x0, dest_mask=0x2)
*Mar  1 15:36:11.504: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_end: (dstVdbPtr=
0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
   digit=1,duration=300,xruleCallingTag=0,xruleCalledTag=0, dest_mask=0x2), digit_tone_mode=0
```

The phone at the terminating gateway enters digit 2**.**

```
*Mar  1 15:36:11.604: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_begin: (dstVdbPt
r=0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
    digit=2, digit_begin_flags=0x0, rtp_timestamp=0x0
    rtp_expiration=0x0, dest_mask=0x2)
*Mar  1 15:36:11.904: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_digit_end: (dstVdbPtr=
0x637EC1E0, dstCallId=0x2C, srcCallId=0x2D,
   digit=2,duration=300,xruleCallingTag=0,xruleCalledTag=0, dest_mask=0x2), digit_tone_mode=0
Router#
Router#
*Mar  1 15:36:14.476: //-1/xxxxxxxxxxxx/CCAPI/cc_handle_periodic_timer: Calling
the callback, ccTimerctx - 0x628B6330
*Mar  1 15:36:14.476: //-1/xxxxxxxxxxxx/CCAPI/ccTimerStart: ccTimerctx - 0x628B6330
Router#
Router# !call hung up  The user at the terminating gateway hangs up the call.
Router#
```

The voice gateway receives a disconnect message from the terminating gateway. The cause code is 0x10 which is normal call clearing.

```
*Mar  1 15:36:22.916: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnected: (vdbPtr=
0x62EC61A4, callID=0x2D, cause=0x10)
*Mar  1 15:36:22.920: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(11=CC_EV_CALL_
DISCONNECTED), cid(45), disp(0)
*Mar  1 15:36:22.920: //45/45F2AAE28044/SSAPP:0:21/ssaTraceSct: cid(45)st(SSA_CS
_ACTIVE)ev(SSA_EV_CALL_DISCONNECTED)
oldst(SSA_CS_ALERT_RCVD)cfid(21)csize(2)in(0)fDest(0)
*Mar  1 15:36:22.920: //45/45F2AAE28044/SSAPP:0:21/ssaTraceSct: -cid2(44)st2(SSA
_CS_ACTIVE)oldst2(SSA_CS_ACTIVE)
*Mar  1 15:36:22.920: ssa: Disconnected cid(45) state(5) cause(0x10)
```

The voice gateway begins tearing down the conference and dropping the bridge.

```
*Mar  1 15:36:22.920: //-1/xxxxxxxxxxxx/CCAPI/ccConferenceDestroy: (confID=0x15,
 tag=0x0)
*Mar  1 15:36:22.920: //45/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done: (confID=0
x15, srcIF=0x62EC61A4, srcCallID=0x2D, dstCallID=0x2C, disposition=0 tag=0x0)
*Mar  1 15:36:22.920: //44/xxxxxxxxxxxx/CCAPI/cc_api_bridge_drop_done: (confID=0
x15, srcIF=0x637EC1E0, srcCallID=0x2C, dstCallID=0x2D, disposition=0 tag=0x0)
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(30=CC_EV_CONF_
DESTROY_DONE), cid(44), disp(0)
*Mar  1 15:36:22.924: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct:
cid(44)st(SSA_CS_CONF_DESTROYING)ev(SSA_EV_CONF_DESTROY_DONE)
oldst(SSA_CS_ACTIVE)cfid(21)csize(2)in(1)fDest(1)
*Mar  1 15:36:22.924: //44/45F2AAE28044/SSAPP:10002:21/ssaTraceSct: -cid2(45)st2
(SSA_CS_CONF_DESTROYING)oldst2(SSA_CS_ACTIVE)
*Mar  1 15:36:22.924: //45/45F2AAE28044/SSAPP:0:-1/ssaConfDestroyDone:
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/CCAPI/ccCallDisconnect: (callID=0x2C, cause=0x10
tag=0x0)
```

The voice gateway stops call accounting on the incoming call, indicated by the leg_type=True message. The cause code is then set for the originating leg.

```
*Mar  1 15:36:22.924: //44/45F2AAE28044/CCAPI/ccCallDisconnect: calling accounting start
for callID=44 leg_type=1
*Mar  1 15:36:22.924: //44/45F2AAE28044/CCAPI/ccCallDisconnect: existing_cause =
 0x0, new_cause = 0x10
*Mar  1 15:36:22.924: //44/xxxxxxxxxxxx/CCAPI/cc_api_get_transfer_info: (callID=0x2C)
*Mar  1 15:36:22.924: //45/xxxxxxxxxxxx/CCAPI/ccCallDisconnect: (callID=0x2D, cause=0x10
tag=0x0)
```

The voice gateway stops call accounting for the outgoing call, indicated by the leg_type=False message. The cause code is verified for the terminating leg.

```
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: calling accounting start
for callID=45 leg_type=0
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: existing_cause =
 0x10, new_cause = 0x10
*Mar  1 15:36:22.924: //45/45F2AAE28044/CCAPI/ccCallDisconnect: using the existing_cause
0x10
*Mar  1 15:36:22.928: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_transfer_info: (callID=0x2D)
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_api_icpif: expect factor = 0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/g113_calculate_impairment: (delay=79,
      loss=0), Io=0 Iq=0 Idte=0 Idd=0 Ie=10 Itot=10
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: the remote
 IP is 171.69.85.111
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: hwidb is Ethernet0/0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: reduce callnum of
entry: 0, voip: 0, mmoip: 0
*Mar  1 15:36:22.932: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: remove an entry
*Mar  1 15:36:22.932: //45/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnect_done: (vdbPtr=0x62EC61A4,
 callID=0x2D, disp=0, tag=0x0)
*Mar  1 15:36:22.932: //45/45F2AAE28044/CCAPI/ccTDPvtProfileTableObjectAccessManager: No
profileTable set for callID[45]
*Mar  1 15:36:22.936: //45/xxxxxxxxxxxx/CCAPI/ccTDUtilGetDataByRef: No tdObject
found in profileTable for tagID[6] of callID[45]
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: not incoming
 entry
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: entry's incoming FALSE.
*Mar  1 15:36:22.936: //45/45F2AAE28044/CCAPI/cc_delete_call_entry: is_incoming
is FALSE
*Mar  1 15:36:22.940: //45/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(12=CC_EV_CALL_
DISCONNECT_DONE), cid(45), disp(0)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: cid(45)st(SSA_CS
_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)
oldst(SSA_CS_ACTIVE)cfid(-1)csize(2)in(0)fDest(0)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaTraceSct: -cid2(44)st2(SSA
_CS_DISCONNECTING)oldst2(SSA_CS_CONF_DESTROYING)
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaDisconnectDone:
*Mar  1 15:36:22.940: //45/45F2AAE28044/SSAPP:0:-1/ssaAAA_CheckAccounting: accounting
generation enabled
*Mar  1 15:36:22.940: //45/xxxxxxxxxxxx/CCAPI/ccCallSetAAA_Accounting: callID=0x2D,
```

```
accounting=0
*Mar  1 15:36:22.944: //-1/xxxxxxxxxxxx/CCAPI/cc_decr_if_call_volume: not the VoIP or MMoIP
*Mar  1 15:36:22.948: //44/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnect_done: (vdbPtr=0x637EC1E0,
 callID=0x2C, disp=0, tag=0x0)
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: ccFreeRawMsg
Info(0x6307595C)
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: Decrement call volume
counter 1
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: current call volume: 0
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: entry's incoming TRUE.
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: is_incoming
is TRUE
*Mar  1 15:36:22.948: //44/45F2AAE28044/CCAPI/cc_delete_call_entry: Deleting
profileTable[0x6380E11C]
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructTDHashProfileTab: Destructor
Profile Table (0x6380E11C)
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructInstanceTDObject:
tdObject[0x63401148] tagID[5]
*Mar  1 15:36:22.948: //-1/xxxxxxxxxxxx/CCAPI/ccTDDestructInstanceTDObject:
tdObject[0x638BC1AC] tagID[6]
*Mar  1 15:36:22.956: //44/xxxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(12=CC_EV_CALL_
DISCONNECT_DONE), cid(44), disp(0)
*Mar  1 15:36:22.956: //44/45F2AAE28044/SSAPP:10002:-1/ssaTraceSct:
cid(44)st(SSA_CS_DISCONNECTING)ev(SSA_EV_CALL_DISCONNECT_DONE)
oldst(SSA_CS_CONF_DESTROYING)cfid(-1)csize(1)in(1)fDest(1)
Router#
*Mar  1 15:36:22.956: //44/45F2AAE28044/SSAPP:10002:-1/ssaDisconnectDone:
```

Related Commands

Command	Description
debug voip ccapi error	Traces error logs in the CCAPI.
voice call debug	Allows configuration of the voice call debug output.

debug voip confmsp

To display debugging information from the Conference Media Service Provider (CONFMSP) and its related applications, use the **debug voip confmsp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip confmsp

no debug voip confmsp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2SX	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.

Examples

The following is sample output from the **debug voip confmsp** command:

```
Router# debug voip confmsp

CONFMSP debugging is on
.
.
.
00:06:44:confmsp_setup_request:callID (6),
00:06:44:confmsp_setup_request:conf structure  (63DD27E4) created,
00:06:44:confmsp_bridge:confID(4), callIDs(6,5) xmitFunc 61D46D4C, dstIF 64912880

00:06:44:confmsp_bridge:confID(4), callIDs(6,5) event queued
00:06:44:confmsp_act_bridge: state = CONFMSP_STATE_SETUP, event=EV_CONFMSP_BRIDGING
00:06:44:confmsp_act_bridge:codec 1, codec_bytes 160, vad 1
00:06:44:confmsp_act_bridge:codec 1, codec_bytes 160, vad 1
00:06:44:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_SETUP,
event:EV_CONFMSP_BRIDGING, next_state:CONFMSP_STATE_CONNECT_CONFEREE
00:06:44:confmsp_get_dsmp_req_status:condition to be returned FALSE
00:06:44:confmsp_connect_response:found conf (63DD27E4) dsmp ret is 1

00:06:44:confmsp_act_bridge_success: state = CONFMSP_STATE_CONNECT_CONFEREE,
event=EV_CONFMSP_CNFRE_CONNECT_RESP_SUCCESS confmsp_caps_ind:context = 65241B34

00:06:44:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_CONNECT_CONFEREE,
event:EV_CONFMSP_CNFRE_CONNECT_RESP_SUCCESS, next_state:CONFMSP_STATE_CONNECTED
00:06:44:confmsp_bridge:confID(5), callIDs(6,7) xmitFunc 61D46D4C, dstIF 64912880

00:06:44:confmsp_bridge:confID(5), callIDs(6,7) event queued
00:06:44:confmsp_act_bridge: state = CONFMSP_STATE_SETUP, event=EV_CONFMSP_BRIDGING
00:06:44:confmsp_act_bridge:codec 1, codec_bytes 160, vad 1
```

```
00:06:44:confmsp_act_bridge:codec 1, codec_bytes 160, vad 1
00:06:44:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_SETUP,
event:EV_CONFMSP_BRIDGING, next_state:CONFMSP_STATE_CONNECT_CONFEREE
00:06:44:confmsp_get_dsmp_req_status:condition to be returned FALSE
00:06:44:confmsp_connect_response:found conf (6358A338) dsmp ret is 1

00:06:44:confmsp_act_bridge_success: state = CONFMSP_STATE_CONNECT_CONFEREE,
event=EV_CONFMSP_CNFRE_CONNECT_RESP_SUCCESS confmsp_caps_ind:context = 63588E70

00:06:44:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_CONNECT_CONFEREE,
event:EV_CONFMSP_CNFRE_CONNECT_RESP_SUCCESS, next_state:CONFMSP_STATE_CONNECTED
00:06:44:confmsp_bridge:confID(6), callIDs(6,8) xmitFunc 61D46D4C, dstIF 64912880

00:06:44:confmsp_bridge:confID(6), callIDs(6,8) event queued
00:06:44:confmsp_act_bridge: state = CONFMSP_STATE_SETUP, event=EV_CONFMSP_BRIDGING
00:06:44:confmsp_act_bridge:codec 1, codec_bytes 160, vad 1
00:06:44:confmsp_act_bridge:codec 1, codec_bytes 160, vad 1
00:06:44:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_SETUP,
event:EV_CONFMSP_BRIDGING, next_state:CONFMSP_STATE_CONNECT_CONFEREE
00:06:44:confmsp_get_dsmp_req_status:condition to be returned FALSE
00:06:44:confmsp_connect_response:found conf (6358CE50) dsmp ret is 1

00:06:44:confmsp_act_bridge_success: state = CONFMSP_STATE_CONNECT_CONFEREE,
event=EV_CONFMSP_CNFRE_CONNECT_RESP_SUCCESS confmsp_caps_ind:context = 63DD2524

00:06:44:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_CONNECT_CONFEREE,
event:EV_CONFMSP_CNFRE_CONNECT_RESP_SUCCESS, next_state:CONFMSP_STATE_CONNECTED
00:07:28:confmsp_bdrop:confID(4), callIDs(6,5)
00:07:28:confmsp_bdrop:confID(4), callIDs(6,5) event queued
00:07:28:confmsp_act_bdrop: state = CONFMSP_STATE_CONNECTED, event=EV_CONFMSP_BRIDGEDROP
00:07:28:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_CONNECTED,
event:EV_CONFMSP_BRIDGEDROP, next_state:CONFMSP_STATE_DISCONNECT_CONFEREE
00:07:28:confmsp_get_dsmp_req_status:condition to be returned FALSE
00:07:28:confmsp_disconnect_response:found conf (63DD27E4)

00:07:28:confmsp_connect_response:found conf (63DD27E4) dsmp ret is 10

00:07:28:confmsp_act_bdrop_success: state = CONFMSP_STATE_DISCONNECT_CONFEREE,
event=EV_CONFMSP_CNFRE_DISCONNECT_RESP_SUCCESS
00:07:28:CNFSM:cur_container:confmsp container,
cur_state:CONFMSP_STATE_DISCONNECT_CONFEREE,
event:EV_CONFMSP_CNFRE_DISCONNECT_RESP_SUCCESS, next_state:CONFMSP_STATE_BRIDGE_DROPPED
00:07:28:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_BRIDGE_DROPPED,
event:CNFSM_LAMBDA_EVENT, next_state:CNFSM_NO_STATE_CHANGE
00:07:29:confmsp_bdrop:confID(6), callIDs(6,8)
00:07:29:confmsp_bdrop:confID(6), callIDs(6,8) event queued
00:07:29:confmsp_bdrop:confID(5), callIDs(6,7)
00:07:29:confmsp_bdrop:confID(5), callIDs(6,7) event queued
00:07:29:confmsp_act_bdrop: state = CONFMSP_STATE_CONNECTED, event=EV_CONFMSP_BRIDGEDROP
00:07:29:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_CONNECTED,
event:EV_CONFMSP_BRIDGEDROP, next_state:CONFMSP_STATE_DISCONNECT_CONFEREE
00:07:29:confmsp_get_dsmp_req_status:condition to be returned FALSE
00:07:29:confmsp_act_bdrop: state = CONFMSP_STATE_CONNECTED, event=EV_CONFMSP_BRIDGEDROP
00:07:29:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_CONNECTED,
event:EV_CONFMSP_BRIDGEDROP, next_state:CONFMSP_STATE_DISCONNECT_CONFEREE
00:07:29:confmsp_get_dsmp_req_status:condition to be returned FALSE
00:07:29:confmsp_disconnect_response:found conf (6358CE50)

00:07:29:confmsp_connect_response:found conf (6358CE50) dsmp ret is 10

00:07:29:confmsp_act_bdrop_success: state = CONFMSP_STATE_DISCONNECT_CONFEREE,
event=EV_CONFMSP_CNFRE_DISCONNECT_RESP_SUCCESS
00:07:29:CNFSM:cur_container:confmsp container,
cur_state:CONFMSP_STATE_DISCONNECT_CONFEREE,
event:EV_CONFMSP_CNFRE_DISCONNECT_RESP_SUCCESS, next_state:CONFMSP_STATE_BRIDGE_DROPPED
00:07:29:confmsp_act_terminate: state = CONFMSP_STATE_BRIDGE_DROPPED,
event=CNFSM_LAMBDA_EVENT
00:07:29:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_BRIDGE_DROPPED,
event:CNFSM_LAMBDA_EVENT, next_state:CNFSM_NULL_STATE
00:07:29:confmsp_free_conf:freeing 6358CE50

00:07:29:confmsp_disconnect_response:found conf (6358A338)
```

```
00:07:29:confmsp_connect_response:found conf (6358A338) dsmp ret is 10

00:07:29:confmsp_act_bdrop_success: state = CONFMSP_STATE_DISCONNECT_CONFEREE,
event=EV_CONFMSP_CNFRE_DISCONNECT_RESP_SUCCESS
00:07:29:CNFSM:cur_container:confmsp container,
cur_state:CONFMSP_STATE_DISCONNECT_CONFEREE,
event:EV_CONFMSP_CNFRE_DISCONNECT_RESP_SUCCESS, next_state:CONFMSP_STATE_BRIDGE_DROPPED
00:07:29:confmsp_act_terminate: state = CONFMSP_STATE_BRIDGE_DROPPED,
event=CNFSM_LAMBDA_EVENT
00:07:29:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_BRIDGE_DROPPED,
event:CNFSM_LAMBDA_EVENT, next_state:CNFSM_NULL_STATE
00:07:29:confmsp_free_conf:freeing 6358A338

00:07:29:confmsp_disconnect:callID (6)
00:07:29:confmsp_disconnect:callID (6) event queued
00:07:29:confmsp_act_disconnected: state = CONFMSP_STATE_BRIDGE_DROPPED,
event=EV_CONFMSP_DISCONNECT
00:07:29:CNFSM:cur_container:confmsp container, cur_state:CONFMSP_STATE_BRIDGE_DROPPED,
event:EV_CONFMSP_DISCONNECT, next_state:CNFSM_NULL_STATE
00:07:29:confmsp_free_conf:freeing 63DD27E4
```

debug voip dcapi

To debug the device control application programming interface (DCAPI), use the **debug voip dcapi**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip dcapi [**error**| **inout**]

no debug voip dcapi [**error**| **inout**]

Syntax Description

error	(Optional) Displays error logs in the DCAPI.
inout	(Optional) Displays the execution path through the DCAPI.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(14)T	This command was introduced.

Usage Guidelines

The **debug voip dcapi error** command traces the error logs in the DCAPI, which is the software layer that interfaces the SCCP Telephony Control Application (STCAPP) with the Cisco CallManager using the Skinny Client Control Protocol (SCCP). Error logs are generated during normal call processing when there are insufficient resources, or when there are problems in the device control API. This debug command shows error events or unexpected behavior in system software.

The **debug voip dcapi inout** command shows how a call executes through the software. This command traces the execution path through the DCAPI during communications with the SCCP service provider interface (SPI) and the call control API (CCAPI) that controls the physical voice port. You can use the output from this command to understand how devices are being handled by the APIs and to see the call setup and teardown operations performed on the telephony call leg.

Examples

Following is sample output from the **debug voip dcapi inout**command during call setup:

```
Router# debug voip dcapi inout
*Jan 27 16:26:23.957: dc_api_device_set_ringer_res: Set Ringer message success
*Jan 27 16:26:23.957: //-1/xxxxxxxxxxxx/CCAPI/dc_api_device_stop_tone_res:

*Jan 27 16:26:23.957: dc_api_device_stop_tone_res: Stop Tone message success
*Jan 27 16:26:23.957: //-1/xxxxxxxxxxxx/CCAPI/dc_api_media_open_rcv_channel:

*Jan 27 16:26:23.957: dc_api_media_open_rcv_channel: evt DC_EV_MEDIA_OPEN_RCV_CHNL is
successsfully enqueued to app
*Jan 27 16:26:23.957: //-1/xxxxxxxxxxxx/CCAPI/dc_api_device_stop_tone_res:

*Jan 27 16:26:23.957: dc_api_device_stop_tone_res: Stop Tone message success
```

```
*Jan 27 16:26:23.957: //-1/xxxxxxxxxxxx/CCAPI/dc_api_device_call_state_res:

*Jan 27 16:26:23.957: dc_api_device_call_state_res: Call State message success
*Jan 27 16:26:23.957: //-1/xxxxxxxxxxxx/CCAPI/dc_api_device_call_info_res:
```

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

Table 2: debug voip dcapi Field Descriptions

Field	Description
nn :*nn* :*nn* :	Timestamp time in hours (military format), minutes, and seconds that indicates when the DCAPI event occurred.
dc_api_*message:*	The DCAPI event in which the SCCP SPI translation occurred.

Related Commands

Command	Description
debug voip application stcapp events	Debugs STCAPP events.
debug voip application stcapp functions	Debugs STCAPP functions.

debug voip dialpeer

To display information about the voice dial peers, use the **debug voip dialpeer** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip dialpeer [**all**| **default**| **detail**| **error** [**call [informational]**| **software [informational]**]| **function**| **inout**]

no debug voip dialpeer

Syntax Description

all	(Optional) Displays all dialpeer debugging messages.
default	(Optional) Displays dialpeer inout and error debugging messages. This option also runs if no keywords are added.
detail	(Optional) Displays detailed dialpeer information.
error	(Optional) Displays dialpeer error messages.
call	(Optional) Displays call processing errors.
informational	(Optional) Displays minor errors and major errors. Without the **informational** keyword, only major errors are displayed.
software	(Optional) Displays software processing errors.
function	(Optional) Displays dialpeer functions.
inout	(Optional) Displays dialpeer in/out functions.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command replaces the **debug dialpeer** command.

Usage Guidelines

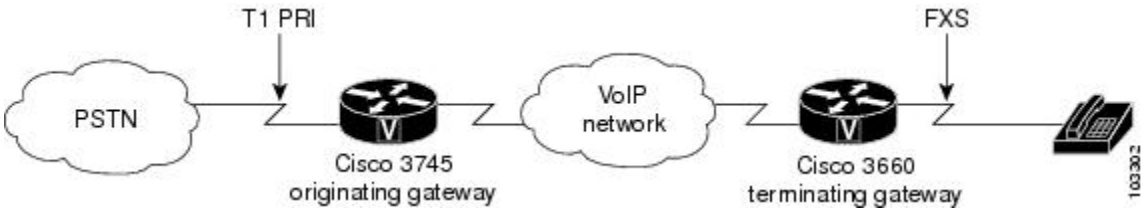
Disable console logging and use buffered logging before using the **debug voip dialpeer**command. Using the **debug voip dialpeer** command generates a large volume of debugging messages, which can affect router performance.

Examples

The following examples show output for variations of the **debug voip dialpeer** command:

For these examples, the topology shown in the figure below is used.

Figure 2: Network Topology for debug voip dialpeer Output Examples



Examples

```
Router# debug voip dialpeer detail

voip dialpeer detail debugging is on
Router#
```

The following event identifies the called number:

```
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpAssociateIncomingPeerCore:
   Match Rule=DP_MATCH_INCOMING_DNIS; Called Number=83103
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchPeertype:
   Is Incoming=TRUE, Number Expansion=FALSE
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Dial String=83103, Expanded String=83103, Calling Number=
   Timeout=TRUE, Is Incoming=TRUE, Peer Info Type=DIALPEER_INFO_SPEECH
```

The following event identifies the incoming dial peer and shows that it has been matched:

```
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/MatchNextPeer:
   Result=Success(0); Incoming Dial-peer=1 Is Matched
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpAssociateIncomingPeerCore:
   Match Rule=DP_MATCH_INCOMING_DNIS; Called Number=83103
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchPeertype:
   Is Incoming=TRUE, Number Expansion=FALSE
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Dial String=83103, Expanded String=83103, Calling Number=
   Timeout=TRUE, Is Incoming=TRUE, Peer Info Type=DIALPEER_INFO_FAX
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Result=-1
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpAssociateIncomingPeerCore:
   Match Rule=DP_MATCH_ANSWER; Calling Number=4085550111
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchPeertype:
   Is Incoming=TRUE, Number Expansion=FALSE
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Dial String=, Expanded String=, Calling Number=4085550111T
   Timeout=TRUE, Is Incoming=TRUE, Peer Info Type=DIALPEER_INFO_FAX
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Result=-1
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpAssociateIncomingPeerCore:
   Match Rule=DP_MATCH_ORIGINATE; Calling Number=4085550111
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchPeertype:
   Is Incoming=TRUE, Number Expansion=FALSE
*Apr 18 21:07:35.291: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
```

```
   Dial String=, Expanded String=, Calling Number=4085550111T
   Timeout=TRUE, Is Incoming=TRUE, Peer Info Type=DIALPEER_INFO_FAX
*Apr 18 21:07:35.295: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Result=-1
```

The following event shows the number expansion. This is the only dial peer debug command that shows the number expansion.

```
*Apr 18 21:07:35.295: //-1/xxxxxxxxxxxx/DPM/dpMatchCore:
   Dial String=83103, Expanded String=3600, Calling Number=
   Timeout=TRUE, Is Incoming=FALSE, Peer Info Type=DIALPEER_INFO_SPEECH
```

The next few lines show matching for the outgoing dial peer. These lines show the matching sequence if the first match is not available.

```
*Apr 18 21:07:35.295: //-1/xxxxxxxxxxxx/DPM/MatchNextPeer:
   Result=Success(0); Outgoing Dial-peer=3600 Is Matched
*Apr 18 21:07:35.295: //-1/xxxxxxxxxxxx/DPM/MatchNextPeer:
   Result=Success(0); Outgoing Dial-peer=36 Is Matched
*Apr 18 21:07:35.295: //-1/xxxxxxxxxxxx/DPM/MatchNextPeer:
   Result=Success(0); Outgoing Dial-peer=360 Is Matched
*Apr 18 21:07:35.295: //-1/23ED4B1B8010/DPM/dpMatchCore:
   Dial String=83103, Expanded String=3600, Calling Number=
   Timeout=TRUE, Is Incoming=FALSE, Peer Info Type=DIALPEER_INFO_SPEECH
```

Examples

```
Router# debug voip dialpeer inout

voip dialpeer inout debugging is on
```

The following event shows the calling and called numbers:

```
*May  1 19:32:11.731: //-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:
   Calling Number=4085550111, Called Number=3600, Voice-Interface=0x0,
   Timeout=TRUE, Peer Encap Type=ENCAP_VOIP, Peer Search Type=PEER_TYPE_VOICE,
   Peer Info Type=DIALPEER_INFO_SPEECH
```

The following event shows the incoming dial peer:

```
*May  1 19:32:11.731: //-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:
   Result=Success(0) after DP_MATCH_INCOMING_DNIS; Incoming Dial-peer=100
*May  1 19:32:11.731: //-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:
   Calling Number=4085550111, Called Number=3600, Voice-Interface=0x0,
   Timeout=TRUE, Peer Encap Type=ENCAP_VOIP, Peer Search Type=PEER_TYPE_VOICE,
   Peer Info Type=DIALPEER_INFO_SPEECH
*May  1 19:32:11.731: //-1/6372E2598012/DPM/dpAssociateIncomingPeerCore:
   Result=Success(0) after DP_MATCH_INCOMING_DNIS; Incoming Dial-peer=100
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersCore:
   Calling Number=, Called Number=3600, Peer Info Type=DIALPEER_INFO_SPEECH
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersCore:
   Match Rule=DP_MATCH_DEST; Called Number=3600
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersCore:
   Result=Success(0) after DP_MATCH_DEST
*May  1 19:32:11.735: //-1/6372E2598012/DPM/dpMatchPeersMoreArg:
   Result=SUCCESS(0)
```

The following event shows the matched dial peers in the order of priority:

```
List of Matched Outgoing Dial-peer(s):
     1: Dial-peer Tag=3600
     2: Dial-peer Tag=36
```

Related Commands

Command	Description
call-block (dial peer)	Enables blocking of incoming calls on the dial peer.

Command	Description
carrier-id (dial-peer)	Identifies the carrier handling the incoming call.
session target (ENUM)	Specifies the ENUM search table for the target session.
show dial-peer voice	Displays the configuration of the dial peer.
translation-profile (dial-peer)	Assigns a translation profile to the dial peer.
trunkgroup (dial-peer)	Assigns a trunk group to the dial peer.
trunk-group-label (dial-peer)	Identifies the trunk group handling the incoming call.

debug voip dsm

To troubleshoot the DSP stream manager (DSM) subsystem, use the **debug voip dsm** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip dsm [**all**| **dsp**| **error**| **rtp**| **session**| **stats**| **tone**| **vofr**]

no debug voip dsm

Syntax Description

all	(Optional) Displays all DSM debugging messages.
dsp	(Optional) Enables a digital signal processor (DSP) message trace.
error	(Optional) Displays DSM error messages.
rtp	(Optional) Enables Real-Time Protocol (RTP) debugging on DSM.
session	(Optional) Enables session debugging.
stats	(Optional) Displays DSM statistics.
tone	(Optional) Displays tone debugging.
vofr	(Optional) Enables Voice over Frame Relay (VoFR) debugging on the VPM.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command replaces the **debug vtsp dsp** command.
12.3(14)T	The **vofr** keyword is no longer available in Cisco IOS Release 12.3(14)T.

Usage Guidelines

To debug VoIP calls, use this command in conjunction with **debug voip vtsp** command and **debug voip dsmp** commands. All the related information for media processing is now available by using Distributed Stream Media Processor (DSMP) . DSM is responsible for creating streams and issuing connections between them.

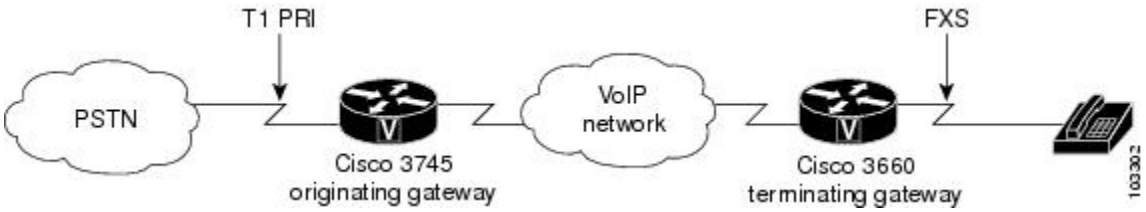
Examples

The following examples show output for variations of the **debug voip dsm** command:

- debug voip dsm, on page 42
- debug voip dsm, on page 42
- debug voip dsm, on page 42

For these examples, the topology shown in the figure below is used.

Figure 3: Network Topology for debug voip dsm Examples



Examples

```
Router# debug voip dsm session

DSP Stream Manager session debugging is on
*Apr 18 21:15:39.679: //-1/44A507668015/DSM:(2/1:23):0:0:0/dsm_start_basic_sm: .
*Apr 18 21:15:39.679: //-1/44A507668015/DSM:(2/1:23):0:8:4/dsm_start_basic_sm: dsp resource
 manager opened. ret 4
*Apr 18 21:15:39.679: //-1/44A507668015/DSM:(2/1:23):0:8:4/dsm_open_voice_and_set_params:
.
*Apr 18 21:15:39.679: //-1/44A507668015/DSM:(2/1:23):0:8:4/set_playout_dmgr: playout default
```

The following event shows the echo cancellation setting:

```
*Apr 18 21:15:39.679: //-1/44A507668015/DSM:(2/1:23):0:8:4/dsm_dsp_echo_canceller_control:
 echo_cancel: 1
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsp_stream_mgr_play_tone: .
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_INIT E:E_DSM_CC_GEN_TONE]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_gen_tone: Tone is not
on, ignoring
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_INIT]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_INIT E:E_DSM_CC_BRIDGE]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_bridge: .
```

The following event indicates that modem relay is not supported on the specified port, which is port 2/1:23:

```
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_bridge: Modem Relay NOT
 Supported on this end-point/voice-port.
disabling it...
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGING]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGING E:E_DSM_CC_CAPS_IND]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: .
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: RTP
PT:NTE[101],NSE[100],FaxInd[96],FaxAck[97],CiscoDT
121],FaxRelay[122],CASsig[123],ClearChan[125],PCMu[0],PCMa[8]Codec[4], TxDynamicPayload[0],
 RxDynamicPayload[0]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: dtmf relay:
mode=1, codec=1
```

```
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: Modem Relay
NOT Supported on this end-point/voice-port
disabling it...
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: passthrough:
cap_modem_proto 4,cap_modem_codec 1, cap_
em_redundancy 0, payload 103, modem_relay 0, gw-xid 0
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_debug_caps_encap: Encap 1,
Vad 2, Codec 0x4, CodecBytes 20,
             FaxRate 1, FaxBytes 20, FaxNsf 0xAD0051
             SignalType 2
             DtmfRelay 1, Modem 2, SeqNumStart 0x5A1
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: FORKING
Parameters are forking mask: 0, simple_forking
dec_mask: 0, complex_forking_codec_mask 0
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ind: playout: [
mode:1,init:60, min:40, max:200]. data_mode
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGING]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGING E:E_DSM_CC_CAPS_ACK]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ack: .
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ack: passthrough:
cap_modem_proto 4, cap_modem_codec 1, cap
dem_redundancy 0, payload 103, modem_relay 0, gw-xid 0
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_caps_ack: Named Telephone
 Event payload rcv 101, tx 101
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_switch_codec: Required codec
 is 16,  current dsp codec is -1
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_switch_codec: codec = 16
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_PENDING_CODEC_SWITCH]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_PENDING_CODEC_SWITCH E:E_DSM_DS_PEND_SUCCESS]
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_act_pend_codec_success: .
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_open_voice_and_set_params:
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/set_playout_dmgr: playout default
```

The following event shows the echo cancellation setting:

```
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_dsp_echo_canceller_control:
 echo_cancel: 1
```

The following event shows that the codec has changed:

```
*Apr 18 21:15:39.695:
//43/44A507668015/DSM:(2/1:23):0:8:4/dsm_setup_stream_after_switch_codec_succ: codec change
 success
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_add_fork: .
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_update_fork_info: add_fork=0
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_get_xmit_info_node: .
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_update_fork_info:  xmit func
 is 61A7CDC4, context is 64F42DA0 peer_c
_id: 44, stream_count: 1, update_flag 0
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_update_fork_info: The stream
 bit-mask is 1
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_update_fork_info: The stream
 type is 0
*Apr 18 21:15:39.695: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_update_fork_info: The logical
 ssrc is 64 for stream 0
*Apr 18 21:15:39.699: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_update_stream_count:
*Apr 18 21:15:39.699:  g711_voice_count=0 g711_avt_count = 0
 g711_voice_avt_count = 0 complex_voice_count = 1
 complex_avt_count = 0 complex_voice_avt_count = 0
*Apr 18 21:15:39.699: //43/44A507668015/DSM:(2/1:23):0:8:4/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGED]
```

Examples

```
Router# debug voip dsm stats

DSP Stream Manager stats debugging is on
```

The following event shows that the DSM is requesting statistics:

```
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGED E:E_DSM_CC_REQ_PACK_STAT]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_cc_stats_req: .
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_cc_stats_req:
-1->dmgr=0x645461E0, stats_reqs=0
```

The following events show statistics for DSM transmission, reception, delay, and errors:

```
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGED]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGED E:E_DSM_DSP_GET_TX]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res: .
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res:
-4->dmgr=0x645461E0,stats_reqs=3
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGED]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGED E:E_DSM_DSP_GET_RX]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res: .
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res:
-4->dmgr=0x645461E0,stats_reqs=2
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGED]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGED E:E_DSM_DSP_GET_VP_DELAY]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res: .
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res:
-4->dmgr=0x645461E0,stats_reqs=1
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [B SM: R:FSM_OK ->
S:S_DSM_BRIDGED]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_exec: [Feat SM: S:NONE B SM:
 S:S_DSM_BRIDGED E:E_DSM_DSP_GET_VP_ERROR]
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res: .
Router#
*May  1 19:40:09.775: //43/7EE6F84B8016/DSM:(4/0/0):-1:1:1/dsm_act_packet_stats_res:
-4->dmgr=0x645461E0,stats_reqs=0
```

Examples

The following is sample output from the **debug voip dsm** command, with Cisco IOS Release 12.3(14)T software, when a VoIP call is in transition to the connected state:

```
Router# debug voip dsm

*Jun  8 20:10:33.205: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_create: .
*Jun  8 20:10:33.209: //28/6F1FA7868003/DSM:(2/0:23):-1/set_echo_canceller_data: echo_cancel:
 1
*Jun  8 20:10:33.209: //28/6F1FA7868003/DSM:(2/0:23):-1/set_echo_canceller_data: echo_flags:
 55, echo_len: 512
*Jun  8 20:10:33.217: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_reserve_resource_cb: .
*Jun  8 20:10:33.217: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_set_elog_enable: .
*Jun  8 20:10:33.477: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_reinit_platform_info:
 .
*Jun  8 20:10:33.477: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_reserve_resource_cb: .
*Jun  8 20:10:33.485: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_bridge: .
*Jun  8 20:10:33.485: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_bridge:
*Jun  8 20:10:33.485:  dsp_stream_mgr_bridge, src_call_id 28, dst_call_id 27
*Jun  8 20:10:33.485: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_bridge: creating
packet streams
*Jun  8 20:10:33.489: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_insert_conn_array_entry:
dmgr->connArr.count = 2
*Jun  8 20:10:33.489: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_do_caps_ind: .
*Jun  8 20:10:33.493: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_cap_ind_ack:
src_call_id = 28, dst_call_id = 27, is_cap_ack = 1
*Jun  8 20:10:33.493: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_cap_ind_ack:
src_call_id = 28, dst_call_id = 27, is_cap_ack = 0
*Jun  8 20:10:33.493: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_ind_negotiation: Caps in
caps indication:
```

```
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: playout: [
mode:1,init:60, min:40, max:250]. data_mode:0
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: RTP
PT:NTE[101],NSE[100],FaxInd[96],FaxAck[97],CiscoDTMF[121],FaxRelay[122],CASsig[123],ClearChan[125],PCMu[0],PCMa[8]Codec[4],
 TxDynamicPayload[0], RxDynamicPayload[0]
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: passthrough:
cap_modem_proto 4,cap_modem_codec 1, cap_modem_redundancy 1, payload 100, modem_relay 0,
gw-xid 0
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_debug_caps_encap: Encap 1, Vad
 2, Codec 0x4, CodecBytes 20,
             FaxRate 2, FaxBytes 20, FaxNsf 0xAD0051
             SignalType 2
             DtmfRelay 1, Modem 2, SeqNumStart 0x0
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_ind_negotiation: Caps after
 caps negotiation:
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: playout: [
mode:1,init:60, min:40, max:250]. data_mode:0
*Jun  8 20:10:33.497: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: RTP
PT:NTE[101],NSE[100],FaxInd[96],FaxAck[97],CiscoDTMF[121],FaxRelay[122],CASsig[123],ClearChan[125],PCMu[0],PCMa[8]Codec[4],
 TxDynamicPayload[0], RxDynamicPayload[0]
*Jun  8 20:10:33.501: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: passthrough:
cap_modem_proto 4,cap_modem_codec 1, cap_modem_redundancy 1, payload 100, modem_relay 0,
gw-xid 0
*Jun  8 20:10:33.501: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_debug_caps_encap: Encap 1, Vad
 2, Codec 0x4, CodecBytes 20,
             FaxRate 2, FaxBytes 20, FaxNsf 0xAD0051
             SignalType 2
             DtmfRelay 1, Modem 2, SeqNumStart 0x0
*Jun  8 20:10:33.501: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_cap_ind_ack: packet
 streams already created during bridging
*Jun  8 20:10:33.501: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_ind_negotiation: Caps in
caps indication:
*Jun  8 20:10:33.501: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: playout: [
mode:1,init:60, min:40, max:250]. data_mode:0
*Jun  8 20:10:33.501: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: RTP
PT:NTE[101],NSE[100],FaxInd[96],FaxAck[97],CiscoDTMF[121],FaxRelay[122],CASsig[123],ClearChan[125],PCMu[0],PCMa[8]Codec[4],
 TxDynamicPayload[0], RxDynamicPayload[0]
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: passthrough:
cap_modem_proto 4,cap_modem_codec 1, cap_modem_redundancy 1, payload 100, modem_relay 0,
gw-xid 0
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_debug_caps_encap: Encap 1, Vad
 2, Codec 0x4, CodecBytes 20,
             FaxRate 2, FaxBytes 20, FaxNsf 0xAD0051
             SignalType 2
             DtmfRelay 1, Modem 2, SeqNumStart 0x0
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_ind_negotiation: Caps after
 caps negotiation:
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: playout: [
mode:1,init:60, min:40, max:250]. data_mode:0
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: RTP
PT:NTE[101],NSE[100],FaxInd[96],FaxAck[97],CiscoDTMF[121],FaxRelay[122],CASsig[123],ClearChan[125],PCMu[0],PCMa[8]Codec[4],
 TxDynamicPayload[0], RxDynamicPayload[0]
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_caps_dump: passthrough:
cap_modem_proto 4,cap_modem_codec 1, cap_modem_redundancy 1, payload 100, modem_relay 0,
gw-xid 0
*Jun  8 20:10:33.505: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_debug_caps_encap: Encap 1, Vad
 2, Codec 0x4, CodecBytes 20,
             FaxRate 2, FaxBytes 20, FaxNsf 0xAD0051
             SignalType 2
             DtmfRelay 1, Modem 2, SeqNumStart 0x0
*Jun  8 20:10:33.509: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_connect_cb: .
*Jun  8 20:10:36.229: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_req_stats: .
*Jun  8 20:10:36.233: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_stats_cb: .
*Jun  8 20:10:38.265: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_req_stats: .
*Jun  8 20:10:38.269: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_stats_cb: .
*Jun  8 20:10:43.481: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_req_stats: .
*Jun  8 20:10:43.489: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_stats_cb: .
```

The following is sample output from the **debug voip dsm** command, with Cisco IOS Release 12.3(14)T software, when a VoIP call is in transition from connected to the disconnected state:

```
Router# debug voip dsm
```

```
*Jun  8 20:12:14.701: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_req_stats: .
*Jun  8 20:12:14.705: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_stats_cb: .
*Jun  8 20:12:18.721: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_bridge_drop:
*Jun  8 20:12:18.721:  dsp_stream_mgr_bridge_drop, src_call_id 28, dst_call_id 27
*Jun  8 20:12:18.721: //28/6F1FA7868003/DSM:(2/0:23):-1/dsm_delete_conn_array_entry:
dmgr->connArr.count = 0
*Jun  8 20:12:18.737: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_stats_cb: .
*Jun  8 20:12:18.765: //28/6F1FA7868003/DSM:(2/0:23):-1/dsp_stream_mgr_destroy: .
*Jun  8 20:12:18.765:
//28/6F1FA7868003/DSM:(2/0:23):-1/dsmapi_accept_modem_passthrough_session:  : dmgr: 6561C520,
 active sessions 0, max sessions: 16 rejected sessions till now: 0
*Jun  8 20:12:18.769: //28/6F1FA7868003/DSM:(2/0:23):-1/dsmp_dsmapi_reserve_resource_cb: .
```

Related Commands

Command	Description
debug voip dsmp	Displays debugging information from the DSMP and its related applications.
debug voip vtsp	Displays information about the VTSP.

debug voip dsmp

To display debugging information from the Distributed Stream Media Processor (DSMP) and its related applications, use the **debug voip dsmp** command in privileged EXEC mode. To disable debugging output, use the **no**form of this command.

debug voip dsmp [**all**| **default**| **error**| **event**| **function**| **individual**| **inout**| **rtp**| **session**| **stats**| **tone**| **vofr**]

no debug voip dsmp

Syntax Description

all	(Optional) Enables all DSMP debugging (except **stats**).
default	(Optional) Activates inout, error, and event debugging.
error	(Optional) Enables DSMP error debugging.
event	(Optional) Enables state machine debugging.
function	(Optional) Enables procedure tracing.
individual	(Optional) Enables individual DSMP debugging.
inout	(Optional) Enables subsystem inout debugging.
rtp	(Optional) Enables Real-Time Protocol (RTP) debugging on DSMP.
session	(Optional) Enables session debugging.
stats	(Optional) Enables DSMP statistics debugging.
tone	(Optional) Enables tone debugging.
vofr	(Optional) Enables Voice over Frame Relay (VoFR) debugging on the VPM.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.

Release	Modification
12.3(14)T	The **all**, **default**, **error**, **event**, **function**, **individual**, **inout**, **rtp**, **session**, **stats**, **tone**, and **vofr**keywords were added to the command.
12.2SX	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.

Usage Guidelines

To debug VoIP calls, use this command in conjunction with **debug voip vtsp** command and **debug voip dsm** commands. All the related information for media processing is now available by using DSMP . DSM is responsible for creating streams and issuing connections between them.

Examples

The following is sample output from the **debug voip dsmp** command for transcoding call:

```
Router# debug voip dsmp

Syslog logging:enabled (11 messages dropped, 2 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, 236 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages:disabled
    Trap logging:level informational, 25 message lines logged

Log Buffer (3000000 bytes):

00:10:18:dsmpAddStream::
00:10:18:dsmpAddStream::
00:10:18:dsmpAddStream::
00:10:18:dsmpAddStream::
00:10:18:dsmpConnectXcodeBinderReq
00:10:18:dsmp_process_event:
00:10:18:dsmp_process_event:evt->requestType = E_DSMP_CC_XCODE_REQ
00:10:18:dsmpxc_act_alloc_rsc::state = S_DSMPXC_INITIAL event = E_DSMP_CC_XCODE_REQ
00:10:18:dsp_ret = 4 (if failed, cause = 0)
00:10:18:dsp_intf = 0x63DD21D8
00:10:18:dsmp_set_state_var:message to DSP
    successful,ret::4
00:10:18:CNFSM:cur_container:xcoder_container, cur_state:S_DSMPXC_INITIAL,
event:E_DSMP_CC_XCODE_REQ, next_state:S_DSMPXC_RSC_ALLOCING
00:10:18:dsmp_is_ret_succ::
00:10:18:dsmpxc_act_init_rsc::state = S_DSMPXC_RSC_ALLOCING event = CNFSM_LAMBDA_EVENT
00:10:18:dsmp_set_state_var:message to DSP
    successful,ret::4
00:10:18:CNFSM:cur_container:xcoder_container, cur_state:S_DSMPXC_RSC_ALLOCING,
event:CNFSM_LAMBDA_EVENT, next_state:S_DSMPXC_RSC_INITING
00:10:18:dsmp_is_ret_succ::
00:10:18:dsmpxc_act_open_rsc::state = S_DSMPXC_RSC_INITING event = CNFSM_LAMBDA_EVENT
00:10:18:dsmp_set_state_var:message to DSP
    successful,ret::4
00:10:18:CNFSM:cur_container:xcoder_container, cur_state:S_DSMPXC_RSC_INITING,
event:CNFSM_LAMBDA_EVENT, next_state:S_DSMPXC_RSC_OPENING
00:10:18:dsmp_is_ret_succ::
00:10:18:dsmpxc_act_program_rsc::state = S_DSMPXC_RSC_OPENING event = CNFSM_LAMBDA_EVENT
00:10:18:dsmp_set_state_var:message to DSP
    successful,ret::4
00:10:18:CNFSM:cur_container:xcoder_container, cur_state:S_DSMPXC_RSC_OPENING,
event:CNFSM_LAMBDA_EVENT, next_state:S_DSMPXC_XCODE_PEND
```

```
00:10:18:CNFSM:new_container:xcoding_container
00:10:18:dsmp_is_ret_succ::
00:10:18:dsmpxc_act_succ_conn_req::state = S_DSMPXC_XCODE_PEND event = CNFSM_LAMBDA_EVENT
00:10:18:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_XCODE_PEND,
event:CNFSM_LAMBDA_EVENT, next_state:S_DSMPXC_XCODE
00:10:18:no_stream_in_session
Router# clear log
Clear logging buffer [confirm]
Router#
Router# clear logsh logg

Syslog logging:enabled (11 messages dropped, 2 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, 274 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (4096 bytes)
    Count and timestamp logging messages:disabled
    Trap logging:level informational, 25 message lines logged

Log Buffer (3000000 bytes):

00:10:50:dsmpDeleteStream
00:10:50:dsmpDeleteStream
00:10:50:dsmpDeleteStream
00:10:50:dsmpDeleteStream
00:10:50:dsmp_process_event:
00:10:50:dsmp_process_event:evt->requestType = E_DSMP_CC_DELETE_STREAM
00:10:50:dsmpxc_act_delete_stream::state = S_DSMPXC_XCODE event = E_DSMP_CC_DELETE_STREAM
00:10:50:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_XCODE,
event:E_DSMP_CC_DELETE_STREAM, next_state:CNFSM_NO_STATE_CHANGE
00:10:50:no_stream_in_session
00:10:50:dsmp_process_event:evt->requestType = E_DSMP_CC_DELETE_STREAM
00:10:50:dsmpxc_act_delete_stream::state = S_DSMPXC_XCODE event = E_DSMP_CC_DELETE_STREAM
00:10:50:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_XCODE,
event:E_DSMP_CC_DELETE_STREAM, next_state:CNFSM_NO_STATE_CHANGE
00:10:50:no_stream_in_session
00:10:50:dsmp_process_event:evt->requestType = E_DSMP_CC_DELETE_STREAM
00:10:50:dsmpxc_act_delete_stream::state = S_DSMPXC_XCODE event = E_DSMP_CC_DELETE_STREAM
00:10:50:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_XCODE,
event:E_DSMP_CC_DELETE_STREAM, next_state:CNFSM_NO_STATE_CHANGE
00:10:50:no_stream_in_session
00:10:50:dsmp_process_event:evt->requestType = E_DSMP_CC_DELETE_STREAM
00:10:50: dsmpxc_act_delete_stream::state = S_DSMPXC_XCODE event = E_DSMP_CC_DELETE_STREAM

00:10:50:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_XCODE,
event:E_DSMP_CC_DELETE_STREAM, next_state:CNFSM_NO_STATE_CHANGE
00:10:50:no_stream_in_session
00:10:50:dsmpxc_act_stop_rsc::state = S_DSMPXC_XCODE event = CNFSM_LAMBDA_EVENT
00:10:50:dsmp_set_state_var:message to DSP
    successful,ret::4
00:10:50:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_XCODE,
event:CNFSM_LAMBDA_EVENT, next_state:S_DSMPXC_CLOSING
00:10:50:dsmp_is_ret_succ::
00:10:50:dsmpxc_act_dealloc_rsc::state = S_DSMPXC_CLOSING event = CNFSM_LAMBDA_EVENT
00:10:50:dsmp_set_state_var:message to DSP
    successful,ret::4
00:10:50:CNFSM:cur_container:xcoding_container, cur_state:S_DSMPXC_CLOSING,
event:CNFSM_LAMBDA_EVENT, next_state:S_DSMPXC_STILL_CLOSING
00:10:50:CNFSM:new_container:xcoder_container
00:10:50:dsmp_is_ret_succ::
00:10:50:dsmpxc_act_start_timer::state = S_DSMPXC_STILL_CLOSING event = CNFSM_LAMBDA_EVENT

00:10:50:CNFSM:cur_container:xcoder_container, cur_state:S_DSMPXC_STILL_CLOSING,
event:CNFSM_LAMBDA_EVENT, next_state:CNFSM_NO_STATE_CHANGE
00:10:50:dsmp_process_event:evt->requestType = E_DSMP_DSPRM_CLOSE_COMPLETE
00:10:50:dsmpxc_act_terminate::state = S_DSMPXC_STILL_CLOSING event =
E_DSMP_DSPRM_CLOSE_COMPLETE
00:10:50:dsmpxc_act_terminate Removing the program based stream = 1
00:10:50:CNFSM:cur_container:xcoder_container, cur_state:S_DSMPXC_STILL_CLOSING,
event:E_DSMP_DSPRM_CLOSE_COMPLETE, next_state:CNFSM_NULL_STATE
```

```
00:10:50:dsmp_free_session
00:10:50:dsmp_process_event:
```

The following is sample output from the **debug voip dsmp** command, with Cisco IOS Release 12.3(14)T software, when a VoIP call is in transition to the connected state:

```
Router# debug voip dsmp

*May 22 04:12:17.775: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: callID = 24, confID = -1,
 streamType = TDM, streamDir = INPUT, ownerContext = 0x0000000C, codec = 0x0, fax_modem_type
 =0, XmitFn = 0x00000000, multicastStreamDtmfType = 0
*May 22 04:12:17.775: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: streamID = 2
*May 22 04:12:17.779: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: callID = 24, confID = -1,
 streamType = TDM, streamDir = OUTPUT, ownerContext = 0x0000000C, codec = 0x0, fax_modem_type
 =0, XmitFn = 0x00000000, multicastStreamDtmfType = 0
*May 22 04:12:17.779: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: streamID = 3
*May 22 04:12:17.779: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpReserveGwResource: sIdLegInput1 =
2, sIdLegOutput1 = 3, requesterCallID = 24, respFunc = 0x61CD1EE0
*May 22 04:12:17.787: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_new_or_existing_gw_session: created
 a new session = 0x657EEE20
*May 22 04:12:17.787: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_enlist_stream: Stream 2 is enlisted,
 total = 1
*May 22 04:12:17.787: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_enlist_stream: Stream 3 is enlisted,
 total = 2
*May 22 04:12:17.787: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_INITIAL, event:E_DSMP_CC_RESERVE_RESOURCE_REQ]
*May 22 04:12:17.791: CNFSM: cur_container:rsc_main_container, cur_state:S_DSMP_INITIAL,
event:E_DSMP_CC_RESERVE_RESOURCE_REQ
*May 22 04:12:17.791: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_gw_act_alloc_rsc:
*May 22 04:12:17.791: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_decide_pgm_based_stm:
*May 22 04:12:17.791: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:17.791: CNFSM: new_container:rsc_allocating_container
*May 22 04:12:17.791: CNFSM: next_state:S_DSMP_RSC_ALLOCATING
*May 22 04:12:17.791: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_alloc_succ:
*May 22 04:12:17.791: CNFSM: next_state:S_DSMP_RSC_ALLOCATED
*May 22 04:12:18.047: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpReserveGwResource: sIdLegInput1 =
2, sIdLegOutput1 = 3, requesterCallID = 24, respFunc = 0x61CD1EE0
*May 22 04:12:18.047: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_RSC_ALLOCATED, event:E_DSMP_CC_RESERVE_RESOURCE_REQ]
*May 22 04:12:18.051: CNFSM: cur_container:rsc_allocating_container,
cur_state:S_DSMP_RSC_ALLOCATED, event:E_DSMP_CC_RESERVE_RESOURCE_REQ
*May 22 04:12:18.051: CNFSM: new_container:rsc_main_container
*May 22 04:12:18.051: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_reopen:
*May 22 04:12:18.051: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:18.051: CNFSM: history stored state: S_DSMP_RSC_ALLOCATED, container:
rsc_allocating_container
*May 22 04:12:18.051: CNFSM: updated current container: rsc_main_container
*May 22 04:12:18.051: CNFSM: next_state:S_DSMP_RSC_REOPENING
*May 22 04:12:18.051: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_alloc_succ:
*May 22 04:12:18.051: CNFSM: restoring history state: S_DSMP_RSC_ALLOCATED
*May 22 04:12:18.055: CNFSM: restoring history container: rsc_allocating_container
*May 22 04:12:18.055: CNFSM: new_container:rsc_allocating_container
*May 22 04:12:18.055: CNFSM: next_state:S_DSMP_RSC_ALLOCATED
*May 22 04:12:18.055: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpStopPlay: sIdLegOutput = 3
*May 22 04:12:18.059: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_RSC_ALLOCATED, event:E_DSMP_CC_STOP_PLAY_REQ]
*May 22 04:12:18.059: CNFSM: cur_container:rsc_allocating_container,
cur_state:S_DSMP_RSC_ALLOCATED, event:E_DSMP_CC_STOP_PLAY_REQ
*May 22 04:12:18.059: CNFSM: new_container:rsc_main_container
*May 22 04:12:18.059: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_ignore:
*May 22 04:12:18.059: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:18.063: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: callID = 23, confID = 12,
 streamType = PACKET, streamDir = INPUT, ownerContext = 0x0000000C, codec = 0x1,
fax_modem_type =0, XmitFn = 0x6158E1F8, multicastStreamDtmfType = -1
*May 22 04:12:18.067: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: streamID = 1
*May 22 04:12:18.067: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: callID = 23, confID = 12,
 streamType = PACKET, streamDir = OUTPUT, ownerContext = 0x0000000C, codec = 0x1,
fax_modem_type =0, XmitFn = 0x6158E1F8, multicastStreamDtmfType = -1
*May 22 04:12:18.067: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpAddStream: streamID = 4
*May 22 04:12:18.071: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpUpdateRtpMode: sId = 1, callID = 24,
```

```
 mode = 1
*May 22 04:12:18.071: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpModifyReq: sIdLegInput = 1,
sIdLegOutput = 4, callID = 24
*May 22 04:12:18.075: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpConnectGwBinderReq:
   sIdLegInputTdm1=2, sIdLegOutputPak1=4, sIdLegInputPak2=1, sIdLegOutputTdm2=3, Call Id=24
*May 22 04:12:18.075: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpModifyReq: sIdLegInput = 2,
sIdLegOutput = 3, callID = 24FORKING Parameters are forking mask: 7,
simple_forking_codec_mask: 327679, complex_forking_codec_mask 327679
*May 22 04:12:18.075: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_process_event: No session is
associated to the streams.
*May 22 04:12:18.075: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_caps:
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_packet_stream_caps:
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
   CALL_ERROR; DSMP Session Is NULL
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:exit@2908
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
   CALL_ERROR; DSMP Session Is NULL
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:exit@2908
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
   CALL_ERROR; DSMP Session Is NULL
*May 22 04:12:18.079: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:exit@2908
*May 22 04:12:18.083: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.083: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
   CALL_ERROR; DSMP Session Is NULL
*May 22 04:12:18.083: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:exit@2908
*May 22 04:12:18.083: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_packet_stream_caps:
*May 22 04:12:18.083: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_tdm_stream_caps:
*May 22 04:12:18.083: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_tdm_stream_caps:exit@385
*May 22 04:12:18.087: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_tdm_stream_caps:
*May 22 04:12:18.087: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_tdm_stream_caps:exit@385
*May 22 04:12:18.087: //-1/F401BFC88006/DSMP:():-1/dsmp_enlist_stream: Stream 1 is enlisted,
 total = 3
*May 22 04:12:18.087: //-1/F401BFC88006/DSMP:():-1/dsmp_enlist_stream: Stream 4 is enlisted,
 total = 4
*May 22 04:12:18.087: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_RSC_ALLOCATED, event:E_DSMP_CC_CONNECT_REQ]
*May 22 04:12:18.091: CNFSM: cur_container:rsc_allocating_container,
cur_state:S_DSMP_RSC_ALLOCATED, event:E_DSMP_CC_CONNECT_REQ
*May 22 04:12:18.091: //-1/F401BFC88006/DSMP:():-1/dsmp_is_req_not_in_proc_dsp_ready:
*May 22 04:12:18.091: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_init:
*May 22 04:12:18.091: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:18.091: CNFSM: history stored state: S_DSMP_RSC_ALLOCATED, container:
rsc_allocating_container
*May 22 04:12:18.091: CNFSM: new_container:rsc_main_container
*May 22 04:12:18.095: CNFSM: next_state:S_DSMP_RSC_INITING
*May 22 04:12:18.095: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_decide_pgm_based_stm:
*May 22 04:12:18.095: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_init_succ: pgm_base_stm_id
 = 1, service_id = 26, codec =4
*May 22 04:12:18.095: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:18.099: CNFSM: next_state:S_DSMP_RSC_OPENING
*May 22 04:12:18.099: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_open_succ:
*May 22 04:12:18.099: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.099: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_static_params:
*May 22 04:12:18.099: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_static_params: servic_type=1
base stream=1 tdm_caps=0x6557E454 pkt_caps=0x6557EC3C
*May 22 04:12:18.099: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_static_params:
*May 22 04:12:18.103:  ip_tones = 0, pstn_tones = 0
*May 22 04:12:18.103: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_conf_static_params: voice detection
 disabled
*May 22 04:12:18.103: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_conf_static_params: silence detection
 disabled
*May 22 04:12:18.103: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:18.107: CNFSM: next_state:S_DSMP_RSC_STATIC_CONF
*May 22 04:12:18.107: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_start_service:
*May 22 04:12:18.107: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:18.107: CNFSM: next_state:S_DSMP_RSC_STARTING
```

```
*May 22 04:12:18.107: //-1/F401BFC88006/DSMP:():-1/dsmp_act_rsc_start_succ:
*May 22 04:12:18.107: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_dynamic_params:
*May 22 04:12:18.111: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_dynamic_params: servic_type=1
 base stream=1
*May 22 04:12:18.111: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.111: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.111: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_leg_voice_elog_write:
*May 22 04:12:18.111: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_detector_params:
*May 22 04:12:18.115: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:18.115: CNFSM: next_state:S_DSMP_RSC_DYNAMIC_CONF
*May 22 04:12:18.115: CNFSM: new_container:rsc_running_container
*May 22 04:12:18.115: CNFSM: next_state:S_DSMP_MCAST_CHECK
*May 22 04:12:18.115: //-1/F401BFC88006/DSMP:():-1/dsmp_is_mcast:
*May 22 04:12:18.119: //-1/F401BFC88006/DSMP:():-1/dsmp_is_mcast:exit@1487
*May 22 04:12:18.119: //-1/F401BFC88006/DSMP:():-1/dsmp_is_not_mcast:
*May 22 04:12:18.119: //-1/F401BFC88006/DSMP:():-1/dsmp_is_mcast:
*May 22 04:12:18.119: //-1/F401BFC88006/DSMP:():-1/dsmp_is_mcast:exit@1487
*May 22 04:12:18.119: //-1/F401BFC88006/DSMP:():-1/dsmp_is_not_mcast:exit@1500
*May 22 04:12:18.119: CNFSM: next_state:S_DSMP_DTMF_FSK_MODE_CHECK
*May 22 04:12:18.119: //-1/F401BFC88006/DSMP:():-1/dsmp_is_fsk_or_dtmf:
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_diagnostic:
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_diagnostic:exit@1587
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_not_dtmf_fsk:
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_fsk_or_dtmf:
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_diagnostic:
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_diagnostic:exit@1587
*May 22 04:12:18.123: //-1/F401BFC88006/DSMP:():-1/dsmp_is_not_dtmf_fsk:exit@1600
*May 22 04:12:18.127: CNFSM: next_state:S_DSMP_NOT_DTMF_FSK
*May 22 04:12:18.127: //-1/F401BFC88006/DSMP:():-1/dsmp_is_simple_voice:
*May 22 04:12:18.127: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_get_stm_service_type:
*May 22 04:12:18.127: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_simple_voice_start:
*May 22 04:12:18.127: //-1/F401BFC88006/DSMP:():-1/dsmp_connect_req_done:
*May 22 04:12:18.127: //-1/F401BFC88006/DSMP:():-1/dsmp_enlist_connection: Connection is
added, enlisted LegIn: 2, enlisted LegOut: 4
*May 22 04:12:18.131: //-1/F401BFC88006/DSMP:():-1/dsmp_enlist_connection: Connection is
added, enlisted LegIn: 1, enlisted LegOut: 3
*May 22 04:12:18.131: CNFSM: new_container:simple_voice_container
*May 22 04:12:18.131: CNFSM: next_state:S_DSMP_VC_RUNNING
*May 22 04:12:18.131: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_MODIFY_REQ]
*May 22 04:12:18.135: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_MODIFY_REQ
*May 22 04:12:18.135: CNFSM: new_container:rsc_running_container
*May 22 04:12:18.135: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_save_modify_caps:
*May 22 04:12:18.135: //-1/F401BFC88006/DSMP:():-1/dsmp_save_modify_caps:
*May 22 04:12:18.135: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_packet_stream_caps:
*May 22 04:12:18.139: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_packet_stream_caps:
*May 22 04:12:18.139: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_tdm_stream_caps:
*May 22 04:12:18.139: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_save_modify_tdm_stream_caps:
*May 22 04:12:18.139: //-1/F401BFC88006/DSMP:():-1/dsmp_save_stream_detectors:
*May 22 04:12:18.139: CNFSM: history stored state: S_DSMP_VC_RUNNING, container:
simple_voice_container
*May 22 04:12:18.139: CNFSM: updated current container: rsc_running_container
*May 22 04:12:18.143: CNFSM: next_state:S_DSMP_MODIFY_CAPS_SAVED
*May 22 04:12:18.143: //-1/F401BFC88006/DSMP:():-1/dsmp_is_modify_to_modem_passthru:
*May 22 04:12:18.143: //-1/F401BFC88006/DSMP:():-1/dsmp_is_static_params_changed:
*May 22 04:12:18.143: //-1/F401BFC88006/DSMP:():-1/dsmp_is_modify_to_modem_passthru:
*May 22 04:12:18.143: //-1/F401BFC88006/DSMP:():-1/dsmp_is_dynamic_params_changed:
*May 22 04:12:18.143: //-1/F401BFC88006/DSMP:():-1/dsmp_is_modify_to_modem_passthru:
*May 22 04:12:18.143: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_dynamic_params_changed:
*May 22 04:12:18.147: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_dynamic_params:
*May 22 04:12:18.147: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_dynamic_params: servic_type=1
 base stream=1
*May 22 04:12:18.147: //-1/F401BFC88006/DSMP:():-1/dsmp_conf_detector_params:
*May 22 04:12:18.147: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_conf_detector_params: fax: 1
*May 22 04:12:18.147: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmp_conf_detector_params: modem: 1
*May 22 04:12:18.151: CNFSM: restoring history state: S_DSMP_VC_RUNNING
*May 22 04:12:18.151: CNFSM: restoring history container: simple_voice_container
*May 22 04:12:18.151: CNFSM: new_container:simple_voice_container
*May 22 04:12:18.151: CNFSM: next_state:S_DSMP_VC_RUNNING
*May 22 04:12:18.163: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpUpdateRtpMode: sId = 1, callID = 24,
 mode = 3
```

```
*May 22 04:12:18.163: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpUpdateRtpMode: sId = 1, callID = 24,
 mode = 3
*May 22 04:12:30.947: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpUpdateRtpMode: sId = 1, callID = 24,
 mode = 3
```

The following is sample output from the **debug voip dsmp** command, with Cisco IOS Release 12.3(14)T software, when a VoIP call is in transition from connected to the disconnected state:

```
Router# debug voip dsmp


*May 22 04:12:30.951: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpDisconnectGwBinder: sIdLegInputTdm1
 = 2, sIdLegOutputPak1 = 4, sIdLegInputPak2 = 1, sIdLegOutputTdm2 = 3 requesterCallID = 24
*May 22 04:12:30.951: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpDeleteStream: streamID = 1, callID
= 23, ownerContext = 0x00000000
*May 22 04:12:30.951: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpDeleteStream: streamID = 4, callID
= 23, ownerContext = 0x00000000
*May 22 04:12:30.951: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_DISCONNECT]
*May 22 04:12:30.955: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_DISCONNECT
*May 22 04:12:30.955: CNFSM: new_container:rsc_running_container
*May 22 04:12:30.955: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_cc_disconnect:
*May 22 04:12:30.955: //-1/F401BFC88006/DSMP:():-1/dsmp_delist_connection: Connection is
delisted, delisted LegIn: 2, delisted LegOut: 4, total conn_count is = 1
*May 22 04:12:30.955: //-1/F401BFC88006/DSMP:():-1/dsmp_delist_connection: Connection is
delisted, delisted LegIn: 1, delisted LegOut: 3, total conn_count is = 0
*May 22 04:12:30.955: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:30.955: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM]
*May 22 04:12:30.959: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM
*May 22 04:12:30.959: CNFSM: new_container:rsc_running_container
*May 22 04:12:30.959: CNFSM: new_container:rsc_main_container
*May 22 04:12:30.959: //-1/F401BFC88006/DSMP:():-1/dsmp_not_last_stream:
*May 22 04:12:30.959: //-1/F401BFC88006/DSMP:():-1/dsmp_is_last_stream:
*May 22 04:12:30.959: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_not_last_stream:
*May 22 04:12:30.959: //-1/F401BFC88006/DSMP:():-1/remove_stream_from_DB:
*May 22 04:12:30.959: //-1/F401BFC88006/DSMP:():-1/process_software_multicast_streams:
*May 22 04:12:30.959: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:30.959: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM]
*May 22 04:12:30.959: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM
*May 22 04:12:30.963: CNFSM: new_container:rsc_running_container
*May 22 04:12:30.963: CNFSM: new_container:rsc_main_container
*May 22 04:12:30.963: //-1/F401BFC88006/DSMP:():-1/dsmp_not_last_stream:
*May 22 04:12:30.963: //-1/F401BFC88006/DSMP:():-1/dsmp_is_last_stream:
*May 22 04:12:30.963: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_not_last_stream:
*May 22 04:12:30.963: //-1/F401BFC88006/DSMP:():-1/remove_stream_from_DB:
*May 22 04:12:30.963: //-1/F401BFC88006/DSMP:():-1/process_software_multicast_streams:
*May 22 04:12:30.963: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:30.967: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpStopPlay: sIdLegOutput = 3
*May 22 04:12:30.975: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_STOP_PLAY_REQ]
*May 22 04:12:30.975: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_STOP_PLAY_REQ
*May 22 04:12:30.975: CNFSM: new_container:rsc_running_container
*May 22 04:12:30.975: CNFSM: new_container:rsc_main_container
*May 22 04:12:30.975: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_ignore:
*May 22 04:12:30.975: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:31.011: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpStopPlay: sIdLegOutput = 3
*May 22 04:12:31.011: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpDeleteStream: streamID = 2, callID
= 24, ownerContext = 0x0000000C
*May 22 04:12:31.011: //-1/xxxxxxxxxxxx/DSMP:():-1/dsmpDeleteStream: streamID = 3, callID
= 24, ownerContext = 0x0000000C
*May 22 04:12:31.015: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_STOP_PLAY_REQ]
*May 22 04:12:31.015: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_CC_STOP_PLAY_REQ
*May 22 04:12:31.015: CNFSM: new_container:rsc_running_container
*May 22 04:12:31.015: CNFSM: new_container:rsc_main_container
```

```
*May 22 04:12:31.015: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_ignore:
*May 22 04:12:31.015: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:31.015: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM]
*May 22 04:12:31.015: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM
*May 22 04:12:31.019: CNFSM: new_container:rsc_running_container
*May 22 04:12:31.019: CNFSM: new_container:rsc_main_container
*May 22 04:12:31.019: //-1/F401BFC88006/DSMP:():-1/dsmp_not_last_stream:
*May 22 04:12:31.019: //-1/F401BFC88006/DSMP:():-1/dsmp_is_last_stream:
*May 22 04:12:31.019: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_not_last_stream:
*May 22 04:12:31.019: //-1/F401BFC88006/DSMP:():-1/remove_stream_from_DB:
*May 22 04:12:31.019: //-1/F401BFC88006/DSMP:():-1/process_software_multicast_streams:
*May 22 04:12:31.019: CNFSM: next_state:CNFSM_NO_STATE_CHANGE
*May 22 04:12:31.019: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM]
*May 22 04:12:31.019: CNFSM: cur_container:simple_voice_container,
cur_state:S_DSMP_VC_RUNNING, event:E_DSMP_DELETE_STREAM
*May 22 04:12:31.019: CNFSM: new_container:rsc_running_container
*May 22 04:12:31.023: CNFSM: new_container:rsc_main_container
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/dsmp_not_last_stream:
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/dsmp_is_last_stream:
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/dsmp_is_last_stream:
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_stop:
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/remove_stream_from_DB:
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/process_software_multicast_streams:
*May 22 04:12:31.023: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:31.023: CNFSM: history stored state: S_DSMP_VC_RUNNING, container:
simple_voice_container
*May 22 04:12:31.023: CNFSM: updated current container: rsc_main_container
*May 22 04:12:31.027: CNFSM: new_container:rsc_closing_container
*May 22 04:12:31.027: CNFSM: next_state:S_DSMP_RSC_STOPPING
*May 22 04:12:31.027: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_rsc_stopping_close:
*May 22 04:12:31.027: //-1/F401BFC88006/DSMP:():-1/dsmp_set_state_var: Message to DSP is
successful
*May 22 04:12:31.027: CNFSM: next_state:S_DSMP_DSPMGR_CLOSING
*May 22 04:12:31.027: //-1/F401BFC88006/DSMP:():-1/dsmp_exec:
   [state:S_DSMP_DSPMGR_CLOSING, event:E_DSMP_DSPRM_CLOSE_COMPLETE]
*May 22 04:12:31.027: CNFSM: cur_container:rsc_closing_container,
cur_state:S_DSMP_DSPMGR_CLOSING, event:E_DSMP_DSPRM_CLOSE_COMPLETE
*May 22 04:12:31.027: //-1/F401BFC88006/DSMP:():-1/dsmp_gw_act_dspmgr_closing_complete:
*May 22 04:12:31.031: //-1/F401BFC88006/DSMP:():-1/remove_stream_from_DB:
*May 22 04:12:31.031: //-1/F401BFC88006/DSMP:():-1/process_software_multicast_streams:
*May 22 04:12:31.031: //-1/F401BFC88006/DSMP:():-1/remove_stream_from_DB:
*May 22 04:12:31.031: //-1/F401BFC88006/DSMP:():-1/process_software_multicast_streams:
*May 22 04:12:31.031: CNFSM: next_state:CNFSM_NULL_STATE
*May 22 04:12:31.031: //-1/F401BFC88006/DSMP:():-1/dsmp_free_session:
```

Related Commands

Command	Description
debug voip dsm	Displays debugging information from the DSM subsystem.
debug voip vtsp	Displays information about the VTSP.

debug voip dspapi

To troubleshoot the digital signal processor (DSP) application programming interface (API), use the **debug voip dspapi** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip dspapi [**all**| **command**| **default**| **detail**| **error** [**call [informational]**| **software [informational]**]| **function**| **inout**| **notification**| **response**]

no debug voip dspapi

Syntax Description

all	(Optional) Displays all DSP API debugging messages.
command	(Optional) Displays DSP API commands.
default	(Optional) Displays DSP API detail, error, and inout debugging messages. This option also runs if no keywords are added.
detail	(Optional) Displays detailed information about commands sent to the DSP. This command is used in conjunction with other **debug voip dspapi** commands to show additional details when you use the **command**, **notification**, and **response** keywords.
error	(Optional) Displays DSP API errors.
call	(Optional) Displays call processing errors.
informational	(Optional) Displays minor errors and major errors. Without the **informational** keyword, only major errors are displayed.
software	(Optional) Displays software processing errors.
function	(Optional) Displays DSP API functions.
inout	(Optional) Displays output for the **command**, **notification**, and **response** keywords.
notification	(Optional) Displays DSP API notification messages.
response	(Optional) Displays DSP API response messages.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command replaces the **debug dspapi**command.
12.3(14)T	T.38 fax relay call statistics were made available to Call Detail Records (CDRs) through Vendor-Specific Attributes (VSAs) and added to the call log.

Usage Guidelines

DSP API message events used to communicate with DSPs are intended for use with Connexant (NextPort) and Texas Instruments (54x) DSPs.

Caution This command severely impacts performance and should be used only for single-call debug capture.

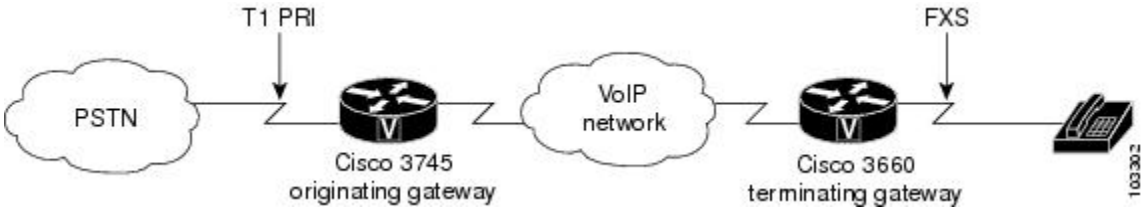
Examples

The following examples show output for variations of the **debug voip dspapi** command:

- debug voip dspapi, on page 56
- debug voip dspapi, on page 56
- debug voip dspapi, on page 56

For these examples, the topology shown in the figure below is used.

Figure 4: Network Topology for debug voip dspapi Examples



Examples

```
Router# debug voip dspapi command
voip dspapi command debugging is ON
Router#
*Apr 18 21:33:48.347: //-1/CD89F6A78020/DSPAPI/[2/1:23]/dsp_init:
*Apr 18 21:33:48.347: //-1/CD89F6A78020/DSPAPI/[2/1:23]/dsp_voice_config_params:
*Apr 18 21:33:48.347:     9 parameters
```

The following lines show encapsulation settings, jitter, inband signaling, echo cancellation, gain, and other quality of service (QoS) settings:

```
   [0] ENCAP RTP: Tx SSRC=0, Rx SSRC=0, Tx VPXCC=0, Rx VPXCC=0
    IFP payload type=122, SID support=1, TSE payload=101, Sequence number start=0
    Redundancy=0, ClearChannel payload type=0, Fax payload type=0
    Alaw PCM switchover=0, MuLAW PCM switchover=0
    DTMF payload type=0, NTE receive payload type=101
    Dynamic payload=0, Codec=g711ulaw
    [1] PO_JITTER: mode=2 initial=60(ms) max=200(ms) min=40(ms) fax_nom=300(ms)
    [2] INBAND_SIG: mode=0x1 enable
    [3] ECHO_CANCEL: Flags=0x37, Echo length=64(ms)
    [4] IDLE_CODE_DET: Enable=0, Code=0x0, Duration=6000(ms)
    [5] GAIN: Input=0.0(dB), Output=0.0(dB)
    [6] CNG: 1
    [7] INFO_FIELD_SIZE: 160 bytes
    [8] DIGIT_RELAY: 2
*Apr 18 21:33:48.359: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_voice_get_capabilities:
*Apr 18 21:33:48.359: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_stop_service:
    NONE (0)
*Apr 18 21:33:48.359: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_open_service:
    G729IETF (25)
*Apr 18 21:33:48.359: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_voice_config_params:
```

The following lines show settings for encapsulation, packet suppression, and voice activity detection (VAD):

```
*Apr 18 21:33:48.359:     3 parameters
    [0] ENCAP RTP: Tx SSRC=64, Rx SSRC=0, Tx VPXCC=0, Rx VPXCC=0
    IFP payload type=122, SID support=19, TSE payload=101, Sequence number start=3238
    Redundancy=0, ClearChannel payload type=125, Fax payload type=122
    Alaw PCM switchover=8, MuLAW PCM switchover=0

 DTMF payload type=121, NTE receive payload type=101
    Dynamic payload=0, Codec=g729r8
    [1] PAK_SUPPRESS: 0
    [2] VAD: Enable=1, Threshold=-38(dBm)
.
.
.
*Apr 18 21:33:48.363: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_start_service:
    G729IETF (25)
*Apr 18 21:33:48.363: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_voice_config_params:
*Apr 18 21:33:48.363:     1 parameter
    [0] VAD: Enable=1, Threshold=-38(dBm)
*Apr 18 21:33:50.867: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_query_info:
    Request ID=1, Reset Flag=FALSE Q:PO_Delay PO_Error TX RX
.
.
.
*Apr 18 21:34:15.031: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_query_info:
    Request ID=5, Reset Flag=TRUE Q:Error
*Apr 18 21:34:15.031: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_query_info:
    Request ID=5, Reset Flag=TRUE Q:Levels
*Apr 18 21:34:15.031: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_inband_tone_off:
*Apr 18 21:34:15.031: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_stop_service:
    G729IETF (25)
*Apr 18 21:34:15.031: //66/CD89F6A78020/DSPAPI/[2/1:23:66]/dsp_close_service:
    G729IETF (25)
```

Examples

```
Router# debug voip dspapi inout

voip dspapi inout debugging is ON
*May  1 19:59:15.579: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_init:
*May  1 19:59:15.579: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_voice_config_params:
*May  1 19:59:15.579:     9 parameters
```

The following lines show encapsulation settings, jitter, inband signalling, echo cancellation, gain, and other quality of service (QoS) settings:

```
    [0] ENCAP RTP: Tx SSRC=0, Rx SSRC=0, Tx VPXCC=0, Rx VPXCC=0
    IFP payload type=122, SID support=1, TSE payload=101, Sequence number start=0
    Redundancy=0, ClearChannel payload type=0, Fax payload type=0
    Alaw PCM switchover=0, MuLAW PCM switchover=0
```

```
    DTMF payload type=0, NTE receive payload type=101
    Dynamic payload=0, Codec=g711ulaw
    [1] PO_JITTER: mode=2 initial=60(ms) max=200(ms) min=40(ms) fax_nom=300(ms)
    [2] INBAND_SIG: mode=0x1 enable
    [3] ECHO_CANCEL: Flags=0x17, Echo length=8(ms)
    [4] IDLE_CODE_DET: Enable=0, Code=0x0, Duration=6000(ms)
    [5] GAIN: Input=0.0(dB), Output=-6550.6(dB)
    [6] CNG: 1
    [7] INFO_FIELD_SIZE: 160 bytes
    [8] DIGIT_RELAY: 2
.
.
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_voice_get_capabilities:
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_stop_service:
    NONE (0)
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_open_service:
    G729IETF (25)
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_voice_config_params:
*May  1 19:59:15.587:     3 parameters
    [0] ENCAP RTP: Tx SSRC=0, Rx SSRC=0, Tx VPXCC=0, Rx VPXCC=0
    IFP payload type=122, SID support=19, TSE payload=101, Sequence number start=2977
    Redundancy=0, ClearChannel payload type=125, Fax payload type=122
    Alaw PCM switchover=8, MuLAW PCM switchover=0
    DTMF payload type=121, NTE receive payload type=101
    Dynamic payload=0, Codec=g729r8
    [1] PAK_SUPPRESS: 0
    [2] VAD: Enable=1, Threshold=-38(dBm)
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_idle_service:
    G729IETF (25)
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_voice_config_params:
*May  1 19:59:15.587:     3 parameters
    [0] INFO_FIELD_SIZE: 20 bytes
    [1] ENCAP RTP: Tx SSRC=64, Rx SSRC=0, Tx VPXCC=0, Rx VPXCC=0
    IFP payload type=122, SID support=19, TSE payload=101, Sequence number start=2977
    Redundancy=0, ClearChannel payload type=125, Fax payload type=122
    Alaw PCM switchover=8, MuLAW PCM switchover=0
    DTMF payload type=121, NTE receive payload type=101
    Dynamic payload=0, Codec=g729r8
    [2] DIGIT_RELAY: 2
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_start_service:
    G729IETF (25)
*May  1 19:59:15.587: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_voice_config_params:
*May  1 19:59:15.587:     1 parameter
    [0] VAD: Enable=1, Threshold=-38(dBm)
*May  1 19:59:15.591: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_inband_tone_on:
    Tone ID=1, Direction=2, Num frequencies=2
    Frequency(hz): a=440 b=480, Amplitude(dB): a=-16.0 b=-16.0
    Cadence1(ms): ON=2000 OFF=4000
    Cadence2(ms): ON=0 OFF=0
    Cadence3(ms): ON=0 OFF=0
    Cadence4(ms): ON=0 OFF=0
    Frequency(hz): a2=25667 b2=51816 a3=0 b3=1 a4=24596 b4=52484
    ITO option group: 1
*May  1 19:59:17.195: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_inband_tone_off:
01:19:04: %PS-3-MULTFAIL: There is more than one failure with the  Power System 1; please
resolve problems immediately
```

The following statistics repeat for each DSP query. The transmit (tx) and receive (rx) statistics show number of packets, comfort noise settings, duration, and packet status.

```
*May  1 19:59:18.051: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_query_info:
    Request ID=1, Reset Flag=FALSE Q:PO_Delay PO_Error TX RX
*May  1 19:59:18.051: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_tx_stats:
    Request ID=1, Packets: Voice=113, Signaling=0, ComfortNoise=1
    TX duration=2460(ms): Voice=2260(ms), FAX=0(ms)
*May  1 19:59:18.051: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_rx_stats:
    Request ID=1, Packets: Voice=33, Signalling=0, ComfortNoise=1
    RX duration=840(ms): Voice=640(ms), FAX=0(ms)
    Packets: Bad Sequence=0, Bad Protocol=0, Late=0, Early=1
*May  1 19:59:18.051: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_playout_delay_stats:
    Request ID=1, Current=70(ms), MIN=70(ms), MAX=70(ms)
    Clock offset=80(ms), Inter arrival jitter=67082420(ms)
```

```
*May  1 19:59:18.051: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_playout_error_stats:
    Request ID=1
    Concealment: Predictive=0(ms), Interpolative=0(ms), Silence=0(ms)
    Retro Memory update=0(ms), Buffer overflow=10(ms)
    Talkspurt endpoint detection errors=0
*May  1 19:59:19.827: //67/2BA0E0758024/DSPAPI/[4/0/0 (67)]/dsp_query_info:
    Request ID=1, Reset Flag=FALSE Q:PO_Delay PO_Error TX RX
.
.
.
```

Examples

This output shows the fax relay statistics.

```
Router# debug voip dspapi
voip dspapi debugging is ON
.
.
.
May  7 21:32:16.472 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_playout_error_stats:
    Request ID=1
    Concealment: Predictive=0(ms), Interpolative=0(ms), Silence=0(ms)
    Retro Memory update=0(na)(ms), Buffer overflow=0(ms)
    Talkspurt endpoint detection errors=0
May  7 21:32:18.996 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_query_info:
    Request ID=1, Reset Flag=FALSE Q:PO_Delay PO_Error TX RX
May  7 21:32:18.996 UTC: np_vsmgr_dispatch_voice_rsp(1/3): VOICE_LINK_INFO_RSP_NTF Received

May  7 21:32:18.996 UTC: request_id = 0x01, request_type = 0x0F
May  7 21:32:18.996 UTC: VOICE_TRANSMIT_STATS(1/3): num_voice_packets 36 num_sig_packets 0
 num_cn_packets 1 transmit_duration AD2 end_point_detection 0
May  7 21:32:18.996 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_tx_stats:
    Request ID=1, Packets: Voice=54, Signaling=0, ComfortNoise=1
    TX duration=2770(ms): Voice=0(ms), FAX=0(na)(ms)
May  7 21:32:18.996 UTC: VOICE_RECEIVE_STATS(1/3): num_voice_packets 20 num_sig_packets 0
num_cn_packets 2 receive_duration AD2  voice_receive_duration 0 num_pos_packets 0
num_bph_packets 0 num_late_packets 0 num_early_packets 0
May  7 21:32:18.996 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_rx_stats:
    Request ID=1, Packets: Voice=32, Signalling=0, ComfortNoise=2
    RX duration=2770(ms): Voice=0(ms), FAX=0(na)(ms)
    Packets: Bad Sequence=0, Bad Protocol=0, Late=0, Early=0
May  7 21:32:18.996 UTC: VOICE_PLAYOUT_DELAY_STATS(1/3): curr_playout_delay 5A
min_playout_delay 5A max_playout_delay 5A clock offset 2F07E72
May  7 21:32:19.000 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_playout_delay_stats:
    Request ID=1, Current=90(ms), MIN=90(ms), MAX=90(ms)
    Clock offset=49315442(ms), Inter arrival jitter=0(na)(ms)
May  7 21:32:19.000 UTC: VOICE_PLAYOUT_ERROR(1/3): pred_conceal 0x0 inter_conceal 0x0
silence_conceal 0x0 buffer_overflow 0x0 endpt_det_error 0x0
May  7 21:32:19.000 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_playout_error_stats:
    Request ID=1
    Concealment: Predictive=0(ms), Interpolative=0(ms), Silence=0(ms)
    Retro Memory update=0(na)(ms), Buffer overflow=0(ms)
    Talkspurt endpoint detection errors=0
May  7 21:32:21.456 UTC: VOICE_DET_STATUS_CHANGE_NTF(1/3): detector mask: 1 timestamp
51709BF8
May  7 21:32:21.456 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_voice_det_status_change:
    Status=1, Timestamp=1366334456, Tone ID=0, Trigger=TRUE
May  7 21:32:21.464 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_voice_config_params:
May  7 21:32:21.464 UTC:     1 parameter
    [0] PAK_SUPPRESS: 1
May  7 21:32:21.464 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_stop_service:
    G729IETF (26)
May  7 21:32:21.464 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_config_params:
    1 parameters
May  7 21:32:21.464 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_open_service:
    FAX_RELAY (27)
May  7 21:32:21.464 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_is_call_pending:
    Call is not PENDING
May  7 21:32:21.464 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_set_call_pending:
    Set PENDING state
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_query_link_info:
```

```
    Request ID=0, Group ID=1
May  7 21:32:21.504 UTC: vsm(1/3): np_vsmgr_voice_state_change() - state IDLE
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_is_call_pending:
    Call is PENDING
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_reset_call_pending:
    Reset PENDING state
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_call_status:
    Status=PENDING_SUCCESS
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_config_params:
    11 parameters
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_start_service:
    FAX_RELAY (27)
May  7 21:32:21.504 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_set_call_pending:
    Set PENDING state
May  7 21:32:22.556 UTC: vsm(1/3): np_vsmgr_voice_state_change() - state ACTIVE
May  7 21:32:22.556 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_is_call_pending:
    Call is PENDING
May  7 21:32:22.556 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_reset_call_pending:
    Reset PENDING state
May  7 21:32:22.556 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_call_status:
    Status=PENDING_SUCCESS
May  7 21:32:22.564 UTC: FAX_RELAY_LINK_INFO_RSP_NTF: slot 1 port 3 timestamp 76082770
fr-entered (20ms)
May  7 21:32:22.564 UTC: chan_id [3/1:D (8)] np_vsmgr_fax_relay_link_info_response:
May  7 21:32:29.712 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x4
May  7 21:32:30.436 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x2
May  7 21:32:30.784 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x1
May  7 21:32:33.936 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x42
May  7 21:32:34.280 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x41
May  7 21:32:39.676 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x21
May  7 21:32:39.676 UTC: np_fax_relay_t30_decode : Rx Direction
May  7 21:32:39.736 UTC: FARELAY_INIT_HS_MOD : 0x8
May  7 21:33:10.385 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x7D
May  7 21:33:13.073 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x31
May  7 21:33:15.217 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_t30_decode:
    T30 msg : 0x5F
May  7 21:33:16.073 UTC: FAX_RELAY_DET_STATUS_CHANGE: slot: 1 port: 3 detector mask 0x2
May  7 21:33:16.073 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_det_status:
    Status=2, Timestamp=716372818
May  7 21:33:16.073 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_query_info:
    Request ID=5, Reset Flag=FALSE Q:FaxRelay
May  7 21:33:16.097 UTC: FAX_RELAY_DATA_PUMP_STATS(1/3) - valid:0x3FFC1F55 state_code:0x1
level:0x18 phase_jitter:0x0 freq_offset:0x0 eqm:0x7FFE jit_depth:0x38B jit_buf_ov:0x0
tx_paks:0x5A rx_pkts:0x62C inv_pkts:0x0 oos_pkts:0x0 hs_mod:0x8 init_hs_mod:0x8 tx_pgs:0x0
 rx_pgs:0x1 ecm:0x1 nsf_country:0x0 nsf_manuf_len:0x20
nsf_manuf:0031B8EE80C48511DD0D0000DDDD0000DDDD00000000000000022ED00B0A400 encap:0x1
pkt_loss_con:0x0
May  7 21:33:16.097 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_fax_relay_stats:
    Request ID=5, MAX jitter depth=907, MAX net RX qdepth=0(na)
    Jitter buffer overflow=0, Net RX qoverflow=0(na)
    Packets: TX=90 TX drops=0(na)
    Packets: RX=1580 RX loss=0(na), RX invalid=0, RX OOSequence=0
    HS modulation=8, Pages: TX=0 RX=1
    MAX TX In qdepth=0(na), MAX RX Out qdepth=0(na)
    MAX HS buffer usage=0(na), TX In qoverflow=0(na), RX Out qoverflow=0(na)
    FAX: State=1, level=24, Phase jitter=0, Frequency offset=0, EQM=32766
    Initial HS modulation=8, Fax Direction=2, ECM Enabled=1
    NSF Countery Code=0, nsf_manuf_code[32]=0031
    Encapsulation Protocol=1, Pkt Loss Conceal=0
May  7 21:33:16.101 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_stop_service:
    FAX_RELAY (27)
May  7 21:33:16.101 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_open_service:
    G729IETF (26)
May  7 21:33:16.101 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_is_call_pending:
    Call is not PENDING
```

```
May  7 21:33:16.101 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_set_call_pending:
    Set PENDING state
May  7 21:33:16.985 UTC: FAX_RELAY_LINK_INFO_RSP_NTF: slot 1 port 3 timestamp 76518179
fr-end
May  7 21:33:17.001 UTC: vsm(1/3): np_vsmgr_voice_state_change() - state IDLE
May  7 21:33:17.001 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_is_call_pending:
    Call is PENDING
May  7 21:33:17.001 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_reset_call_pending:
    Reset PENDING state
May  7 21:33:17.001 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_call_status:
    Status=PENDING_SUCCESS
May  7 21:33:17.001 UTC: //8/D6635DD58005/DSPAPI/[1/0:3]/dsp_voice_config_params:
May  7 21:33:17.001 UTC:     4 parameters
.
.
.
```

Related Commands

Command	Description
debug voip hpi	Enables debugging for HPI message events.

debug voip dump-file-acct

To display debugging messages related to file accounting flushing processes, use the **debug voip dump-file-acct** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip dump-file-acct

no debug voip dump-file-acct

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging of file accounting processes is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(15)XY	This command was introduced.
12.4(20)T	This command was integrated into Cisco IOS Release 12.4(20)T.

Usage Guidelines

This command displays event and error information about the file accounting processes for flushing the buffer and writing the call detail records (CDRs) to the file.

Examples

The following example displays output from the **debug voip dump-file-acct**command:

```
Router# debug voip dump-file-acct

*May 10 06:31:10.187: voice_file_acct_write:
*May 10 06:31:10.187: file_acct_write_local: file accounting buffer overflow,dumping to
file
*May 10 06:31:10.187: voice_file_acct_initiate_dump_to_file: ctx_id=2, url=flash:cdr
*May 10 06:31:10.187: voice_file_acct_lock, ctx_id=2, refcnt=2
*May 10 06:31:10.187: create_file_acct_buffer: buffer of 63k created from chunk 0x46B5F474
*May 10 06:31:10.187: file_acct_write_local: message (len=640) written to file_acct:
30080 bytes left
*May 10 06:31:10.191: handle_file_acct_dump_request
*May 10 06:31:10.191: handle_file_acct_dump_request: pick up dump request (ctx_id=2)
*May 10 06:31:10.191: open_file_acct_dump_file:
url=flash:cdr_ragdenCME1_05_10_2007_06_30_28.191  < == shows url.
*May 10 06:31:10.215:  Secondary mode file acct is successful
*May 10 06:31:10.215: handle_file_acct_dump_request :to_write is 29748  <== shows how much
 is written to.
*May 10 06:31:10.219: : File accounting,write successful to file
*May 10 06:31:10.219: handle_file_acct_dump_request :to_write is 640
*May 10 06:31:10.219: : File accounting,write successful to file
*May 10 06:31:10.323: voice_file_acct_unlock, ctx_id=2  refcnt=1
```

Related Commands

Command	Description
debug voip fileacct	Displays debugging messages related to generating attributes for file accounting.
gw-accounting	Enables an accounting method for collecting CDRs.
primary	Sets the primary location for storing the CDRs generated for file accounting.

debug voip eddri

To turn on debugging for the event dispatcher and data repository interface (EDDRI), use the **debug voip eddri**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip eddri {**event**| **timers**| **prefix**| **all**}

no debug voip eddri {**event**| **timers**| **prefix**| **all**}

Syntax Description

event	Turns on debugging for EDDRI events.
timers	Turns on debugging for EDDRI timers.
prefix	Turns on debugging for the prefix database.
all	Turns on debugging all EDDRI activities.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(1)	This command was introduced.

Usage Guidelines

There is always a performance penalty when using **debug** commands.

The EDDRI notifies TGREP when an attribute changes on some subsystems. EDDRI interacts with the dial peer subsystem, the trunk group subsystems, call control API (CCAPI) subsystem and the customer relationship management (CRM) subsystem to notify changes in particular attributes. EDDRI is responsible for creating the prefix database.

Examples

The following example shows sample output from the **debug voip eddri**command:

```
21:00:53: eddri_interesting_ac_pt: new AC_curr 22 FD_curr -5 SD_curr -5
21:00:53: eddri_interesting_ac_pt: percent trigger diff 4
21:00:53: eddri_interesting_ac_pt: Interesting Point
21:00:53: eddri_send_prefix_event_to_clients : reason 0x40 num_prefix 1
```

With the send prefix event the available circuits value and the triggers for reporting are updated.

```
21:00:53: eddri_send_prefix_event_to_clients attr 0xFF ev_id 1 qid 0x64209230 reason 0x40
eddri_dequeue_event : dequeue event
21:00:53: eddri_interesting_ac_pt : tc 23 IAC 22 lwm 5 hwm 50 pct_trigger 2 oneMinusW 933
```

```
21:00:53: eddri_interesting_ac_pt: old AC_curr 23 FD_curr 0 SD_curr 0
21:00:53: eddri_interesting_ac_pt: new AC_curr 22 FD_curr -5 SD_curr -5
21:00:53: eddri_interesting_ac_pt: percent trigger diff 4
21:00:53: eddri_interesting_ac_pt: Interesting Point
21:00:53: eddri_send_prefix_event_to_clients : reason 0x40 num_prefix 1
```

Related Commands

Command	Description
debug tgrep error	Turns on debugging for any errors in functioning.
debug tgrep events	Turns on debugging for main events occurring throughout the subsystem.
debug tgrep fsm	Turns on debugging for FSM activity.
debug tgrep io	Turns on debugging for detailed socket level activities.
debug tgrep messages	Turns on debugging for the movement of TGREP messages.
debug tgrep msgdump	Turns on debugging for the dump of the details of TGREP messages.
debug tgrep timer-event	Turns on debugging for events that are related to the timer.
debug tgrep timers	Turns on debugging for timer activity.
debug tgrep tripr	Turns on debugging for the TRIP Reporter.
show voice eddri prefix	Shows applicable prefixes for the EDDRI.

debug voip enum

To view Voice over IP (VoIP) telephone number mapping (ENUM) information, use the **debug voip enum** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip enum {**detail**| **summary**}

no debug voip enum {**detail**| **summary**}

Syntax Description

detail	Displays detailed output.
summary	Displays summary output.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

Disable console logging and use buffered logging before using the **debug voip enum** command. Using the **debug voip enum** command generates a large volume of debugs, which can affect router performance.

Examples

The following shows sample output from the **debug voip enum detail** command:

The output shows the match number as 5108891234, enum table as 10. Rule 1 in table 10 matched the pattern and after applying the replacement rule, the resulting string is 5108891234. The enum query is sent out for the domain 4.3.2.1.9.8.8.0.1.5.e164.cisco.com. The output then shows the matching Naming Authority Pointer (NAPTR) records obtained in the response. The records are then processed and the final URLs (contact lists) are shown toward the end.

```
Router# debug voip enum detail
enum_resolve_domain:match_num 5108891234 table_indx 10
enum_resolve_domain:rule 1 result string 5108891234
generate_enum_search_string :search string 4.3.2.1.9.8.8.0.1.5.e164.cisco.com
enum_dns_query:name = 4.3.2.1.9.8.8.0.1.5.e164.cisco.com type = 35, ns_server = 0
order 100 pref 10 service sip+E2U flag U
regexp /^.*$/sip:5108891234@1.8.50.14/ replacement
order 200 pref 10 service h323+E2U flag U
regexp /^.*$/h323:5555@1.5.1.1/ replacement
num_elem = 2
NAPTR Record :order 100 pref 10 service sip+E2U
              flags U regexp /^.*$/sip:5108891234@1.8.50.14/
              replacement
```

```
NAPTR Record :order 200 pref 10 service h323+E2U
               flags U regexp /^.*$/h323:5555@1.5.1.1/
               replacement
decode_naptr_record :re_string ^.*$
decode_naptr_record :re_substitution_string sip:5108891234@1.8.50.14
decode_naptr_record :re_flags_string
U_FLAG case, stopping query
new_e164_user sip:5108891234@1.8.50.14
decode_naptr_record :re_string ^.*$
decode_naptr_re
tahoe13#cord :re_substitution_string h323:5555@1.5.1.1
decode_naptr_record :re_flags_string
U_FLAG case, stopping query
new_e164_user h323:5555@1.5.1.1
contact_list :
              sip:5108891234@1.8.50.14
contact_list :
              h323:5555@1.5.1.1
enum_resolve_domain:contact_list 64558450
```

A sample output of the **debug voip enum summary** command is shown below.

The output shows the matching number, the enum table used and the rule in the table that matched the number along with the resulting string. Note that this output is a subset of the output from **debug voip enum detail** command.

```
Router# debug voip enum summary
enum_resolve_domain:match_num 5108891234 table_indx 10
enum_resolve_domain:rule 1 result string 5108891234
```

The table below provides an alphabetical listing of the **debug voip enum** command fields and a description of each field.

Table 3: debug voip enum Field Descriptions

Field	Description
contact_list	Final list of URLs that the gateway will try to contact as an attempt to place the call.
flag	Flag value of a NAPTR record as defined in RFC 2915.
match_num	Number to be used for matching against the enum match table.
name	Fully qualified domain name sent out to Domain Name System (DNS) server
ns_server	Address of the DNS server. If 0, the DNS server configured on the gateway is used.
num_elem	Number of records received in the response.
order	Order in the record, as defined in RFC 2915.
pref	Preference of the record, as defined in RFC 2915.
regexp	Regular expression of the record, as defined in RFC 2915.

Field	Description
replacement	Replacement string of the record, as defined in RFC 2915.
re_flags_string	Flag indicating whether matching and replacement should be case sensitive: • i = Case insensitive • otherwise = Case sensitive
re_string	The first part of the regexp, delimited by "/". This is used to match the incoming string. Refer to RFC 2915.
re_substitution_string	The second part of regexp, delimited by "/".
result string	String that results when match_num is taken through the enum match table for a match. This string will be used to form a FQDN.
rule	Rule number that matched match_num in the enum match table.
search string	String sent out to the DNS server.
service	Service field of the NAPTR record. Refer to RFC 2915.
table_indx	Index of the enum match table picked for this call.
type	Type of record requested in the query: 35 = NAPTR 33 = DNS Service (SRV)

Related Commands

Command	Description
rule (ENUM configuration)	Defines the rule pattern for an ENUM match table.
show voice enum-match-table	Displays the ENUM match table rules.
test enum	Tests the ENUM match table rules.
voice enum-match-table	Initiates the ENUM match table definition.

debug voip event-log

To enable debugging of the event log module, use the **debug voip event-log**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip event-log

no debug voip event-log

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.

Examples

The following is sample output from the **debug voip event-log** command:

```
Router# debug voip event-log

voip event-log debugging is on
Router#
*Jul 18 22:22:45.417: voice_elog_open: ctx_id=1F, size=4, url=
*Jul 18 22:22:45.417: lock_elog, ctx_id=1F, refcnt=1
*Jul 18 22:22:45.417: voice_elog_write:
*Jul 18 22:22:45.417: elog_write_local: message (len=143) written to elog:
1F:1058566965:584:INFO: Call setup indication received, called = 4085550198,  calling =
52927, echo canceller = enable, direct inward dialing 3953 bytes left
*Jul 18 22:22:45.417: voice_elog_write:
*Jul 18 22:22:45.417: elog_write_local: message (len=38) written to elog:
1F:1058566965:585:INFO: Dialpeer = 1 3915 bytes left
*Jul 18 22:22:45.421: voice_elog_open: ctx_id=B, size=4, url=
*Jul 18 22:22:45.421: lock_elog, ctx_id=B, refcnt=1
*Jul 18 22:22:45.421: voice_elog_write:
*Jul 18 22:22:45.421: elog_write_local: message (len=114) written to elog:
B:1058566965:586:INFO: Session started for App-type = generic, URL =
tftp://demo/scripts/master/generic.vxml 3982 bytes left
*Jul 18 22:22:45.421: voice_elog_write:
*Jul 18 22:22:45.421: elog_write_local: message (len=69) written to elog:
B:1058566965:587:INFO: Incoming Telephony call received, LegID = 1F 3913 bytes left
*Jul 18 22:22:45.421: voice_elog_write:
*Jul 18 22:22:45.421: elog_write_local: message (len=89) written to elog:
B:1058566965:588:INFO: LegID = 1F: Calling = 4085550198, called = 52927, dial peer = 1 3824
 bytes left
*Jul 18 22:22:45.421: voice_elog_write:
*Jul 18 22:22:45.421: elog_write_local: message (len=66) written to elog:
B:1058566965:589:INFO: LegID = 1F: Leg State = LEG_INCCONNECTED 3758 bytes left
*Jul 18 22:22:45.433: voice_elog_write:
*Jul 18 22:22:45.437: elog_write_local: message (len=42) written to elog:
1F:1058566965:590:INFO: Digit collection 3873 bytes left
*Jul 18 22:22:45.437: voice_elog_write:
*Jul 18 22:22:45.437: elog_write_local: message (len=57) written to elog:
1F:1058566965:591:INFO: Call connected using codec None 3816 bytes left
*Jul 18 22:22:45.437: voice_elog_write:
```

```
*Jul 18 22:22:45.437: elog_write_local: message (len=85) written to elog:
B:1058566965:592:INFO: Playing prompt #1: tftp://172.19.139.245/audio/ch_welcome.au 3673
bytes left
Router#
*Jul 18 22:22:55.942: voice_elog_write:
*Jul 18 22:22:55.942: elog_write_local: message (len=51) written to elog:
B:1058566975:593:ERR : Prompt play setup failure.  3622 bytes left
*Jul 18 22:22:55.942: voice_elog_write:
*Jul 18 22:22:55.942: elog_write_local: message (len=65) written to elog:
B:1058566975:594:INFO: Script received event = "error.badfetch" 3557 bytes left
*Jul 18 22:22:56.918: voice_elog_write:
*Jul 18 22:22:56.918: elog_write_local: message (len=98) written to elog: 1F:1058
Router#
566976:595:INFO: Inform application call disconnected (cause = normal call clearing (16))
3718 bytes left
*Jul 18 22:22:56.918: voice_elog_write:
*Jul 18 22:22:56.918: elog_write_local: message (len=78) written to elog:
B:1058566976:596:INFO: Script received event = "telephone.disconnect.hangup"
 3479 bytes left
*Jul 18 22:22:56.922: voice_elog_write:
*Jul 18 22:22:56.922: elog_write_local: message (len=89) written to elog:
B:1058566976:597:INFO: LegID = 1F: Call disconnected, cause = normal call clearing (16)
3390 bytes left
*Jul 18 22:22:56.922: voice_elog_write:
*Jul 18 22:22:56.922: elog_write_local: message (len=79) written to elog:
1F:1058566976:598:INFO: Call disconnected (cause = normal call clearing (16))
 3639 bytes left
*Jul 18 22:22:56.930: voice_elog_write:
*Jul 18 22:22:56.930: elog_write_local: message (len=39) written to elog:
1F:1058566976:599:INFO: Call released
 3600 bytes left
*Jul 18 22:22:56.930: voice_elog_close, ctx_id=1F voice_elog_close, ctx_id=19
*Jul 18 22:22:56.930: unlock_elog, ctx_id=19, refcnt=0
*Jul 18 22:22:56.930: delete_elog, ctx_id=19
*Jul 18 22:22:56.930: voice_elog_write:
*Jul 18 22:22:56.930: elog_write_local: message (len=59) written to elog:
B:1058566976:600:INFO: Session done, terminating cause = 3331 bytes left
*Jul 18 22:22:56.930: voice_elog_close, ctx_id=B
```

Related Commands

Command	Description
call application event-log	Enables event logging for voice application instances.
debug voip ais	Enables debugging of the AIS database.

debug voip fastpath

To turn on debugging to monitor VoIP fastpath activity, use the **debug voip fastpath**command in privileged EXEC mode. To turn off VoIP fastpath debugging, use the **no** form of this command.

debug voip fastpath[**invalidate**][*slot*/*port*]

no debug voip fastpath[**invalidate**][*slot*/*port*]

Syntax Description

invalidate	(Optional) Turns on debugging for VoIP fastpath cache invalidation.
slot / *port*	(Optional) Slot and port to be debugged. Slash mark is required.

Command Default

VoIP fastpath debugging does not occur.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(21)	This command was introduced on the Cisco AS5400XM and AS5350XM.

Usage Guidelines

The **debug voip fastpath** command displays the details on every packet that is being switched via fastpath. The **debug voip fastpath invalidate** command displays the details of cache invalidation and cache update. The **debug voip fastpath** command and its options are interchangeable with the **debug voice fastpath** command.

VoIP fastpath is enabled by default. In order to disable it, issue the **no voip-fastpath enable** command in global configuration mode.

When VoIP fastpath is enabled, the IP address and User Datagram Protocol (UDP) port number information for the logical channel that is opened for a specific call are cached. VoIP fastpath prevents the RTP stream from reaching the application layer. Instead, the packets are forwarded at a lower layer to help reduce CPU utilization in high call-volume scenarios.

When supplementary services such as hold or transfer are used, VoIP fastpath causes the router to stream the audio to the cached IP address and UDP port. The new logical channel information (generated after a call on hold is resumed or after a transfer is completed) is disregarded. Traffic must go to the application layer constantly so that redefinition of the logical channel is taken into account and audio is streamed to the new IP address and UDP port pair. Therefore, be sure to disable VoIP fastpath in order to support supplementary services.

Note The **debug voip fastpath** command should be enabled only when there is light traffic on the gateway. Enabling this command can affect the functionality of the gateway.

Examples

The following example shows how to turn on VoIP fastpath debugging, shows how to use the **show debug** command to display what debugging functions are enabled, and provides sample output for the debugging function:

```
Router# debug voip fastpath
Fastpath related debugging is on
Router# show debug
 fastpath:
  Fastpath related debugging is on
Router#
*Nov 14 08:22:35.971: NP VPD(2/01): pak sent via fastpath,part=0x652DEE80 ret=0x000003
len=32
*Nov 14 08:22:35.987: NP VPD(2/01): pak sent via fastpath,part=0x652DEEC0 ret=0x000003
len=32
*Nov 14 08:22:36.011: NP VPD(2/01): pak sent via fastpath,part=0x652DEF00 ret=0x000003
len=32
*Nov 14 08:22:36.031: NP VPD(2/01): pak sent via fastpath,part=0x652DEF40 ret=0x000003
len=32
*Nov 14 08:22:36.051: NP VPD(2/01): pak sent via fastpath,part=0x652DEF80 ret=0x000003
len=32
*Nov 14 08:22:36.071: NP VPD(2/01): pak sent via fastpath,part=0x652DEFC0 ret=0x000003
len=32
*Nov 14 08:22:36.095: NP VPD(2/01): pak sent via fastpath,part=0x652DF000 ret=0x000003
len=32
*Nov 14 08:22:36.111: NP VPD(2/01): pak sent via fastpath,part=0x652DF040 ret=0x000003
len=32
*Nov 14 08:22:36.131: NP VPD(2/01): pak sent via fastpath,part=0x652DF080 ret=0x000003
len=32
*Nov 14 08:22:36.151: NP VPD(2/01): pak sent via fastpath,part=0x652DF0C0 ret=0x000003
len=32
*Nov 14 08:22:36.171: NP VPD(2/01): pak sent via fastpath,part=0x652DF100 ret=0x000003
len=32
*Nov 14 08:22:36.195: NP VPD(2/01): pak sent via fastpath,part=0x652DF140 ret=0x000003
len=32
*Nov 14 08:22:36.207: NP VPD(2/01): pak sent via fastpath,part=0x652DF180 ret=0x000003
len=32
*Nov 14 08:22:36.231: NP VPD(2/01): pak sent via fastpath,part=0x652DF1C0 ret=0x000003
len=32
*Nov 14 08:22:36.251: NP VPD(2/01): pak sent via fastpath,part=0x652DF200 ret=0x000003
len=32
*Nov 14 08:22:36.271: NP VPD(2/01): pak sent via fastpath,part=0x652DF240 ret=0x000003
len=32
*Nov 14 08:22:36.291: NP VPD(2/01): pak sent via fastpath,part=0x652DF280 ret=0x000003
len=32
*Nov 14 08:22:36.315: NP VPD(2/01): pak sent via fastpath,part=0x652DF2C0 ret=0x000003
len=32
*Nov 14 08:22:36.331: NP VPD(2/01): pak sent via fastpath,part=0x652DF300 ret=0x000003
len=32
*Nov 14 08:22:36.351: NP VPD(2/01): pak sent via fastpath,part=0x652DF340 ret=0x000003
len=32
*Nov 14 08:22:36.371: NP VPD(2/01): pak sent via fastpath,part=0x652DF380 ret=0x000003
len=32
*Nov 14 08:22:36.391: NP VPD(2/01): pak sent via fastpath,part=0x652DF3C0 ret=0x000003
len=32
```

The following example shows how to use the **debug voip fastpath***slot*/*port* command to debug slot 2, port 13 on the router:

```
Router# debug voip fastpath 2/013
Fastpath related debugging is on
```

```
*Nov 14 08:28:00.623: NP VPD(2/13): pak sent via fastpath,part=0x652DFFC0 ret=0x000003
len=32
*Nov 14 08:28:00.643: NP VPD(2/13): pak sent via fastpath,part=0x652E0000 ret=0x000003
len=32
*Nov 14 08:28:00.659: NP VPD(2/13): pak sent via fastpath,part=0x652E0080 ret=0x000003
len=32
*Nov 14 08:28:00.831: NP VPD(2/13): pak sent via fastpath,part=0x652E0280 ret=0x000003
len=32
*Nov 14 08:28:00.855: NP VPD(2/13): pak sent via fastpath,part=0x652E0300 ret=0x000003
len=32
*Nov 14 08:28:00.867: NP VPD(2/13): pak sent via fastpath,part=0x652E0380 ret=0x000003
len=32
*Nov 14 08:28:01.031: NP VPD(2/13): pak sent via fastpath,part=0x652E0540 ret=0x000003
len=32
*Nov 14 08:28:01.051: NP VPD(2/13): pak sent via fastpath,part=0x652E0580 ret=0x000003
len=32
*Nov 14 08:28:01.075: NP VPD(2/13): pak sent via fastpath,part=0x652E0640 ret=0x000003
len=32
*Nov 14 08:28:01.231: NP VPD(2/13): pak sent via fastpath,part=0x652E0840 ret=0x000003
len=32
*Nov 14 08:28:01.251: NP VPD(2/13): pak sent via fastpath,part=0x652E07C0 ret=0x000003
len=32
*Nov 14 08:28:01.271: NP VPD(2/13): pak sent via fastpath,part=0x652E0900 ret=0x000003
len=32
*Nov 14 08:28:01.439: NP VPD(2/13): pak sent via fastpath,part=0x652E0AC0 ret=0x000003
len=32
*Nov 14 08:28:01.463: NP VPD(2/13): pak sent via fastpath,part=0x652E0B40 ret=0x000003
len=32
*Nov 14 08:28:01.483: NP VPD(2/13): pak sent via fastpath,part=0x652E0BC0 ret=0x000003
len=32
```

The following example shows how to enable debugging for fastpath cache invalidation on slot 2, port 17, and shows how to display sample output for the debugging function:

```
Router# debug voip fastpath invalidate 2/17

 Fastpath cache invalidation related  debugging is on
Router# show voip call summary
PORT            CODEC    VAD VTSP STATE              VPM STATE
=============== ======== === =================== ======================
6/4:0.20         g729r8    y  S_CONNECT             CSM_OC6_CONNECTED
6/4:0.21         g729r8    y  S_CONNECT             CSM_OC6_CONNECTED
Router# show spe | i a
Country code config : default T1 (u Law)
Country code setting: e1-default
Port state: (s)shutdown   (r)recovery  (t)test  (a)active call
            (b)busiedout  (d)download  (B)bad   (p)busyout pending
Call type : (m)modem      (d)digital   (v)voice (f)fax-relay       (_)not in use
Summary   :
Ports     : Total  540  In-use     2  Free   514  Disabled    24
Calls     : Modem    0  Digital    0  Voice    2  Fax-relay    0
                      SPE          SPE     SPE  SPE   Port         Call
SPE#    Port #        State        Busyout Shut Crash State        Type
2/02    0012-0017     ACTIVE             0    0     0 _____a       _____v
2/03    0018-0023     ACTIVE             0    0     0 a_____       v_____
Router# show logging

Syslog logging: enabled (274 messages dropped, 20 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, 1018 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (8192 bytes)
    Count and timestamp logging messages: disabled
    Trap logging: level informational, 133 message lines logged

Log Buffer (1000000 bytes):
*Nov 14 08:40:36.499:  NP VPD (2/17): Cached header parameter values: header size : 28,
payload size : 13, ssrc : 0x24DB1F03, udp chksum : 0x0
*Nov 14 08:40:36.499:  NP VPD (2/17): Cached IP/UDP pkt details: dest-ip: 31.31.31.1
```

```
src-ip: 31.31.31.3   dport: 0x4070  sport: 0x43A6
*Nov 14 08:40:40.851:  NP VPD (2/17): Cached header parameter values: header size : 28,
payload size : 32, ssrc : 0x24DB1F03, udp chksum : 0x0
*Nov 14 08:40:40.851:  NP VPD (2/17): Cached IP/UDP pkt details: dest-ip: 31.31.31.1
src-ip: 31.31.31.3   dport: 0x4070  sport: 0x43A6
*Nov 14 08:40:40.939:  NP VPD (2/17): Cache being cleared due to change in payload size old
 payload size : 32  new rx payload size : 13 cached ssrc : 24DB1F03
```

Related Commands

Command	Description
debug voice fastpath	Turns on debugging to monitor voice fastpath packets.
show voice call	Displays the call status information for voice ports.
voice fastpath enable	Turns on voice fastpath.

debug voip fileacct

To display debugging messages related to voice attributes for file accounting, use the **debug voip fileacct** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip fileacct

no debug voip fileacct

Syntax Description

This command has no arguments or keywords.

Command Default

Debugging of file accounting is not enabled.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(15)XY	This command was introduced.
12.4(20)T	This command was integrated into Cisco IOS Release 12.4(20)T.

Usage Guidelines

This command displays details about the attributes captured in call detail records (CDRs) and their values for the file accounting feature.

Examples

The following example displays output from the **debug voip fileacct**command:

```
Router# debug voip fileacct

*May 10 06:27:43.719: : add attr:47A815E4 clid(21) 4 5000
*May 10 06:27:43.719: new list: 0x4792614C prev list: 0x47A815D0
*May 10 06:27:43.719: : add attr:47926160 dnis(22) 0
*May 10 06:27:43.719: new list: 0x47C3A2C0 prev list: 0x4792614C
*May 10 06:27:43.719: : add attr:47C3A2D4 subscriber(106) 11 RegularLine
*May 10 06:27:43.719: new list: 0x4517FC04 prev list: 0x47C3A2C0
*May 10 06:27:43.719: : add attr:4517FC18 override_session_time(67) 4 0(0) Telephony Leg
*May 10 06:27:43.719: new list: 0x478C0CA4 prev list: 0x4517FC04
*May 10 06:27:43.719: : add attr:478C0CB8 h323-ivr-out(68) 14 Tariff:Unknown
*May 10 06:27:43.719: new list: 0x477EAFFC prev list: 0x478C0CA4
*May 10 06:27:43.719: : add attr:477EB010 h323-voice-quality(70) 1 0
*May 10 06:27:43.719: new list: 0x4783EF80 prev list: 0x477EAFFC
*May 10 06:27:43.719: : add attr:4783EF94 gw-rxd-cgn(94) 28 ton:0,npi:0,pi:0,si:0,#:5000
*May 10 06:27:43.719: list is 466C17A8, list->featurename is 0,feat id is 11205
*May 10 06:27:43.719: fcur is 466C17A8, attr is "TWC","05/10/2007
06:27:43.695","5000","",0,11205,6510EBF8 FDF611DB A527DA52 74E8B890,2BD8,"","","",""
*May 10 06:27:43.719: : del attr47B8E814 callID(1) 4 11224(2BD8)
*May 10 06:27:43.719: : del attr45250054 cdr type(2) 4 0(0)
*May 10 06:27:43.719: : del attr452C52F0 leg type(3) 4 1(1)
*May 10 06:27:43.719: : del attr47914064 h323-conf-id(4) 35 6510EBF8 FDF611DB A527DA52
74E8B890
```

Related Commands

Command	Description
debug voip dump-file-acct	Displays debugging messages related to file accounting flushing processes.
gw-accounting	Enables an accounting method for collecting CDRs.
primary	Sets the primary location for storing the CDRs generated for file accounting.

debug voip fpi call-rate

To enable the call-rate computation, use the **debug voip fpi call-rate**. To disable the debugging output, use the **no** form of this command

debug voip fpi call-rate

no debug voip fpi call-rate

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 3.9S	This command was introduced.

Usage Guidelines

Use the **debug voip fpi call-rate** command in conjunction with the **show voip fpi call-rate**command.

debug voip h221

To debug telephony call control information, use the **debug voip h221**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip h221 [**all**| **default**| **error** [**call [informational]**| **software [informational]**]| **function**| **individual**| **inout**| **raw [decode]**]

no debug voip h221

Syntax Description

all	(Optional) Enables all H.221 debugging, except the raw option.
default	(Optional) Activates function, inout, error call, and software debugging.
error	(Optional) Enables H.221 call error and software error debugging.
error [call]	(Optional) Enables H.221 major call processing error debugs related to the H.221 subsystem.
error [call [informational]]	(Optional) Enables H.221 major and informational call processing error debugs related to the H.221 subsystem.
error [software]	(Optional) Enables H.221 major software error debugs related to the H.221 subsystem.
error [software [informational]]	(Optional) Enables H.221 major and informational software error debugs related to the H.221 subsystem.
function	(Optional) Enables procedure tracing.
individual	(Optional) Activates individual H.221 debugging.
inout	(Optional) Enables subsystem inout debugging.
raw	(Optional) Displays raw BAS messages.
raw [decode]	(Optional) Decodes raw BAS data.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.4(11)T	This command was introduced.

Usage Guidelines

This command enables debugging for H.221 message events (voice telephony call control information).

Note This command provides the same results as the **debug voice h221** command.

Caution We recommend that you log the output from the **debug voip h221 all** 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.

Use the **debug voip h221 individual** *x* command, (where *x* is an index number for a debug category), to activate a single debug, selected by index number instead of entering a group of debug commands. See the table below for a list of debug categories and corresponding index numbers.

Table 4: Indexes and Categories for the debug voip h221 individual command

Index Number	Debug Category
1, 2, 30, 31, 32	Secondary number exchange
5, 6, 14, 15, 16, 22	Audio mode/caps
7, 10, 12, 13, 17, 28	Video mode/caps
8, 9, 23	B-channel mode/caps
11, 24, 33	Miscellaneous command exchange
18	Bandwidth calculations
19, 20, 21	DSP configuration
3, 4, 25, 27, 42, 43	General caps/internal
26	Non-standard caps/command
29	Loop request
34, 35, 36, 37, 38, 39, 40, 41	BAS squelch

Examples

The raw keyword displays the raw BAS information coming from or to the DSP. It is displayed in a hexadecimal octet format. The **decode** option decodes the BAS information into a readable English format.

The following is sample output from the **debug voip h221 raw** decode command:

```
BAS=81:1 0 0 0 0 0 0 1: AUDIO CAPS=g711 a-law
BAS=82:1 0 0 0 0 0 1 0: AUDIO CAPS=g711 u-law
BAS=84:1 0 0 0 0 1 0 0: AUDIO CAPS=g722 48k
BAS=85:1 0 0 0 0 1 0 1: AUDIO CAPS=g728
BAS=F9:1 1 1 1 1 0 0 1: H.242 MBE start indication
BAS=02:0 0 0 0 0 0 1 0: H.242 MBE length=2
BAS=0A:0 0 0 0 1 0 1 0: H.242 MBE type=H.263 caps
BAS=8A:1 - - - - - - -: Always 1
BAS=8A:- 0 0 0 1 - - -: H.263 MPI=1
BAS=8A:- - - - - 0 1 -: H.263 FORMAT=h.263_cif
BAS=8A:- - - - - - - 0: No additional options
```

Related Commands

Command	Description
debug voip ccapi	Enables debugging for the call control application programming interface (CCAPI) contents.
debug voip rtp	Enables debugging for Real-Time Transport Protocol (RTP) named event packets.

debug voip h324

To debug video call control information, use the **debug voip h324**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip h324 [**all**| **function**| **inout**| **default**| **individual** [*number*]| **message**| **error** [**software [informational]**| **call [informational]**]]

Syntax Description

all	(Optional) Enables all H.324 debugging except raw and raw decode.
default	(Optional) Activates function, inout, error call, and software debugging.
error	(Optional) Enables H.324 call error and software error debugging.
error [call]	(Optional) Enables H.324 major call processing error debugs related to the H.324 subsystem.
error [call [informational]]	(Optional) Enables H.324 major and informational call processing error debugs related to the H.324 subsystem.
error [software]	(Optional) Enables H.324 major software error debugs related to the H.324 subsystem.
error [software [informational]]	(Optional) Enables H.324 major and informational software error debugs related to the H.324 subsystem.
function	(Optional) Enables procedure tracing.
individual	(Optional) Activates individual H.324 debugging.
inout	(Optional) Enables subsystem inout debugging.
message	(Optional) Enables H.245 message display to/from H.324. Only displays message types, for message detail, use debug h245 asn1.
number	Index number. Number of debug category. See

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.4(22)T	This command was introduced.

Usage Guidelines

This command enables debugging for H.324 message events (video call control information).

Note This command is the same as the **debug voice h324**command**.**

Caution We recommend that you log the output from the **debug voip h324 all** 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.

Use the **debug voip h324 individual** *index-number* command, where *index number is*a debug category, to activate a single debug.

This is helpful when trying to see a specific problem, without having a large number of debug output being generated. For example, the user could select the command **debug voip h324 individual 4** to see calls where no video caps arrived from the IP side of the call (SIP to H.324 direction). Multiple debug output can be activated using this command, one at a time. These are not additional debug output to the ones enabled by the command **debug voip h324 all**, just another way to selectively see specific information, without generating large amounts of debug output.

Table 5: Index Numbers and Descriptions for the debug voip h324 Command

Index Number	Description
1	Shows incoming H.245 message type
2	Shows MSD master/slave determination upon receiving MSD from peer
3	Warns that no audio caps were found from IP leg (not necessarily an error).
4	Warns that no video caps were found from IP leg (not necessarily an error).
5	Shows MSD master/slave determination when sending MSDack.
6	Displays media type being sent (audio/video), when sending MES message.
7	Displays H.223 parameters when sending TCS.

Index Number	Description
8	Displays OLC information, when sending audio OLC.
9	Displays OLC information, when sending video OLC.
10	Displays OLCack information, when sending OLCack.
11	Displays OLCrej information, when sending OLCrej.
12	Displays digit begin sent, when sending USER INPUT message.
13-15	Displays internal status bits of h245 messages sent/received in the h324 subsystem. No user data is provided.
16	Displays master/slave determination when MSDack is received.
17	Displays media type when MESack is received.
18	Displays media type when MESrej is received.
19	Displays OLC information, when receiving audio OLC.
20	Displays OLC information, when receiving video OLC.
21	Displays media type when OLCack is received.
22	Displays media type when OLCrej is received.
23	Displays message type, when an H.245 miscellaneous message is received (for example FastVideoUpdate).
24	Displays digit being received, when receiving USER INPUT message.
25	Displays message type, when an H.245 miscellaneous message is sent (for example FastVideoUpdate).
26	Displays outgoing message command type. No user data is provided with this debug.
27	Displays the initial H.223 mux level received from the peer, reported by the DSP.

Index Number	Description
28	Displays information about either OLCack or OLCrej being sent in response to an OLC request.
29	Displays the audio codec being opened with the IP leg.
30	Displays the video codec being opened with the IP leg. Should always be the same as the video codec with the H.324 leg.
31	Displays when Cisco IOS is sending the DSP either the H.223 multiplex table, or AL information. No user data is provided.
32	Indicates the digit being sent to the IP leg, through the RFC 2833 procedure.
33-34	Displays the parameters being sent to the DSP to configure either audio or video.
35	Displays information about the H.223 multiplex table being sent to the DSP.
36	Displays information about the H.223 AL configuration being sent to the DSP.
37-38	Indicates message arriving from IP leg. No user data is provided.
39	Displays information when receiving VENDOR ID message. This may show the type of equipment being connected to on the H.324 leg, if the peer adds the information to the message.
40	Displays the new H.223 multiplex level being configured.
41	Displays the new H.223 maximum PDU size being configured.
42	Indicates when the internal video capability memory has been released. No user data is provided.
43	Indicates when an empty capability set (ECS) has arrived from the IP leg of the call.
44	Indicates when a new capability set has arrived from the IP leg after an ECS has arrived.

Index Number	Description
45	Displays the dynamic payload number from the IP leg (H.324 to IP direction).

debug voip hpi

To enable debugging for Host Port Interface (HPI) message events, use the **debug voip hpi**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip hpi [**all**| **command**| **default**| **detail**| **error** [**call [informational]**| **software [informational]**]| **function**| **inout**| **notification**| **response**| **stats**| **checker**]

no debug voip hpi

Syntax Description

all	(Optional) Displays all HPI debugging messages.
command	(Optional) Displays commands that are being sent to the 54x DSP.
default	(Optional) Displays HPI detail, error, and inout debugging messages and also runs if no keywords are added.
detail	(Optional) Displays detailed information about commands for the HPI. This command is used in conjunction with other **debug voip hpi** commands to show additional details when you use the **command**, **notification**, and **response** keywords.
error	(Optional) Displays HPI error messages.
call	(Optional) Displays call processing errors.
informational	(Optional) Displays minor and major errors. Without the **informational** keyword, only major errors are displayed.
software	(Optional) Displays software processing errors.
function	(Optional) Displays HPI functions.
inout	(Optional) Displays the output for the **command**, **notification**, **response**, and **stats** keywords.
notification	(Optional) Displays notification messages that are sent from the 54x DSP (for example, tone detection notification).
response	(Optional) Displays responses to commands that are sent by the 54x DSP (for example, responses to statistic requests).

stats	(Optional) Displays HPI statistics.
checker	(Optional) Displays HPI checker operations.

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command replaces the **debug hpi** command.
12.3(14)T	The **checker** keyword was added.

Usage Guidelines

This command enables debugging for HPI message events, which are used to communicate with digital signal processors (DSPs).

Use the **debug voip hpi all** command to view gateway DSP modem relay termination codes. The DSP-to-host messages for the modem relay termination indicate to the host the modem relay session termination time, physical or link layer, and other probable causes for disconnection. On receiving this indication from the DSP, the host can disconnect the call or place the channel in the modem passthrough state.

Examples

The following is sample output from the **debug voip hpi all** command for an incoming ISDN call:

```
Router# debug voip hpi all

01:28:44: //-1/xxxxxxxxxxxx/HPI/[]/hpi_dspmgr_open:
```

The following event shows that the HPI has identified the call, as shown by the GUID, but the call leg has not been specified, as shown by the -1 value in the CallEntry ID:

```
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_dspmgr_open:
    Allocated DSP resource: dsp_intf=0x64AF0EEC hpi_cdb=0x64ACED34 ret=1
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_dspmgr_open:
    Exit Line # 9411
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_init:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_init:
    Open channel
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_init:
    Packet details:
      Packet Length (16), Channel Id (1), Packet Id (74)
      ALawULawSelect=A Law Associated SignalingChannel (128)
      Timeslot=0 SerialPort=0
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_voice_config_params:
```

For each packet-related event, information about the packet is shown following the event. The following two events show the Real-Time Protocol (RTP) packet:

```
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_encap_config:
    RTP information
```

```
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_encap_config:
    Packet details:
      Packet Length (38), Channel Id (1), Packet Id (92) TransportProtocol=2
      t_ssrc=0x00 r_ssrc=0x00 t_vpxcc=0x0 r_vpxcc=0x0
      sid_support=0 tse_payload=101 seq_num=0x0 redundancy=0
      cc_payload_type=0 fax_payload_type=0 alaw_pcm_switchover=0
      mulaw_pcm_switchover=0 dtmf_payload_type=0 nte_rcv_payload_type=101
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_set_playout_config:
    Packet details:
      Packet Length (18), Channel Id (1), Packet Id (76)
      Mode=1, Initial=60, Min=40, Max=200, fax_nom=300
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_inband_sig:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_fax_enable:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_fax_enable:
    Enable FAX
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_fax_enable:
    Packet details:
      Packet Length (8), Channel Id (1), Packet Id (67)
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_echo_cancel:
    Packet details:
      Packet Length (14), Channel Id (1), Packet Id (66)
      flags=0x00000B00, Threshold=-21, SuppressorCoverage=7
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_idle_code_det:
    Packet details:
      Packet Length (14), Channel Id (1), Packet Id (116)
      Enable (FALSE), Code=0x00000000, Duration (6000 ms)
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_set_gain:
    Packet details:
      Packet Length (12), Channel Id (1), Packet Id (91)
      Gain: In=0, Out=0
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/caplog_hpi_msg_log:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_cng_config:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_info_field_size_config:
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_digit_relay_config:
    Exit Line # 4162
01:28:44: //-1/3FE022AC8009/HPI/[2/0:23]/hpi_dspmgr_update_callid:
```

At this point, the HPI identifies the call leg, as shown by the CallEntry ID changing from -1 to 11.

```
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_voice_get_capabilities:
    Exit Line # 5073
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_stop_service:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_idle_service:
    Packet details:
      Packet Length (8), Channel Id (1), Packet Id (68)
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_open_service:
    Setting codec g729r8
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_set_codec:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_dsprm_callback:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_voice_config_params:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_digit_relay_config:
    Exit Line # 4162
```

The RTP packet is shown again, but now more information is available, such as payload types. The packet ID identifies this as the same RTP packet shown earlier.

```
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_encap_config:
    RTP information
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_encap_config:
    Packet details:
      Packet Length (38), Channel Id (1), Packet Id (92) TransportProtocol=2
      t_ssrc=0x040 r_ssrc=0x00 t_vpxcc=0x0 r_vpxcc=0x0
      sid_support=1 tse_payload=101 seq_num=0x13D3 redundancy=0
      cc_payload_type=125 fax_payload_type=122 alaw_pcm_switchover=8
      mulaw_pcm_switchover=0 dtmf_payload_type=121 nte_rcv_payload_type=101
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
```

```
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_pak_suppress:
    Stop packet suppression
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_pak_suppress:
    Packet details:
      Packet Length (10), Channel Id (1), Packet Id (106)
      Mode=1
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_vad_enable:
    Packet details:
      Packet Length (18), Channel Id (1), Packet Id (78)
      VAD=1 (ON): Threshold=-38, VADTime=250 Aggressive=0, Noise=-62
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_init:
    Open channel
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_init:
    Packet details:
      Packet Length (16), Channel Id (1), Packet Id (74)
      ALawULawSelect=A Law Associated SignalingChannel (128)
      Timeslot=0 SerialPort=0
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_voice_config_params:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_encap_config:
    RTP information
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_encap_config:
    Packet details:
      Packet Length (38), Channel Id (1), Packet Id (92) TransportProtocol=2
      t_ssrc=0x040 r_ssrc=0x00 t_vpxcc=0x0 r_vpxcc=0x0
      sid_support=1 tse_payload=101 seq_num=0x13D3 redundancy=0
      cc_payload_type=125 fax_payload_type=122 alaw_pcm_switchover=8
      mulaw_pcm_switchover=0 dtmf_payload_type=121 nte_rcv_payload_type=101
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_set_playout_config:
    Packet details:
      Packet Length (18), Channel Id (1), Packet Id (76)
      Mode=1, Initial=60, Min=40, Max=200, fax_nom=300
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_inband_sig:
```

In the following several events, fax is enabled. Packets for echo cancellation, gain, voice activity detection (VAD), and other parameters appear.

```
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_fax_enable:
    Enable FAX
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_fax_enable:
    Packet details:
      Packet Length (8), Channel Id (1), Packet Id (67)
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_echo_cancel:
    Packet details:
      Packet Length (14), Channel Id (1), Packet Id (66)
      flags=0x00000B00, Threshold=-21, SuppressorCoverage=7
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_idle_code_det:
    Packet details:
      Packet Length (14), Channel Id (1), Packet Id (116)
      Enable (FALSE), Code=0x00000000, Duration (6000 ms)
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_set_gain:
    Packet details:
      Packet Length (12), Channel Id (1), Packet Id (91)
      Gain: In=0, Out=0
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_cng_config:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_info_field_size_config:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_digit_relay_config:
    Exit Line # 4162
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_start_service:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_voice_mode:
    Packet details:
      Packet Length (28), Channel Id (1), Packet Id (73)
      CodingType=20, Voice FieldSize (20), VAD Flag (250),
      EchoLength=512, ComfortNoise=1, inband_detect=0x00000001,
```

```
      DigitRelay=2, AGC Flag=0, ECAN TestGroup=0,
      ECAN TestNumber=0, DynamicPayload=0
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_idle_code_det:
    Packet details:
      Packet Length (14), Channel Id (1), Packet Id (116)
      Enable (FALSE), Code=0x00000000, Duration (6000 ms)
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_start_service:
    Exit Line # 2816
01:28:44: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_send_data_to_dsp:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_info:
```

Transmit and receive events are shown, along with packet information.

```
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_tx:
    Packet details:
      Packet Length (10), Channel Id (1), Packet Id (86)
      ResetFlag (0x00000000)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_rx:
    Packet details:
      Packet Length (10), Channel Id (1), Packet Id (87)
      ResetFlag (0x00000000)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_pd:
    Playout delay
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_pd:
    Packet details:
      Packet Length (10), Channel Id (1), Packet Id (83)
      ResetFlag (0x00000000)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_pe:
    Playout error
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_pe:
    Packet details:
      Packet Length (10), Channel Id (1), Packet Id (84)
      ResetFlag (0x00000000)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_query_info:
    Exit Line # 6578
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_message:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
```

Statistics for each of the events are displayed.

```
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_query_tx:
    Got TX stats
    Packet details:
      Packet Length (36), Channel Id (1), Packet Id (199)
      TX Packets (87), Signaling Packets (0) ComfortNoise Packets (0)
      Transmit Duration (1750)
      Voice Transmit Duration (1750), FAX Transmit Duration (0)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_message:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_query_rx:
    Got RX stats
    Packet details:
      Packet Length (120), Channel Id (1), Packet Id (200)
      RX Packets (5): Signaling (0), ComfortNoise (1)
      Receive Duration (1750): Voice (70) FAX (0)
      Packet Counts: OOSequence (0), Bad header (0), Late (1), Early (0)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_message:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_query_pd:
    Got Playout Delay stats...
    Packet details:
      Packet Length (24), Channel Id (1), Packet Id (196)
      RX Delay: CurrentEstimate=69 Low WaterMark (69) High WaterMark (70)
      Clock Offset (-279863943)
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_message:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/caplog_hpi_msg_log:
01:28:46: //11/3FE022AC8009/HPI/[2/0:23:11]/hpi_receive_query_pe:
```

```
    Got Playout Error stats
    Packet details:
      Packet Length (32), Channel Id (1), Packet Id (197)
      Predictive Concealment Duration (0)
      Interpolative Concealment Duration (0)
      Silence Concealment Duration (0)
      Retroactive Memory Update (0)
      Buffer overflow discard duration (10)
      Talkspurt Detection Errors (0)
```

The following sample output from the **debug voip hpi checker** command helps verify the operations of the HPI checker:

```
Router# debug voip hpi checker
*May 19 06:30:53.532: hpi [] DSP [0/0x0] S_HPI_CLOSED(0) E_HPI_DSPRM_OPEN/SET_CODEC(54)
*May 19 06:30:53.532: hpi [] DSP [0/0x0] -> S_HPI_CLOSED(0)
*May 19 06:30:53.620: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_CLOSED(0)
E_HPI_DSP_OPEN_VOICE_CHANNEL(11)
*May 19 06:30:53.620: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_IDLE(1)
*May 19 06:30:53.620: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1)
E_HPI_DSPRM_OPEN/SET_CODEC(54)
*May 19 06:30:53.620: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_IDLE(1)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1) E_HPI_DSP_ENC_CONFIG(29)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_IDLE(1)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1)
E_HPI_DSP_SET_VOICE_PLAYOUT_DELAY(13)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_IDLE(1)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1) E_HPI_DSP_IDLE_CODE_CONTROL
 (50)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_IDLE(1)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1) E_HPI_DSP_VOICE_MODE(10)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GEN_PACKET_CONTROL(41)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_ECHO_CANCELLER_CONTROL(3)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_SET_GAINS(28)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_DIGIT_RELAY(22)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_VAD_ENABLE(15)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GEN_PACKET_CONTROL(41)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_INBAND_DETECTOR_CONTROL(45)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.636: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_INBAND_DETECTOR_CONTROL(45)
*May 19 06:30:53.636: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1) E_HPI_DSP_VOICE_MODE(10)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.624: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GEN_PACKET_CONTROL(41)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_ECHO_CANCELLER_CONTROL(3)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_SET_GAINS(28)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_DIGIT_RELAY(22)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_VAD_ENABLE(15)
*May 19 06:30:53.628: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GEN_PACKET_CONTROL(41)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_INBAND_DETECTOR_CONTROL(45)
```

```
*May 19 06:30:53.632: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:53.636: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_INBAND_DETECTOR_CONTROL(45)
*May 19 06:30:53.636: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_GET_TX_STAT(23)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_GET_RX_STAT(24)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GET_VOICE_PLAYOUT_DELAY(20)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GET_VOICE_PLAYOUT_ERROR(21)
*May 19 06:30:56.512: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_GET_TX_STAT(23)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_GET_RX_STAT(24)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GET_VOICE_PLAYOUT_DELAY(20)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GET_VOICE_PLAYOUT_ERROR(21)
*May 19 06:30:59.384: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:31:06.524: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2)
E_HPI_DSP_GEN_PACKET_CONTROL(41)
*May 19 06:31:06.524: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:31:06.532: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_GET_LEVELS(26)
*May 19 06:31:06.536: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:31:06.536: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_GET_ERROR_STAT(0)
*May 19 06:31:06.536: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_VOICE(2)
*May 19 06:31:06.572: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_VOICE(2) E_HPI_DSP_IDLE_MODE(5)
*May 19 06:31:06.572: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_IDLE(1)
*May 19 06:31:06.572: hpi [2/0:23 (22)] DSP [0/0x0] S_HPI_IDLE(1)
E_HPI_DSP_CLOSE_VOICE_CHANNEL(12)
*May 19 06:31:06.572: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_CLOSED(0)
*May 19 06:31:06.576: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_CLOSED(0)
*May 19 06:31:06.576: hpi [2/0:23 (22)] DSP [0/0x0] -> S_HPI_CLOSED(0)
```

Related Commands

Command	Description
show voice hpi capture	Verifies capture status and statistics.

debug voip ipipgw

To turn on debugging for the Cisco Multiservice IP-to-IP Gateway (IPIPGW), use the **debug voip ipipgw** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip ipipgw

no debug voip ipipgw

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(13)T3	This command was introduced.

Examples

The following example shows debugging output from a Cisco IPIPGW:

```
Aug  8 15:24:30.626 EDT: cch323_build_early_fastStart_cap_response:
ccb-remote_fastStart=0x63C20630
Aug  8 15:24:30.626 EDT:
cch323_build_early_fastStart_cap_response:symm_mask=1,tempOtherCodec=5,templocalCodec=5,audioFastStartArray=0x63C1299C
The following line shows fast start response beginning for the inbound leg of the IP-to-IP
 call:
Aug  8 15:24:30.626 EDT: cch323_build_early_fastStart_cap_response: Assuming ipipgw on
inbound leg.
Aug  8 15:24:30.626 EDT: Function: cch323_build_olc_for_ccapi, Line: 1198,
audioFastStartArray=0x63C1299C
Aug  8 15:24:30.626 EDT: cch323_build_olc_for_ccapi: channel_info ptr=0x63C203F0, ccb
ptr=0x63C18580
The following lines indicate the inbound call leg will send the channel information to the
 outbound call leg (not yet created):
Aug  8 15:24:30.626 EDT: cch323_build_olc_for_ccapi: Channel Information:
        Logical Channel Number (fwd/rev): 1
        Channel address (fwd/rev):        0x10C0C27
        RTP  Channel (fwd/rev):           19362
        RTCP Channel (fwd/rev):           19363
        QoS Capability (fwd/rev):         0
        Symmetric Audio Cap Mask:         0x1
        Symmetric Audio Codec Bytes:      160
        Flow Mode:                        0
        Silence Suppression:              0
Aug  8 15:24:30.626 EDT: cch323_build_olc_for_ccapi:NumOfElements = 1 idx = 1
```

The following line indicates the inbound call leg is set to work in IP-to-IP mode (0xF0):

```
Aug  8 15:24:30.630 EDT: cch323_set_h245_state_mc_mode_incoming: h245 state m/c mode=0xF0
```

The following line indicates flow mode for incoming call leg is set to FLOW_THROUGH (incoming callid = 35). At this point Session Application opens the outbound leg. Some output is omitted here.

```
Aug  8 15:24:30.630 EDT: cch323_media_flow_mode: IPIPGW(35):Flow Mode=1
Aug  8 15:24:30.630 EDT: cch323_set_h245_state_mc_mode_outgoing:call_spi_mode = 1
```

The following line indicates the outbound call leg is set to work in IP-to-IP mode (0xF0):

```
Aug  8 15:24:30.630 EDT: cch323_set_h245_state_mc_mode_outgoing: h245 state m/c mode=0xF0
Aug  8 15:24:30.630 EDT: cch323_get_peer_info line 1022:
Aug  8 15:24:30.630 EDT: cch323_get_peer_info line 1026:
Aug  8 15:24:30.630 EDT: cch323_set_pref_codec_list:IPIPGW(36):peer channel present: dp
pref mask=FFFFFFFF
Aug  8 15:24:30.630 EDT: cch323_set_pref_codec_list:IPIPGW(36):first preferred
codec(bytes)=5(160)
```

The following line indicates the outbound call leg is set to FLOW_THROUGH (outbound callid = 36):

```
Aug  8 15:24:30.630 EDT: cch323_get_peer_info: Flow Mode set to FLOW_THROUGH for callId 36
Aug  8 15:24:30.642 EDT: cch323_build_local_encoded_fastStartOLCs: state_mc_mode=0xF0 on
outbound leg.
Aug  8 15:24:30.642 EDT: cch323_build_local_encoded_fastStartOLCs:srcAddress = 0x10C0C30,
h245_lport = 0, flow mode = 1, minimum_qos=0
Aug  8 15:24:30.642 EDT: cch323_generic_open_logical_channel: IPIPGW: current codec =
5:160:160.
```

The following line indicates the IPIPGW received fast start response from the remote (called party) entity of the outbound call leg:

```
Aug  8 15:24:30.658 EDT: Function: cch323_receive_fastStart_cap_response Line: 2800
Aug  8 15:24:30.658 EDT: Function: cch323_build_olc_for_ccapi, Line: 1198,
audioFastStartArray=0x63C1259C
Aug  8 15:24:30.658 EDT: cch323_build_olc_for_ccapi: channel_info ptr=0x63C12738, ccb
ptr=0x631A4D68
Aug  8 15:24:30.658 EDT: cch323_build_olc_for_ccapi: Channel Information:
        Logical Channel Number (fwd/rev): 1
        Channel address (fwd/rev):        0x10C0C28
        RTP  Channel (fwd/rev):           19128
        RTCP Channel (fwd/rev):           19129
        QoS Capability (fwd/rev):         0
        Symmetric Audio Cap Mask:         0x1
        Symmetric Audio Codec Bytes:      160
        Flow Mode:                        0
        Silence Suppression:              0
Aug  8 15:24:30.658 EDT: cch323_build_olc_for_ccapi:NumOfElements = 1 idx = 1
Aug  8 15:24:30.658 EDT: Function: cch323_do_open_channel_ind Line: 1080
Aug  8 15:24:30.658 EDT: Function: cch323_open_channel_ind Line: 1132
```

The following lines indicates the outbound call leg (36) sends the channel response back to the inbound call leg (35) via CCAPI:

```
Aug  8 15:24:30.658 EDT: cch323_receive_fastStart_cap_response: callID 0x24(36),
audioFastStartArray = 0x0.
Aug  8 15:24:30.658 EDT: cch323_peer_channel_ind: IPIPGW:### chn info coming in chn_ind()
Aug  8 15:24:30.658 EDT: cch323_peer_channel_ind: IPIPGW(35):giving event to Fast start
logic.
Aug  8 15:24:30.658 EDT: Function: cch323_do_open_channel Line: 5557
Aug  8 15:24:30.658 EDT: cch323_do_open_channel: line:5566, ccb->status=0x4000000
Aug  8 15:24:30.658 EDT: cch323_do_open_channel:srcAddress = 0x10C0C30, h245_lport = 18308,
 minimum_qos=0
Aug  8 15:24:30.658 EDT: cch323_build_fastStart_cap_response: Start...
Aug  8 15:24:30.658 EDT: cch323_build_fastStart_cap_response: selectCodec=5, codec_mask=1,
 configured_codecBytes=160
        forward_codecBytes=160, reverse_codecBytes=160, audioFastStartArray=0x63C1299C
Aug  8 15:24:30.658 EDT: cch323_prepare_fastStart_cap_response line 2138
Aug  8 15:24:30.658 EDT: cch323_prepare_fastStart_cap_response: callID 0x23(35),
audioFastStartArray = 0x0.
```

```
Aug  8 15:24:30.658 EDT: cch323_prepare_fastStart_cap_response,
ccb->local_fastStart=0x63C183C0, srcAddr=0x10C0C30, lport=18308, rport=19362, rc=1
Aug  8 15:24:30.658 EDT: cch323_build_fastStart_cap_response: local_fastStart=0x63C183C0,
negotiated_codec=5, negotiated_codec_bytes=160
Aug  8 15:24:30.658 EDT: cch323_build_fastStart_cap_response: Received peer cap info. Notify
 RAS state machine (possible BRQ).
```

Outbound leg, at this point, has prepared the fast start response to be sent to the originating (calling party). This is sent in the next outgoing call control message (such as ALERT or PROGRESS):

```
Aug  8 15:24:30.658 EDT: cch323_build_fastStart_cap_response: Done.
Aug  8 15:24:30.658 EDT: cch323_do_open_channel: line:5644, ccb->status=0x4004200
Aug  8 15:24:30.674 EDT: cch323_h245_connection_sm: state = 0 event=5 ccb=63C18580
Aug  8 15:24:30.674 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.678 EDT: cch323_h245_cap_ind: IPIPGW(35): masks au=0x1 data=0xC uinp=0x32.
```

The following line indicates the inbound call leg (35) received capability set (CAPSET) message**:**

```
Aug  8 15:24:30.678 EDT: cch323_run_h245_cap_in_sm:IPIPGW(35): got incoming CAPSET msg.
Aug  8 15:24:30.678 EDT: cch323_do_transparent_cap_ind: IPIPGW(35):mask sent to other leg=1
```

The following lines show the inbound call leg (35) forwarding the TCS to the outbound leg and waiting for the response of the outbound call leg (TCSACK or TCSREJ):

```
Aug  8 15:24:30.678 EDT: cch323_run_h245_cap_in_sm:IPIPGW(35):suppressTCS: our TCS will be
 sent based on peer.
Aug  8 15:24:30.678 EDT: cch323_h245_cap_notify:IPIPGW(35):not xmiting CAPSACK: wait for
peer to ack.
Aug  8 15:24:30.678 EDT: cch323_caps_ind: IPIPGW(36):setting the mask to new : current
mask=0x4FFFF new mask=0x1.
Aug  8 15:24:30.678 EDT: cch323_caps_ind: IPIPGW(36): ExtendedCapsPresent
Aug  8 15:24:30.678 EDT: cch323_set_dtmf_relay_mask: IPIPGW(36): extract dtmf-caps from
caps struct
Aug  8 15:24:30.678 EDT: cch323_set_dtmf_relay_mask: IPIPGW(36): After extracting dtmf-caps
 from caps structccb->user_caps.user_input_bit_mask[0x1C]
```

The following line shows the outbound leg sending the TCS to the called party. No codec filter is configured on outbound dial-peer (FFFFFFFF):

```
Aug  8 15:24:30.678 EDT: cch323_prepare_preferred_codec_list: IPIPGW(36):munging
caps:5:1:FFFFFFFF
Aug  8 15:24:30.678 EDT: cch323_prepare_preferred_codec_list: IPIPGW(36):final mask=1
Aug  8 15:24:30.678 EDT: cch323_peer_caps_ind_common:IPIPGW(36):starting 245 via tunnel
Aug  8 15:24:30.678 EDT: cch323_h245_connection_sm: state = 0 event=1 ccb=631A4D68
Aug  8 15:24:30.678 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.678 EDT: cch323_h245_start_cap_sm: IPIPGW(36): starting the cap/msd machine.
Aug  8 15:24:30.678 EDT: cch323_send_generic_caps: IPIPGW: audiomask raw =0x1.
Aug  8 15:24:30.678 EDT: cch323_set_pref_codec_list:IPIPGW(36):first preferred
codec(bytes)=21(0)
```

The following line shows the outbound leg forwarding the TCS over H.225 tunnel (starting H.245 via tunnel):

```
Aug  8 15:24:30.678 EDT: cch323_send_generic_caps: IPIPGW:[trans]audio mask after
operation=0x1.
```

The following lines show master-slave determination events passing from inbound to outbound and vice versa:

```
Aug  8 15:24:30.678 EDT: cch323_run_passthru_msd: IPIPGW(36):event = H245_EVENT_MSD
Aug  8 15:24:30.678 EDT: cch323_h245_connection_sm: state = 0 event=5 ccb=63C18580
Aug  8 15:24:30.678 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.678 EDT: cch323_run_passthru_msd: IPIPGW(35):event = H245_EVENT_MS_IND
Aug  8 15:24:30.678 EDT: cch323_h245_connection_sm: state = 2 event=5 ccb=631A4D68
Aug  8 15:24:30.678 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.678 EDT: cch323_h245_cap_ind: IPIPGW(36): masks au=0x1 data=0xC uinp=0x32.
Aug  8 15:24:30.678 EDT: cch323_run_h245_cap_in_sm:IPIPGW(36): got incoming CAPSET msg.
Aug  8 15:24:30.678 EDT: cch323_do_transparent_cap_ind: IPIPGW(36):mask sent to other leg=1
```

The following lines show the outbound leg forwarding the TCS to the other leg and waiting for its response (TCSACK or TCSREJ):

```
Aug  8 15:24:30.678 EDT: cch323_run_h245_cap_in_sm:IPIPGW(36):suppressTCS: our TCS will be
 sent based on peer.
Aug  8 15:24:30.678 EDT: cch323_h245_cap_notify:IPIPGW(36):not xmiting CAPSACK: wait for
peer to ack.
Aug  8 15:24:30.678 EDT: cch323_run_passthru_msd: IPIPGW(36):event = H245_EVENT_MSD
Aug  8 15:24:30.678 EDT: cch323_caps_ind: IPIPGW(35):setting the mask to new : current
mask=0x4FFFF new mask=0x1.
Aug  8 15:24:30.682 EDT: cch323_caps_ind: IPIPGW(35): ExtendedCapsPresent
Aug  8 15:24:30.682 EDT: cch323_set_dtmf_relay_mask: IPIPGW(35): extract dtmf-caps from
caps struct
Aug  8 15:24:30.682 EDT: cch323_set_dtmf_relay_mask: IPIPGW(35): After extracting dtmf-caps
 from caps structccb->user_caps.user_input_bit_mask[0x1C]
Aug  8 15:24:30.682 EDT: cch323_prepare_preferred_codec_list: IPIPGW(35):munging
caps:21:1:FFFFFFFF
Aug  8 15:24:30.682 EDT: cch323_prepare_preferred_codec_list: IPIPGW(35):final mask=1
Aug  8 15:24:30.682 EDT: cch323_peer_caps_ind_common:IPIPGW(35):starting 245 via tunnel
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: state = 0 event=1 ccb=63C18580
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.682 EDT: cch323_h245_start_cap_sm: IPIPGW(35): starting the cap/msd machine.
Aug  8 15:24:30.682 EDT: cch323_send_generic_caps: IPIPGW: audiomask raw =0x1.
Aug  8 15:24:30.682 EDT: cch323_set_pref_codec_list:IPIPGW(35):first preferred
codec(bytes)=21(0)
```

The following line shows the inbound call leg sending the TCS to the calling party:

```
Aug  8 15:24:30.682 EDT: cch323_send_generic_caps: IPIPGW:[trans]audio mask after
operation=0x1.
Aug  8 15:24:30.682 EDT: cch323_run_passthru_msd: IPIPGW(35):event = H245_EVENT_MSD
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: state = 2 event=5 ccb=631A4D68
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.682 EDT: cch323_run_passthru_msd: IPIPGW(36):event = H245_EVENT_MS_IND
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: state = 2 event=5 ccb=631A4D68
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.682 EDT: cch323_run_h245_cap_out_sm: IPIPGW(36): got caps ack.
Aug  8 15:24:30.682 EDT: cch323_run_h245_cap_out_sm:IPIPGW(36): sending caps ack to other
leg.
Aug  8 15:24:30.682 EDT: Function: cch323_do_caps_ack Line: 1116
Aug  8 15:24:30.682 EDT: cch323_run_passthru_msd: IPIPGW(35):event = H245_EVENT_MSD
Aug  8 15:24:30.682 EDT: cch323_peer_caps_ack: IPIPGW(35):sending caps resp event to CAP
state mc.
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: state = 2 event=5 ccb=631A4D68
Aug  8 15:24:30.682 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.682 EDT: cch323_run_passthru_msd: IPIPGW(36):event = H245_EVENT_MS_CFM
Aug  8 15:24:30.682 EDT: cch323_run_passthru_msd: IPIPGW(35):event = H245_EVENT_MS_DET_RSP
Aug  8 15:24:30.686 EDT: cch323_h245_connection_sm: state = 2 event=5 ccb=63C18580
Aug  8 15:24:30.686 EDT: cch323_h245_connection_sm: listen state=0
Aug  8 15:24:30.686 EDT: cch323_run_h245_cap_out_sm: IPIPGW(35): got caps ack.
```

The following line shows the inbound leg informing the outbound leg of the TCSACK**:**

```
Aug  8 15:24:30.686 EDT: cch323_run_h245_cap_out_sm:IPIPGW(35): sending caps ack to other
leg.
Aug  8 15:24:30.686 EDT: Function: cch323_do_caps_ack Line: 1116
Aug  8 15:24:30.686 EDT: cch323_peer_caps_ack: IPIPGW(36):sending caps resp event to CAP
state mc.
Aug  8 15:24:30.686 EDT: cch323_h245_connection_sm: state = 2 event=5 ccb=63C18580
Aug  8 15:24:30.686 EDT: cch323_h245_connection_sm: listen state=0
```

The following lines show that master-slave determination procedures are completed on both call legs:

```
Aug  8 15:24:30.686 EDT: cch323_run_passthru_msd: IPIPGW(35):event = H245_EVENT_MS_CFM
Aug  8 15:24:30.686 EDT: cch323_run_passthru_msd: IPIPGW(36):event = H245_EVENT_MS_DET_RSP
```

debug voip ivr

Note The **debug voip ivr**command is replaced by the **debug voip application**command. See the **debug voip application**command for more information.

debug voip ivr all

Note The **debug voip ivr all**command is replaced by the **debug voip application all**command. See the **debug voip application all**command for more information.

debug voip ivr applib

Note The **debug voip ivr applib** command is replaced by the **debug voip application core** command. See the **debug voip application core** command for more information.

debug voip ivr callsetup

Note The **debug voip ivr callset**command is replaced by the **debug voip application callset**command. See the **debug voip application callset**command for more information.

debug voip ivr digitcollect

Note The **debug voip ivr digitcollect**command is replaced by the **debug voip application digitcollect**command. See the **debug voip application digitcollect** command for more information.

debug voip ivr dynamic

Note The **debug voip ivr dynamic** command is replaced by the **debug voip application media state** command. See the **debug voip application media state** command for more information.

debug voip ivr error

Note The **debug voip ivr error**command is replaced by the **debug voip application error**command. See the **debug voip application error**command for more information.

debug voip ivr redirect

Note The **debug voip ivr redirect**command is replaced by the **debug voip application redirect**command. See the **debug voip application redirect**command for more information.

debug voip ivr script

Note The **debug voip ivr script** command is replaced by the **debug voip application script** command. See the **debug voip application script** command for more information.

debug voip ivr settlement

Note The **debug voip ivr settlement**command is replaced by the **debug voip application settlement** command. See the **debug voip application settlement**command for more information.

debug voip ivr states

Note The **debug voip ivr states**command is replaced by the **debug voip application media state**command. See the **debug voip application media state** command for more information.

debug voip ivr supplementary-service

Note The **debug voip ivr supplementary-service**command is replaced by the **debug voip application supplementary-service**command. See the **debug voip application supplementary-service** command for more information.

debug voip ivr tclcommands

Note The **debug voip ivr tclcommands**command is replaced by the **debug voip application tclcommands**command. See the **debug voip application tclcommands**command for more information.

debug voip lpcor

To display debugging information for the logical partitioning class of restriction (LPCOR) feature, use the **debug voip lpcor**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip lpcor [**all**| **default**| **detail**| **error** [**call**| **software**]| **function**| **inout**]

no debug voip lpcor [**all**| **default**| **detail**| **error** [**call**| **software**]| **function**| **inout**]

Syntax Description

all	(Optional) Enables all LPCOR debugging.
default	(Optional) Enables error, function, and inout debugging. This is the default option if no keywords are used.
detail	(Optional) Enables detailed trace messages of the LPCOR subsystem.
error	(Optional) Enables LPCOR major call and software error debugging.
call	(Optional) Enables major call error debugging.
software	(Optional) Enables major software error debugging.
function	(Optional) Enables tracing of the functions called by the LPCOR subsystem.
inout	(Optional) Enables function in and out debugging.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
15.0(1)XA	This command was introduced.
15.1(1)T	This command was integrated into Cisco IOS Release 15.1(1)T.

Usage Guidelines

This command enables debugging for LPCOR events.

Examples

The following is sample output from the **debug voip lpcor**command for a call between two phones that was blocked by LPCOR policy validation:

```
*Jun 25 09:52:18.910: //-1/xxxxxxxxxxxx/LPCOR/lpcor_get_index_by_name:
   lpcor local_sccp_phone_1
*Jun 25 09:52:18.910: //-1/xxxxxxxxxxxx/LPCOR/lpcor_get_index_by_name:
   lpcor local_sccp_phone_1 index 1
*Jun 25 09:52:20.114: //-1/xxxxxxxxxxxx/LPCOR/lpcor_get_index_by_peer:
   peer tag 20003, direction 1
*Jun 25 09:52:20.114: //-1/xxxxxxxxxxxx/LPCOR/lpcor_get_index_by_name:
   lpcor local_sccp_phone_2
*Jun 25 09:52:20.114: //-1/xxxxxxxxxxxx/LPCOR/lpcor_get_index_by_name:
   lpcor local_sccp_phone_2 index 2
*Jun 25 09:52:20.114: //-1/xxxxxxxxxxxx/LPCOR/lpcor_get_index_by_peer:
   Return Lpcor Index 2 for Peer Tag 20003 *Jun 25 09:52:20.114:
//-1/xxxxxxxxxxxx/LPCOR/lpcor_index_is_valid:
   lpcor index 1 is valid
*Jun 25 09:52:20.114: //-1/xxxxxxxxxxxx/LPCOR/lpcor_policy_validate:
   Source LPCOR Index=1, Target LPCOR Policy=local_sccp_phone_2  -Traceback= 0x42949584
0x4219C430 0x4219CCDC 0x421A6B60 0x421A6D1C 0x421A75EC 0x421AB328 0x421ACA14 0x421B2518
0x421B2FFC 0x421B7614 0x4217F910 0x421F522C 0x421F89CC 0x421D84FC 0x422033BC *Jun 25
09:52:20.118: //-1/xxxxxxxxxxxx/LPCOR/lpcor_policy_validate:
   Validate Fail; lpcor (source[1] target[2])
```

Related Commands

Command	Description
debug ephone lpcor	Displays debugging information for calls using the LPCOR feature.
debug voip application lpcor	Enables debugging of the LPCOR application system.
show voice lpcor policy	Displays the LPCOR policy for the specified resource group.
voice lpcor enable	Enables LPCOR functionality on the Cisco Unified CME router.
voice lpcor policy	Creates a LPCOR policy for a resource group.

debug voip profile fax

To enable a set of debug commands for fax applications, use the **debug voip profile fax** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip profile fax [**mail**| **relay** [**application**| **signaling**]]

no debug voip profile fax

Syntax Description

mail	Enables the following set of debugs for an onramp or offramp fax mail call: • **debug csm voice** • **debug fax dmsp all** • **debug fax fmsp all** • **debug fax foip all** • **debug fax mmoip aaa all** • **debug fax mspi all** • **debug fax mta all** • **debug isdn q931** • **debug voip application all** • **debug voip application vxml all** • **debug voip ccapi all** • **debug voip dsm all** • **debug voip dspapi all** • **debug voip hpi all** • **debug voip ivr all** • **debug voip vtsp all**

	The following debug commands are enabled for access servers with MICA modem cards: • **debug fax fmsp all** • **debug fax mmoip aaa** • **debug fax mta all** • **debug isdn q931** • **debug voip application all** • **debug voip application vxml all** • **debug voip ccapi all** • **debug voip dsm all** • **debug voip dspapi all** • **debug voip hpi all** • **debug voip ivr all** • **debug voip vtsp all** The following debug options are enabled for access servers with universal port dial feature cards: • **debug fax dmsp all** • **debug fax fmsp all** • **debug fax foip all** • **debug fax mspi all** • **debug voip application vxml all** • **debug voip ivr all**
relay	Enables the **debug fax relay t30 all-level-1** and the sets specified by either the **application** or **signaling** keyword.
application	Enables the following set of debugs for fax relay applications: • **debug voip application all** • **debug voip application vxml all** • **debug voip ccapi all** • **debug voip dialpeer all** • **debug voip ivr all**

signaling	Enables the following set of debugs for fax relay signaling: • **debug cch323 all** • **debug ccsip error** • **debug ccsip messages** • **debug cdapi detail** • **debug cdapi events** • **debug csm voice** • **debug gtd error** • **debug gtd events** • **debug h225 asn1** • **debug h225 events** • **debug h225 q931** • **debug h245 events** • **debug h245 asn1** • **debug isdn q931** • **debug mgcp errors** • **debug mgcp events** • **debug mgcp media** • **debug mgcp packets** • **debug mgcp voipcac** • **debug rtpspi all** • **debug voip ccapi all** • **debug voip dsm all** • **debug voip dspapi all** • **debug voip hpi all** • **debug voip rawmsg** • **debug voip tsp all** • **debug voip vtsp all**

Command Default Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.

Usage Guidelines

The **debug voip profile fax** command can be used to use a set of debug commands at one time. Because this command generates a large amount of messages, router performance can be affected.

Caution The **debug voip profile fax** command generates debug messages from many VoIP components. The number of messages can impact the performance of your router. This command should only be used during low traffic periods.

Examples

Output has been omitted due to its large volume.

Related Commands

Command	Description
debug voip profile help	Displays the sets of commands supported by the **debug voip profile** commands.
debug voip profile modem	Enables a set of debug commands for modem applications.
debug voip profile voice	Enables a set of debug commands for voice.

debug voip profile help

To display the sets of debug commands supported by the **debug voip profile**commands, use the **debug voip profile help** command in privileged EXEC mode.

debug voip profile help

Command Default Debugging is not enabled.

Command Modes Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.

Usage Guidelines The **debug voip profile help** command displays the sets of debug commands supported by the **debug voip profile** commands. It does not display any debug output.

Examples The following is sample output from the **debug voip profile help** command:

```
Router# debug voip profile help

"debug voip profile modem relay signaling" includes:
   debug csm voice
   debug isdn q931
   debug cdapi detail
   debug cdapi events
   debug voip dspapi all
   debug voip hpi all
   debug voip vtsp all
   debug voip tsp all
   debug voip ccapi all
   debug cch323 all
   debug ccsip error
   debug ccsip messages
   debug mgcp errors
   debug mgcp events
   debug mgcp media
   debug mgcp packets
   debug mgcp voipcac
   debug voip dsm all

"debug voip profile voice application" includes:
   debug voip dialpeer all
   debug voip ccapi all
   debug voip ivr all
   debug voip application all
   debug voip application vxml all

"debug voip profile [ voice | modem pass-through ] signaling" includes:
   debug csm voice
   debug isdn q931
   debug cdapi detail
```

```
   debug cdapi events
   debug h225 asn1
   debug h225 events
   debug h225 q931
   debug h245 events
   debug h245 asn1
   debug voip dspapi all
   debug voip hpi all
   debug voip vtsp all
   debug voip tsp all
   debug voip ccapi all
   debug cch323 all
   debug rtpspi all
   debug ccsip error
   debug ccsip messages
   debug mgcp errors
   debug mgcp events
   debug mgcp media
   debug mgcp packets
   debug mgcp voipcac
   debug voip rawmsg
   debug gtd error
   debug gtd events
   debug voip dsm all

"debug voip profile fax mail" includes:
   debug csm voice
   debug isdn q931
   debug voip dspapi all
   debug voip hpi all
   debug voip vtsp all
   debug voip ccapi all
   debug voip ivr all
   debug voip application all
   debug voip application vxml all
   debug fmail client
   debug fmail server
   debug fax mta all
   debug fax receive all
   debug fax send all
   debug text-to-fax
   debug tiff reader
   debug tiff writer
   debug fax mmoip aaa
   debug voip dsm all

"debug voip profile fax relay application" includes:
   debug voip dialpeer all
   debug voip ccapi all
   debug voip ivr all
   debug voip application all
   debug voip application vxml all

"debug voip profile fax relay signaling" includes:
   debug csm voice
   debug isdn q931
   debug cdapi detail
   debug cdapi events
   debug h225 asn1
   debug h225 events
   debug h225 q931
   debug h245 events
   debug h245 asn1
   debug voip dspapi all
   debug voip hpi all
   debug voip vtsp all
   debug voip tsp all
   debug voip ccapi all
   debug cch323 all
   debug rtpspi all
   debug ccsip error
   debug ccsip messages
   debug mgcp errors
```

```
debug mgcp events
debug mgcp media
debug mgcp packets
debug mgcp voipcac
debug voip rawmsg
debug gtd error
debug gtd events
```

Related Commands

Command	Description
debug voip profile fax	Enables a set of debug commands for fax applications.
debug voip profile modem	Enables a set of debug commands for modem applications.
debug voip profile voice	Enables a set of debug commands for voice.

debug voip profile modem

To enable a set of debug commands for modem applications, use the **debug voip profile modem** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip profile modem [**pass-through signaling**| **relay signaling**]

no debug voip profile modem

Syntax Description

pass-through signaling	Enables the following set of debugs for modem pass-through signaling: • **debug cch323 all** • **debug ccsip error all** • **debug ccsip messages** • **debug cdapi detail** • **debug cdapi events** • **debug csm voice** • **debug gtd error** • **debug gtd events** • **debug h225 asn1** • **debug h225 events** • **debug h225 q931** • **debug isdn q931** • **debug mgsp errors all** • **debug mgcp events** • **debug mgcp media** • **debug mgcp packets** • **debug mgcp voipcac** • **debug rtpspi all** • **debug voip ccapi all** • **debug voip dsm all** • **debug voip rawmsg** • **debug voip tsp all** • **debug voip vtsp all** • **debug vpm all**

relay signaling	Enables the following set of debugs for modem relay signaling: • **debug voip ccapi all** • **debug voip vtsp all** • **debug cch323 all** • **debug ccsip error** • **debug ccsip messages all** • **debug mgcp all** • **debug mgcp events** • **debug mgcp media** • **debug mgcp packets** • **debug mgcp voipcac all** • **debug isdn q931**

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.

Usage Guidelines

The **debug voip profile modem** command can be used to use a set of debug commands at one time. Because this command causes a large amount of messages to be generated, router performance can be affected.

Caution The **debug voip profile modem** command generates debug messages from many VoIP components, which generates a large number of debug messages. The number of messages can affect the performance of your router. This command should only be used during low traffic periods.

Examples

Output has been omitted due to its large volume.

Related Commands

Command	Description
debug voip profile fax	Enables a set of debug commands for fax applications.
debug voip profile help	Displays the sets of commands supported by the **debug voip profile** commands.
debug voip profile voice	Enables a set of debug commands for voice.

debug voip profile voice

To enable a set of debug commands for voice, use the **debug voip profile voice** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip profile voice [**application**| **signaling**]

no debug voip profile voice

Syntax Description

application	Enables the following set of debugs for voice applications: • **debug voip applib** • **debug voip application vxml all** • **debug voip ccapi all** • **debug voip ivr all**

signaling	Enables the following set of debugs for voice signaling: • **debug cch323 all** • **debug ccsip error all** • **debug ccsip messages** • **debug cdapi detail** • **debug cdapi events** • **debug csm voice** • **debug gtd error** • **debug gtd events** • **debug h225 asn1** • **debug h225 events** • **debug h225 q931** • **debug isdn q931** • **debug mgsp errors all** • **debug mgcp events** • **debug mgcp media** • **debug mgcp packets** • **debug mgcp voipcac** • **debug rtpspi all** • **debug voip ccapi all** • **debug voip dsm all** • **debug voip rawmsg** • **debug voip tsp all** • **debug voip vtsp all** • **debug vpm all**

Command Default Debugging is not enabled.

Command Modes Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command was introduced.

Usage Guidelines

The **debug voip profile voice** command can be used to use a set of debug commands at one time. Because this command causes a large amount of messages to be generated, router performance can be affected.

Caution The **debug voip profile voice** command generates debug messages from many VoIP components, which generates a large number of debug messages. The number of messages can impact the performance of your router. This command should only be used during low traffic periods.

Examples

Output has been omitted due to its large volume.

Related Commands

Command	Description
debug voip profile fax	Enables a set of debug commands for fax applications.
debug voip profile help	Displays the sets of commands supported by the **debug voip profile** commands.
debug voip profile modem	Enables a set of debug commands for modem applications.

debug voip rawmsg

To display the raw message owner, length, and pointer, use the **debug voip rawmsg** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip rawmsg [detail]

no debug voip rawmsg [detail]

Syntax Description

detail	(Optional) Prints the contents of the raw message in hexadecimal.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(6)T	This command was introduced.
12.2(11)T	The new debug header was added to the following Cisco routers: Cisco 2600 series, Cisco 3620, Cisco 3640, and Cisco 3660; on the following universal gateways: Cisco AS5350, Cisco AS5400, and Cisco AS5850; on the following universal access servers: Cisco AS5300 and Cisco AS5800; and on the Cisco MC3810 multiservice access concentrators.

Usage Guidelines

We recommend that you log output from the **debug voip rawmsg**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 is sample output from the **debug voip rawmsg**command:

```
Router# debug voip rawmsg
voip rawmsg debugging is on
Router#
*Mar  1 01:16:25.155: //-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo: VoIP Raw Msg Al
loc from 1, Length 18 Body 638E0C5
```

These debug messages show that a raw message is allocated for this call. The pointer to the memory location for this raw message is 63075164.

```
*Mar  1 01:16:25.155: //-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo: Raw Message ALL
OCATED: ptr is 63075164, owner is 1, length is 18, msg is 638E0C54, type is 0, p
rotocol id is 0
```

The call control API (CCAPI) gets a setup indicator. It has no information about the callid (-1) and GUID *(*xxxxxxxxxxxx*)*.

```
*Mar  1 01:16:25.159: //-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind:
*Mar  1 01:16:25.159: Raw Message MaMa is TSP owner is CCAPI, length is 77, ptr
is 63075164, type is 0, protocol id is 2
```

The SSAPP at this stage knows about the CallEntry ID (30) but not about GUID (xxxxxxxxx) or the dial-peer (-1).

```
*Mar  1 01:16:25.163: //30/xxxxxxxxxxxx/SSAPP:-1:-1/ssaCallSetupInd:
*Mar  1 01:16:25.163: Raw Message MaMa is TSP owner is SSAPP, length is 77, ptr
is 63075164, type is 0, protocol id is 2
```

The SSAPP learns about the GUID (34C457CD802F) and also learns the incoming dial peer (10002).

```
*Mar  1 01:16:25.163: //30/34C457CD802F/SSAPP:10002:-1/ssaSetupPeer:
*Mar  1 01:16:25.163: Raw Message MaMa is TSP owner is SSAPP, length is 77, ptr
is 63075164, type is 0, protocol id is 2
```

The CCAPI gets the call proceeding for CallEntry ID 30. CCAPI still does not have a GUID (xxxxxxxxxxx).

```
*Mar  1 01:16:25.163: //30/xxxxxxxxxxxx/CCAPI/ccCallProceeding:
```

A new raw message buffer is created and the previous one is released.

```
*Mar  1 01:16:25.163: //-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo: VoIP Raw Msg Al
loc from 10, Length 77 Body 0
*Mar  1 01:16:25.167: //-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo: Raw Message ALL
OCATED: ptr is 630751EC, owner is 10, length is 77, msg is 638E0F0C, type is 0,
protocol id is 0
*Mar  1 01:16:25.167: //30/34C457CD802F/SSAPP:10002:-1/ssaSetupPeer:
*Mar  1 01:16:25.167: ssaSetupPeer: Saved rawmsgpp 630751EC len 77
IAM,
GCI,34c457cd14f911cc802f95f5fabb6b0f?)??p?34999
*Mar  1 01:16:25.167: //30/xxxxxxxxxxxx/CCAPI/ccCallSetupRequest:
*Mar  1 01:16:25.167: Raw Message MaMa is TSP owner is SSAPP, length is 77, ptr
is 63075164, type is 0, protocol id is 2
*Mar  1 01:16:25.167: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*Mar  1 01:16:25.167: Raw Message MaMa is TSP owner is SSAPP, length is 77, ptr
is 63075164, type is 0, protocol id is 2
```

The SSAPP gets a message indicating the digits were passed along the VoIP call leg to the terminating gateway. The CallEntry ID is 30, GUID is 34C457CD802F and the incoming dial peer is 10002.

```
*Mar  1 01:16:25.167: //30/34C457CD802F/SSAPP:10002:-1/ssaReportDigitsDone:
The old raw message 63075164 was freed. The new one is 630751EC.
*Mar  1 01:16:25.179: //-1/xxxxxxxxxxxx/CCAPI/ccFreeRawMsgInfo:
Router#Raw Message FREED: ptr is 63075164, owner is 3, length is 4D, msg is 638E
0DB0, type is 0, protocol id is 2
```

CCAPI got a call proceeding on the second call leg (31); it has no information about the GUID (xxxxxxxxx).

```
*Mar  1 01:16:25.223: //31/xxxxxxxxxxxx/CCAPI/cc_api_call_proceeding:
```

CCAPI got a call alert on the second call leg (31); still no information about the GUID (xxxxxxxxx).

```
*Mar  1 01:16:25.227: //31/xxxxxxxxxxxx/CCAPI/cc_api_call_alert:
```

The alert is sent to the first call leg (30), GUID 34C457CD802F.

```
*Mar  1 01:16:25.227: //30/34C457CD802F/SSAPP:10002:-1/ssaAlert:
*Mar  1 01:16:25.227: //30/xxxxxxxxxxxx/CCAPI/ccCallAlert:
The call is answered at this point and the CCAPI gets a call connect for the second call
leg (CallEntry ID is 31; GUID is xxxxxxxxx).
*Mar  1 01:16:40.975: //31/xxxxxxxxxxxx/CCAPI/cc_api_call_connected:
```

The call connect is sent to the first call leg (30), GUID 34C457CD802F.

```
*Mar  1 01:16:40.975: //30/34C457CD802F/SSAPP:10002:-1/ssaConnect:
*Mar  1 01:16:40.975: //30/xxxxxxxxxxxx/CCAPI/ccCallConnect:
```

The current raw message (ptr 630751EC) is released; a new one will be proclaimed when needed.

```
*Mar  1 01:16:40.975: //-1/xxxxxxxxxxxx/CCAPI/ccFreeRawMsgInfo: Raw Message FREE
D: ptr is 630751EC, owner is 10, length is 4D, msg is 638E0F0C, type is 0,
protocol id is 2
```

A new raw message (ptr 63075274) is proclaimed.

```
*Mar  1 01:17:04.007: //-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo: VoIP Raw Msg Al
loc from 1, Length 4 Body 638E1068
*Mar  1 01:17:04.007: //-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo: Raw Message ALL
OCATED: ptr is 63075274, owner is 1, length is 4, msg is 638E1068, type is 0, protocol id
is 0
```

The call terminates now. CCAPI detects a call disconnect from the first call leg (30) with no GUID (xxxxxxxxx).

```
*Mar  1 01:17:04.007: //30/xxxxxxxxxxxx/CCAPI/cc_api_call_disconnected:
*Mar  1 01:17:04.007: Raw Message MaMa is TSP owner is CCAPI, length is 4, ptr i
s 63075274, type is 0, protocol id is 2
```

The disconnect is sent to the first call leg (30), GUID (34C457CD802F).

```
*Mar  1 01:17:04.011: //30/34C457CD802F/SSAPP:10002:14/ssaDisconnected:
*Mar  1 01:17:04.011: Raw Message MaMa is TSP owner is SSAPP, length is 4, ptr i
s 63075274, type is 0, protocol id is 2
```

The CCAPI disconnects both call legs (incoming 30 and outgoing 31).

```
*Mar  1 01:17:04.011: //30/xxxxxxxxxxxx/CCAPI/ccCallDisconnect:
*Mar  1 01:17:04.011: //31/xxxxxxxxxxxx/CCAPI/ccCallDisconnect:
*Mar  1 01:17:04.011: Raw Message MaMa is TSP owner is SSAPP, length is 4, ptr i
s 63075274, type is 0, protocol id is 2
```

The raw message is released.

```
*Mar  1 01:17:04.015: //-1/xxxxxxxxxxxx/CCAPI/ccFreeRawMsgInfo: Raw Message FREE
D: ptr is 63075274, owner is 3, length is 4, msg is 638E1068, type is 0, protocol id is 2
```

The following example shows output when you use the **debug voip rawmsg detail** command. This example shows that the CCAPI layer received an indication for call setup. The detailed raw message dumps the hex of the message. This output is used to track down data pointing to different variables within the software modules.

```
Router# debug voip rawmsg detail
*Mar  6 17:03:24.169://-1/xxxxxxxxxxxx/CCAPI/ccAllocRawMsgInfo:VoIP Raw Msg Al
loc from 5, Length 0 Body 0
*Mar  6 17:03:24.173://-1/xxxxxxxxxxxx/CCAPI/cc_api_call_setup_ind:
*Mar  6 17:03:24.173:Raw Message MaMa is CCAPI owner is CCAPI, length is 59, ptr is 63045C14,
 type is 0, protocol id is 18
*Mar  6 17:03:24.173:Raw Message is :1C 39 9E 01 00 03 67 74 64 00 00 00 2E 49
 41 4D 2C 0D 0A 47 43 49 2C 35 33 39 61 30 35 39 64 31 39 36 62 31 31 63 63 38 3
9 30 63 39 30 37 65 31 65 31 37 35 61 34 37 0D 0A 0D 0A
```

Related Commands

Command	Description
debug cdapi	Displays information about the call distributor application programming interface.
debug tsp	Displays information about the telephony service provider.
voice call debug	Allows configuration of the voice call debug output.

debug voip rtcp

To enable debugging for Real-Time Transport Control Protocol (RTCP) packets, use the **debug voip rtcp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip rtcp {**error**| **packet**| **session**}

no debug voip rtcp

Syntax Description

error	Prints out a trace for error cases.
packet	Provides debug output for RTCP packets.
session	Provides all session debug information.

Command Default

Debugging for RTCP packets is not enabled.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.2(2)T	This command was introduced.
12.2(11)T	This command was implemented on the Cisco AS5300, Cisco AS5400, and Cisco AS5850.
12.2(15)T	This command was implemented on the Cisco 1751 and Cisco 1760.

Usage Guidelines

When used without a keyword, this command turns on debugging for all events. This command severely impacts performance; use with caution.

Examples

The following is sample output from the **debug voip rtcp** command:

```
Router# debug voip rtcp
1w0d: voip_rtcp_create_session: callID=37, dstCallID=36 laddr=172.19.169.85, lp0
1w0d: voip_rtcp_get_cname: cname=0.0.0@172.19.169.85
1w0d: voip_rtcp_send_event: event=EV_NEW
1w0d: voip_rtcp_new: rtcp_interval=1893
Router#
1w0d: voip_rtcp_send_event: event=EV_STATS
1w0d: voip_rtcp_stats_req: rtcp_interval=3448
1w0d: voip_rtcp_stats_req:delay=45 lost_packets=0 rtt=0
Router#
1w0d: recv:
```

```
1w0d: SR: ssrc=0x1272A94D sr_ntp_h=0xAF44E045 sr_ntp_l=0xA6CE39C sr_timestamp=02
1w0d: SDES: ssrc=0x1272A94D name=1 len=19 data=0.0.0@172.19.169.77
1w0d: rtcp_round_trip_delay: ssrc=0x1D86A955
Router#
1w0d: voip_rtcp_send_event: event=EV_STATS
1w0d: voip_rtcp_stats_req: rtcp_interval=6394
1w0d: voip_rtcp_stats_req:delay=40 lost_packets=0 rtt=0
1w0d: recv:
1w0d: SR: ssrc=0x1272A94D sr_ntp_h=0xAF44E047 sr_ntp_l=0xFFB007F6 sr_timestamp=6
1w0d: SDES: ssrc=0x1272A94D name=1 len=19 data=0.0.0@172.19.169.77
1w0d: rtcp_round_trip_delay: ssrc=0x1D86A955
Router#
1w0d: voip_rtcp_remove_ccb
1w0d: voip_rtcp_send_event: event=EV_DESTROY
1w0d: voip_rtcp_destroy_idle
1w0d: voip_rtcp_close_session
1w0d: Cleaning up sess=62F95F58, sp=19544, dp=17130
```

debug voip rtp

To enable debugging for Real-Time Transport Protocol (RTP) named event packets, use the **debug voip rtp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip rtp {**error**| **session** [**conference**| **dtmf-relay**| **event**| **multicast**| **named-event** [*payload-type*]| **nse**| **text-relay**]| **packet** [**callid** *id-number packet-number*| **remote-ip** *ip-address* **remote-port** *port-number packet-number*]}

no debug voip rtp

Syntax Description

error	Prints out a trace for error cases.
session	Provides all session debug information. If used with a keyword, supplies more specific debug information according to the keywords used.
conference	(Optional) Provides debug information for conference packets.
dtmf-relay	(Optional) Provides debug information for dual-tone multifrequency (DTMF) packets.
event	(Optional) Enables VoIP RTP session generic event debugging trace.
multicast	(Optional) Provides debug information for multicast packets.
named-event	(Optional) Provides debug information for named telephony event (NTE) packets.
nse	(Optional) Provides debug information for named signaling events (NSEs).
text-relay	(Optional) Provides debug information for text-relay packets.
packet	Enables VoIP RTP packet debugging trace.
callid *id-number packet-number*	(Optional) Provides debug information for a specific call ID number (obtained by using the **show voip rtp connections** command). The *packet-number*argument specifies the number of packets to trace so that the display is not flooded.

remote-ip *ip-address* **remote-port** *port-number packet-number*	(Optional) Provides debug information for a remote IP address and RTP port number. The *packet-number*argument specifies the number of packets to trace so that the display is not flooded.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.2(2)XB	This command was introduced.
12.2(8)T	This command was integrated into Cisco IOS Release 12.2(8)T.
12.4(4)XC	This command was implemented on the Cisco AS5300, Cisco AS5400, and Cisco AS5850.
12.2(15)T	This command was implemented on the Cisco 1751 and Cisco 1760.
12.4(6)T	The **text-relay** keyword was added.
12.2SX	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.

Usage Guidelines

This command severely impacts performance and should be used only for single-call debug capture.

Examples

The following example shows debugging output for the **debug voip rtp session named-event**command. The example is for a gateway that sends digits 1, 2, 3, then receives digits 9,8,7. The payload type, event ID, and additional packet payload are shown in each log.

The first three packets indicate the start of the tone (initial packet and two redundant). The last three packets indicate the end of the tone (initial packet and two redundant). The packets in between are refresh packets that are sent every 50 milliseconds (without redundancy).

```
Router# debug voip rtp session named-event
00:09:29:           Pt:99     Evt:1       Pkt:03 00 00  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:03 00 00  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:03 00 00  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:03 01 90  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:03 03 20  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:03 04 B0  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:83 04 C8  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:83 04 C8  <<<Rcv>
00:09:29:           Pt:99     Evt:1       Pkt:83 04 C8  <<<Rcv>
00:09:29:           Pt:99     Evt:2       Pkt:03 00 00  <<<Rcv>
00:09:29:           Pt:99     Evt:2       Pkt:03 00 00  <<<Rcv>
00:09:29:           Pt:99     Evt:2       Pkt:03 00 00  <<<Rcv>
00:09:29:           Pt:99     Evt:2       Pkt:03 01 90  <<<Rcv>
00:09:29:           Pt:99     Evt:2       Pkt:03 03 20  <<<Rcv>
```

```
00:09:29:          Pt:99     Evt:2       Pkt:03 04 B0  <<<Rcv>
00:09:29:          Pt:99     Evt:2       Pkt:83 05 18  <<<Rcv>
00:09:29:          Pt:99     Evt:2       Pkt:83 05 18  <<<Rcv>
00:09:29:          Pt:99     Evt:2       Pkt:83 05 18  <<<Rcv>
00:09:29:          Pt:99     Evt:3       Pkt:03 00 00  <<<Rcv>
00:09:29:          Pt:99     Evt:3       Pkt:03 00 00  <<<Rcv>
00:09:29:          Pt:99     Evt:3       Pkt:03 00 00  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:03 01 90  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:03 03 20  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:03 04 B0  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:03 06 40  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:83 06 80  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:83 06 80  <<<Rcv>
00:09:30:          Pt:99     Evt:3       Pkt:83 06 80  <<<Rcv>
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 01 90
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 03 20
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 04 B0
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:02 06 40
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:82 06 58
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:82 06 58
00:09:31:  <Snd>>> Pt:99     Evt:9       Pkt:82 06 58
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 01 90
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 03 20
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 04 B0
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:02 06 40
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:82 06 90
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:82 06 90
00:09:31:  <Snd>>> Pt:99     Evt:8       Pkt:82 06 90
00:09:31:  <Snd>>> Pt:99     Evt:7       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:7       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:7       Pkt:02 00 00
00:09:31:  <Snd>>> Pt:99     Evt:7       Pkt:02 01 90
00:09:31:  <Snd>>> Pt:99     Evt:7       Pkt:02 03 20
00:09:31:  <Snd>>> Pt:99     Evt:7       Pkt:02 04 B0
00:09:32:  <Snd>>> Pt:99     Evt:7       Pkt:02 06 40
00:09:32:  <Snd>>> Pt:99     Evt:7       Pkt:82 06 58
00:09:32:  <Snd>>> Pt:99     Evt:7       Pkt:82 06 58
00:09:32:  <Snd>>> Pt:99     Evt:7       Pkt:82 06 58
```

The following example shows debugging output for the **debug voip rtp session text-relay**command:

```
Router# debug voip rtp session text-relay
Pt:119    Evt:0     4     247   37    128    Cnt:F7 4B  <Snd>>>
```

Related Commands

Command	Description
text relay protocol	Configures the system-wide protocol type for text packets transmitted between gateways.
text relay rtp	Configures the RTP payload type and redundancy level.

debug voip settlement all

To enable debugging in all settlement areas, use the **debug voip settlement all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement all [**enter**| **error**| **exit**| **misc**| **network**| **security**| **transaction**]

no debug voip settlement all [**enter**| **error**| **exit**| **misc**| **network**| **security**| **transaction**]

Syntax Description

enter	(Optional) Displays all entrances.
error	(Optional) Displays information only if an error occurs.
exit	(Optional) Displays all exits.
misc	(Optional) Displays the details on the code flow of each transaction.
network	(Optional) Displays network connectivity data.
security	(Optional) Displays security and encryption errors.
transaction	(Optional) Displays transaction information.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Usage Guidelines

The **debug voip settlement all** command enables the following debug settlement commands:

- **debug voip settlement enter**
- **debug voip settlement error**
- **debug voip settlement exit**
- **debug voip settlement misc**
- **debug voip settlement network**

- **debug voip settlement security**
- **debug voip settlement transaction**

debug voip settlement enter

To show all the settlement function entrances, enter the **debug voip settlement enter** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement enter

no debug voip settlement enter

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

The following is sample output from the **debug voip settlement enter** command:

```
Router# debug voip settlement enter

00:43:40:OSP:ENTER:OSPPMimeMessageCreate()
00:43:40:OSP:ENTER:OSPPMimeMessageInit()
00:43:40:OSP:ENTER:OSPPMimeMessageSetContentAndLength()
00:43:40:OSP:ENTER:OSPPMimeMessageBuild()
00:43:40:OSP:ENTER:OSPPMimeDataFree()
00:43:40:OSP:ENTER:OSPPMimePartFree()
00:43:40:OSP:ENTER:OSPPMimePartFree()
00:43:40:OSP:ENTER:OSPPMsgInfoAssignRequestMsg()
00:43:40:OSP:ENTER:osppHttpSelectConnection
00:43:40:OSP:ENTER:OSPPSockCheckServicePoint() ospvConnected = <1>
00:43:40:OSP:ENTER:OSPPSockWaitTillReady()
00:43:40:OSP:ENTER:osppHttpBuildMsg()
00:43:40:OSP:ENTER:OSPPSSLSessionWrite()
00:43:40:OSP:ENTER:OSPPSockWrite()
00:43:40:OSP:ENTER:OSPPSockWaitTillReady()
```

debug voip settlement error

To show all the settlement errors, enter the **debug voip settlement error** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement error

no debug voip settlement error

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

```
-1:OSP internal software error.
16:A bad service was chosen.
17:An invalid parameter was passed to OSP.
9010:Attempted to access an invalid pointer.
9020:A time related error occurred.
10010:OSP provider module failed initialization.
10020:OSP provider tried to access a NULL pointer.
10030:OSP provider could not fine transaction collection.
10040:OSP provider failed to obtain provider space.
10050:OSP provider tried to access an invalid handle.
10060:OSP provider has reached the maximum number of providers.
11010:OSP transaction tried to delete a transaction which was not allowed.
11020:OSP transaction tried a transaction which does not exist.
11030:OSP transaction tried to start a transaction, but data had already been delivered.
11040:OSP transaction could not identify the response given.
11050:OSP transaction failed to obtain transaction space.
11060:OSP transaction failed (possibly ran out) to allocate memory.
11070:OSP transaction tried to perform a transaction which is not allowed.
11080:OSP transaction found no more responses.
11090:OSP transaction could not find a specified value.
11100:OSP transaction did not have enough space to copy.
11110:OSP transaction - call id did not match destination.
11120:OSP transaction encountered an invalid entry.
11130:OSP transaction tried to use a token too soon.
11140:OSP transaction tried to use a token too late.
11150:OSP transaction - source is invalid.
11160:OSP transaction - destination is invalid.
11170:OSP transaction - calling number is invalid.
11180:OSP transaction - called number is invalid.
11190:OSP transaction - call id is invalid.
```

```
11200:OSP transaction - authentication id is invalid.
11210:OSP transaction - call id was not found
11220:OSP transaction - The IDS of the called number was invalid.
11230:OSP transaction - function not implemented.
11240:OSP transaction tried to access an invalid handle.
11250:OSP transaction returned an invalid return code.
11260:OSP transaction reported an invalid status code.
11270:OSP transaction encountered an invalid token.
11280:OSP transaction reported a status which could not be identified.
11290:OSP transaction in now valid after it was not found.
11300:OSP transaction could not find the specified destination.
11310:OSP transaction is valid until not found.
11320:OSP transaction - invalid signaling address.
11330:OSP transaction could not find the ID of the transmitter.
11340:OSP transaction could not find the source number.
11350:OSP transaction could not find the destination number.
11360:OSP transaction could not find the token.
11370:OSP transaction could not find the list.
11380:OSP transaction was not allowed to accumulate.
11390:OSP transaction - transaction usage was already reported.
11400:OSP transaction could not find statistics.
11410:OSP transaction failed to create new statistics.
11420:OSP transaction made an invalid calculation.
11430:OSP transaction was not allowed to get the destination.
11440:OSP transaction could not fine the authorization request.
11450:OSP transaction - invalid transmitter ID.
11460:OSP transaction could not find any data.
11470:OSP transaction found no new authorization requests.
12010:OSP security did not have enough space to copy.
12020:OSP security received and invalid argument.
12030:OSP security could not find the private key.
12040:OSP security encountered an un-implemented function.
12050:OSP security ran out of memory.
12060:OSP security received an invalid signal.
12065:OSP security could not initialize the SSL database.
12070:OSP security could not find space for the certificate.
12080:OSP security has no local certificate info defined.
12090:OSP security encountered a zero length certificate.
12100:OSP security encountered a certificate that is too big.
12110:OSP security encountered an invalid certificate.
12120:OSP security encountered a NULL certificate.
12130:OSP security has too many certificates.
12140:OSP security has no storage provided.
12150:OSP security has no private key.
12160:OSP security encountered an invalid context.
12170:OSP security was unable to allocate space.
12180:OSP security - CA certificates do not match.
12190:OSP security found no authority certificates
12200:OSP security - CA certificate index overflow.
13010:OSP error message - failed to allocate memory.
13110:OSP MIME error - buffer is too small.
13115:OSP MIME error - failed to allocate memory.
13120:OSP MIME error - could not find variable.
13125:OSP MIME error - no input was found.
13130:OSP MIME error - invalid argument.
13135:OSP MIME error - no more space.
13140:OSP MIME error - received an invalid type.
13145:OSP MIME error - received an invalid subtype.
13150:OSP MIME error - could not find the specified protocol.
13155:OSP MIME error - could not find MICALG.
13160:OSP MIME error - boundary was not found.
13165:OSP MIME error - content type was not found.
13170:OSP MIME error - message parts were not found.
13301:OSP XML error - received incomplete XML data.
13302:OSP XML error - bad encoding of XML data.
13303:OSP XML error - bad entity in XML data.
13304:OSP XML error - bad name in XML data.
13305:OSP XML error - bad tag in XML data.
13306:OSP XML error - bad attribute in XML data.
13307:OSP XML error - bad CID encoding in XML data.
13308:OSP XML error - bad element found in XML data.
13309:OSP XML error - no element found in XML data.
13310:OSP XML error - no attribute found in XML data.
```

```
13311:OSP XML error - OSP received invalid arguments.
13312:OSP XML error - failed to create a new buffer.
13313:OSP XML error - failed to get the size of a buffer.
13314:OSP XML error - failed to send the buffer.
13315:OSP XML error - failed to read a block from the buffer.
13316:OSP XML error - failed to allocate memory.
13317:OSP XML error - could not find the parent.
13318:OSP XML error - could not find the child.
13319:OSP XML error - data type not found in XML data.
13320:OSP XML error - failed to write a clock to the buffer.
13410:OSP data error - no call id preset.
13415:OSP data error - no token present.
13420:OSP data error - bad number presented.
13425:OSP data error - no destination found.
13430:OSP data error - no usage indicator present.
13435:OSP data error - no status present.
13440:OSP data error - no usage configured.
13445:OSP data error - no authentication indicator.
13450:OSP data error - no authentication request.
13455:OSP data error - no authentication response.
13460:OSP data error - no authentication configuration.
13465:OSP data error - no re-authentication request.
13470:OSP data error - no re-authentication response.
13475:OSP data error - invalid data type present.
13480:OSP data error - no usage information available.
13485:OSP data error - no token info present.
13490:OSP data error - invalid data present.
13500:OSP data error - no alternative info present.
13510:OSP data error - no statistics available.
13520:OSP data error - no delay present.
13610:OSP certificate error - memory allocation failed.
14010:OSP communications error - invalid communication size.
14020:OSP communications error - bad communication value.
14030:OSP communications error - parser error.
14040:OSP communications error - no more memory available.
14050:OSP communications error - communication channel currently in use.
14060:OSP communications error - invalid argument passed.
14070:OSP communications error - no service points present.
14080:OSP communications error - no service points available.
14085:OSP communications error - thread initialization failed.
14086:OSP communications error - communications is shutdown.
14110:OSP message queue error - no more memory available.
14120:OSP message queue error - failed to add a request.
14130:OSP message queue error - no event queue present.
14140:OSP message queue error - invalid arguments passed.
14210:OSP HTTP error - 100 - bad header.
14220:OSP HTTP error - 200 - bad header.
14221:OSP HTTP error - 400 - bad request.
14222:OSP HTTP error - bas service port present.
14223:OSP HTTP error - failed to add a request.
14230:OSP HTTP error - invalid queue present.
14240:OSP HTTP error - bad message received.
14250:OSP HTTP error - invalid argument passed.
14260:OSP HTTP error - memory allocation failed.
14270:OSP HTTP error - failed to create a new connection.
14280:OSP HTTP error - server error.
14290:OSP HTTP error - HTTP server is shutdown.
14292:OSP HTTP error - failed to create a new SSL connection.
14295:OSP HTTP error - failed to create a new SSL context.
14297:OSP HTTP error - service unavailable.
14300:OSP socket error - socket select failed.
14310:OSP socket error - socket receive failed.
14315:OSP socket error - socket send failed.
14320:OSP socket error - failed to allocate memory for the receive buffer.
14320:OSP socket error - socket reset.
14330:OSP socket error - failed to create the socket.
14340:OSP socket error - failed to close the socket.
14350:OSP socket error - failed to connect the socket.
14360:OSP socket error - failed to block I/O on the socket.
14370:OSP socket error - failed to disable nagle on the socket.
14400:OSP SSL error - failed to allocate memory.
14410:OSP SSL error - failed to initialize the context.
14420:OSP SSL error - failed to retrieve the version.
```

```
14430:OSP SSL error - failed to initialize the session.
14440:OSP SSL error - failed to attach the socket.
14450:OSP SSL error - handshake failed.
14460:OSP SSL error - failed to close SSL.
14470:OSP SSL error - failed to read from SSL.
14480:OSP SSL error - failed to write to SSL.
14490:OSP SSL error - could not get certificate.
14495:OSP SSL error - no root certificate found.
14496:OSP SSL error - failed to set the private key.
14497:OSP SSL error - failed to parse the private key.
14498:OSP SSL error - failed to add certificates.
14499:OSP SSL error - failed to add DN.
15410:OSP utility error - not enough space for copy.
15420:OSP utility error - no time stamp has been created.
15430:OSP utility error - value not found.
15440:OSP utility error - failed to allocate memory.
15450:OSP utility error - invalid argument passed.
15500:OSP buffer error - buffer is empty.
15510:OSP buffer error - buffer is incomplete.
15980:OSP POW error.
15990:OSP Operating system conditional variable timeout.
16010:OSP X509 error - serial number undefined.
16020:OSP X509 error - certificate undefined.
16030:OSP X509 error - invalid context.
16040:OSP X509 error - decoding error.
16050:OSP X509 error - unable to allocate space.
16060:OSP X509 error - invalid data present.
16070:OSP X509 error - certificate has expired.
16080:OSP X509 error - certificate not found.
17010:OSP PKCS1 error - tried to access invalid private key pointer
17020:OSP PKCS1 error - unable to allocate space.
17030:OSP PKCS1 error - invalid context found.
17040:OSP PKCS1 error - tried to access NULL pointer.
17050:OSP PKCS1 error - private key overflow.
18010:OSP PKCS7 error - signer missing.
18020:OSP PKCS7 error - invalid signature found.
18020:OSP PKCS7 error - unable to allocate space.
18030:OSP PKCS7 error - encoding error.
18040:OSP PKCS7 error - tried to access invalid pointer.
18050:OSP PKCS7 error - buffer overflow.
19010:OSP ASN1 error - tried to access NULL pointer.
19020:OSP ASN1 error - invalid element tag found.
19030:OSP ASN1 error - unexpected high tag found.
19040:OSP ASN1 error - invalid primitive tag found.
19050:OSP ASN1 error - unable to allocate space.
19060:OSP ASN1 error - invalid context found.
19070:OSP ASN1 error - invalid time found.
19080:OSP ASN1 error - parser error occurred.
19090:OSP ASN1 error - parsing complete.
19100:OSP ASN1 error - parsing defaulted.
19110:OSP ASN1 error - length overflow.
19120:OSP ASN1 error - unsupported tag found.
19130:OSP ASN1 error - object ID not found.
19140:OSP ASN1 error - object ID mismatch.
19150:OSP ASN1 error - unexpected int base.
19160:OSP ASN1 error - buffer overflow.
19170:OSP ASN1 error - invalid data reference ID found.
19180:OSP ASN1 error - no content value for element found.
19190:OSP ASN1 error - integer overflow.
20010:OSP Crypto error - invalid parameters found.
20020:OSP Crypto error - unable to allocate space.
20030:OSP Crypto error - could not verify signature.
20040:OSP Crypto error - implementation specific error.
20050:OSP Crypto error - tried to access invalid pointer.
20060:OSP Crypto error - not enough space to perform operation.
21010:OSP PKCS8 error - invalid private key pointer found.
21020:OSP PKCS8 error - unable to allocate space for operation.
21030:OSP PKCS8 error - invalid context found.
21040:OSP PKCS8 error - tried to access NULL pointer.
21050:OSP PKCS8 error - private key overflow.
22010:OSP Base 64 error - encode failed.
22020:OSP Base 64 error - decode failed.
22510:OSP audit error - failed to allocate memory.
```

```
156010:OSP RSN failure error - no data present.
156020:OSP RSN failure error - data is invalid.
```

Examples

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

```
Router# debug voip settlement error
00:45:50:OSP:OSPPSockProcessRequest:http recv init header failed
00:45:50:OSP:osppHttpSetupAndMonitor:attempt#0 on http=0x6141A514, limit=1 error=14310
```

debug voip settlement exit

To show all the settlement function exits, enter the **debug voip settlement exit** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement exit

no debug voip settlement exit

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

The following is sample output from the **debug voip settlement exit**command:

```
Router# debug voip settlement exit

01:21:10:OSP:EXIT :OSPPMimeMessageInit()
01:21:10:OSP:EXIT :OSPPMimeMessageSetContentAndLength()
01:21:10:OSP:EXIT :OSPPMimeMessageBuild()
01:21:10:OSP:EXIT :OSPPMimePartFree()
01:21:10:OSP:EXIT :OSPPMimePartFree()
01:21:10:OSP:EXIT :OSPPMimeDataFree()
01:21:10:OSP:EXIT :OSPPMimeMessageCreate()
01:21:10:OSP:EXIT :OSPPMsgInfoAssignRequestMsg()
01:21:10:OSP:EXIT :osppHttpSelectConnection
01:21:10:OSP:EXIT :OSPPSockCheckServicePoint() isconnected(1)
01:21:10:OSP:EXIT :osppHttpBuildMsg()
01:21:10:OSP:EXIT :OSPPSockWrite() (0)
01:21:10:OSP:EXIT :OSPPSSLSessionWrite() (0)
01:21:10:OSP:EXIT :OSPPSSLSessionRead() (0)
01:21:10:OSP:EXIT :OSPPSSLSessionRead() (0)
01:21:10:OSP:EXIT :OSPPHttpParseHeader
01:21:10:OSP:EXIT :OSPPHttpParseHeader
01:21:10:OSP:EXIT :OSPPSSLSessionRead() (0)
01:21:10:OSP:EXIT :OSPPUtilMemCaseCmp()
```

debug voip settlement misc

To show the details on the code flow of each settlement transaction, enter the **debug voip settlement misc** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement misc

no debug voip settlement misc

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

The following is sample output from the **debug voip settlement misc**command:

```
Router# debug voip settlement misc

00:52:03:OSP:osp_authorize:callp=0x6142770C
00:52:03:OSP:OSPPTransactionRequestNew:ospvTrans=0x614278A8
00:52:03:OSP:osppCommMonitor:major:minor=(0x2:0x1)
00:52:03:OSP:HTTP connection:reused
00:52:03:OSP:osppHttpSetupAndMonitor:HTTP=0x6141A514, QUEUE_EVENT from eventQ=0x6141A87C,
comm=0x613F16C4, msginfo=0x6142792C
00:52:03:OSP:osppHttpSetupAndMonitor:connected = <TRUE>
00:52:03:OSP:osppHttpSetupAndMonitor:HTTP=0x6141A514, build msginfo=0x6142792C, trans=0x2
00:52:04:OSP:osppHttpSetupAndMonitor:HTTP=0x6141A514, msg built and sent:error=0,
msginfo=0x6142792C
00:52:04:OSP:osppHttpSetupAndMonitor:monitor exit. errorcode=0
00:52:04:OSP:osppHttpSetupAndMonitor:msginfo=0x6142792C, error=0, shutdown=0
00:52:04:OSP:OSPPMsgInfoProcessResponse:msginfo=0x6142792C, err=0, trans=0x614278A8, handle=2
00:52:04:OSP:OSPPMsgInfoChangeState:transp=0x614278A8, msgtype=12 current state=2
00:52:04:OSP:OSPPMsgInfoChangeState:transp=0x614278A8, new state=4
00:52:04:OSP:OSPPMsgInfoProcessResponse:msginfo=0x6142792C, context=0x6142770C, error=0
00:52:04:OSP:osp_get_destination:trans_handle=2, get_first=1, callinfop=0x614275E0
00:52:04:OSP:osp_get_destination:callinfop=0x614275E0 get dest=1.14.115.51,
validafter=1999-01-20T02:04:32Z, validuntil=1999-01-20T02:14:32Z
00:52:04:OSP:osp_parse_destination:dest=1.14.115.51
00:52:04:OSP:osp_get_destination:callinfop=0x614275E0, error=0, ip_addr=1.14.115.51, credit=60
00:52:06:OSP:stop_settlement_ccapi_accounting:send report for callid=0x11, transhandle=2
00:52:06:OSP:osp_report_usage:transaction=2, duration=0, lostpkts=0, lostfrs=0, lostpktr=0,
 lostfrr=0
```

debug voip settlement network

To show all the messages exchanged between a router and a settlement provider, enter the **debug voip settlement network** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement network

no debug voip settlement network

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Usage Guidelines

Using the **debug voip settlement network** command shows messages, in detail, in HTTP and XML formats.

Examples

The following is sample output from the **debug voip settlement network**command:

```
Router# debug voip settlement network

00:47:25:OSP:HTTP connection:reused
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=0, timeout=0, select=1
00:47:25:OSP:osppHttpBuildAndSend():http=0x6141A514 sending:
POST /scripts/simulator.dll?handler HTTP/1.1
Host:1.14.115.12
content-type:text/plain
Content-Length:439
Connection:Keep-Alive
Content-Type:text/plain
Content-Length:370
<?xml version="1.0"?><Message messageId="1" random="8896">
<AuthorisationRequest componentId="1">
<Timestamp>
1993-03-01T00:47:25Z</Timestamp>
<CallId>
<![CDATA[12]]></CallId>
<SourceInfo type="e164">
5551111</SourceInfo>
<DestinationInfo type="e164">
5552222</DestinationInfo>
<Service/>
<MaximumDestinations>
3</MaximumDestinations>
</AuthorisationRequest>
```

```
</Message>
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=0, timeout=1, select=1
00:47:25:OSP:OSPM_SEND:bytes_sent = 577
00:47:25:OSP:OSPPSockProcessRequest:SOCKFD=0, Expecting 100, got
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=1, timeout=1, select=1
00:47:25:OSP:OSPPSSLSessionRead() recving 1 bytes:
HTTP/1.1 100 Continue
Server:Microsoft-IIS/4.0
Date:Wed, 20 Jan 1999 02:01:54 GMT
00:47:25:OSP:OSPPSockProcessRequest:SOCKFD=0, Expecting 200, got
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=1, timeout=1, select=1
00:47:25:OSP:OSPPSSLSessionRead() recving 1 bytes:
HTTP/1.1 200 OK
Server:Microsoft-IIS/4.0
Date:Wed, 20 Jan 1999 02:01:54 GMT
Connection:Keep-Alive
Content-Type:multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1;
boundary=bar
Content-Length:1689
00:47:25:OSP:OSPPSockProcessRequest:SOCKFD=0, error=0, HTTP response
00:47:25:OSP:OSPPSockWaitTillReady:HTTPCONN=0x6141A514, fd=0
00:47:25:OSP:OSPPSockWaitTillReady:read=1, timeout=1, select=1
00:47:25:OSP:OSPPSSLSessionRead() recving 1689 bytes:
--bar
Content-Type:text/plain
Content-Length:1510
<?xml version="1.0"?><Message messageId="1" random="27285">
<AuthorisationResponse componentId="1">
<Timestamp>
1999-01-20T02:01:54Z</Timestamp>
<Status>
<Description>
success</Description>
<Code>
200</Code>
</Status>
<TransactionId>
101</TransactionId>
<Destination>
<AuthorityURL>
http://www.myauthority.com</AuthorityURL>
<CallId>
<![CDATA[12]]></CallId>
<DestinationInfo type="e164">
5552222</DestinationInfo>
<DestinationSignalAddress>
1.14.115.51</DestinationSignalAddress>
<Token encoding="base64">
[illegible]
<UsageDetail>
<Amount>
60</Amount>
<Increment>
1</Increment>
<Service/>
<Unit>
s</Unit>
</UsageDetail>
<ValidAfter>
1999-01-20T01:59:54Z</ValidAfter>
<ValidUntil>
1999-01-20T02:09:54Z</ValidUntil>
</Destination>
<transnexus.com:DelayLimit critical="False">
1000</transnexus.com:DelayLimit>
<transnexus.com:DelayPreference critical="False">
1</transnexus.com:DelayPreference>
</AuthorisationResponse>
</Message>
--bar
```

```
Content-Type:application/pkcs7-signature
Content-Length:31
This is your response signature
--bar--
```

debug voip settlement security

To show all the tracing related to security, such as Secure Socket Layer (SSL) or Secure Multipurpose Internet Mail Extensions (S/MIME), enter the **debug voip settlement security** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement security

no debug voip settlement security

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

Not available because of security issues.

debug voip settlement ssl

To display information about the Secure Socket Layer (SSL) connection, use the **debug voip settlement ssl**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip settlement ssl

no debug voip settlement ssl

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

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

Usage Guidelines

For complete information about the SSL connection, use the **debug voip settlement ssl** command if you see one of the following errors generated from the **debug voip settlement error** command.

```
14400:OSP SSL error - failed to allocate memory.
14410:OSP SSL error - failed to initialize the context.
14420:OSP SSL error - failed to retrieve the version.
14430:OSP SSL error - failed to initialize the session.
14440:OSP SSL error - failed to attach the socket.
14450:OSP SSL error - handshake failed.
14460:OSP SSL error - failed to close SSL.
14470:OSP SSL error - failed to read from SSL.
14480:OSP SSL error - failed to write to SSL.
14490:OSP SSL error - could not get certificate.
14495:OSP SSL error - no root certificate found.
14496:OSP SSL error - failed to set the private key.
14497:OSP SSL error - failed to parse the private key.
14498:OSP SSL error - failed to add certificates.
14499:OSP SSL error - failed to add DN.
```

Examples

The following example shows the debug output when the SSL is making a good connection to the Open Settlement Protocol server:

```
*May 15 11:53:42.871:OSP:
*May 15 11:53:42.871:OSPPSSLConnect:****** SSL HANDSHAKE SUCCEED !!**** retry=2
```

When the SSL connection is closed, the following message appears:

```
*May 15 11:57:42.541:OSP:osp_ssl_close:OSPPSSLClose succeed
```

The following are possible output trace messages:

```
osp_ssl_callback_add_session:session not found, add it.
osp_ssl_callback_add_session:session found, but not equal, delete old one
osp_ssl_callback_add_session:Copy new session data
osp_ssl_callback_add_session:session found and equal. no add
osp_ssl_callback_get_session:No Session exist
osp_ssl_callback_get_session:Session found, copy to sslref length=756
osp_ssl_callback_delete_session:session not found
```

These messages do not indicate an error but indicate the result of the operation.

To display actual error messages, enter the **debug voip settlement error** command.

debug voip settlement transaction

To see all the attributes of the transactions on the settlement gateway, use the **debug voip settlement transaction in privileged EXEC**mode. To disable debugging output, use the **no** form of this command.

debug voip settlement transaction

no debug voip settlement transaction

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(4)XH1	This command was introduced.

Examples

The following is sample output from the originating gateway:

```
00:44:54:OSP:OSPPTransactionNew:trans=0, err=0
00:44:54:OSP:osp_authorize:authorizing trans=0, err=0
router>
00:45:05:OSP:stop_settlement_ccapi_accounting:send report for
callid=7, trans
=0, calling=5710868, called=15125551212, curr_Dest=1
00:45:05:OSP:OSPPTransactionDelete:deleting trans=0
```

The following is sample output from the terminating gateway:

```
00:44:40:OSP:OSPPTransactionNew:trans=0, err=0
00:44:40:OSP:osp_validate:validated trans=0, error=0, authorised=1
```

debug voip tsp

To display information about the telephony service provider (TSP), use the **debug voip tsp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip tsp [**all**| **default**| **error** [**call [informational]**| **software [informational]**]| **event**| **function**| **individual** *range*| **inout**| **rose**]

no debug voip tsp

Syntax Description

all	(Optional) Displays all TSP debugging messages.
default	(Optional) Displays TSP inout, error, and event debugging messages. This option also runs if no keywords are added.
error	(Optional) Displays TSP error messages.
call	(Optional) Displays TSP call-related debugs not generated by other TSP debug options.
informational	(Optional) Displays minor errors and major errors. Without the **informational** keyword, only major errors are displayed.
software	(Optional) Displays software processing errors.
event	(Optional) Displays TSP events.
function	(Optional) Displays TSP functions.
individual	(Optional) Enables individual TSP debugs.
range	For the **individual** keyword, the range is an integer value from 1 to 68. For specific range values, see the table below.
inout	(Optional) Displays TSP function entry/exit debugs.
rose	(Optional) Enables the remote operations service element. This debug displays information about ISDN-related elements.

Table 6: TSP Individual Debug Values

Value	TSP Debug Function
1	INDIVIDUAL_TSP_DEBUG_TDM_HAIRPIN_CONNECT_001
2	INDIVIDUAL_TSP_DEBUG_TDM_HAIRPIN_DISCONNECT_002
3	INDIVIDUAL_TSP_DEBUG_CCRAWMSG_ENCAP_003
4	INDIVIDUAL_TSP_DEBUG_CDAPI_FORM_MSG_BASIC_SS_INFO_004
5	INDIVIDUAL_TSP_DEBUG_CDAPI_FORM_MSG_005
6	INDIVIDUAL_TSP_DEBUG_CDAPI_FORM_MSG_006
7	INDIVIDUAL_TSP_DEBUG_CDAPI_SEND_MSG_007
8	INDIVIDUAL_TSP_DEBUG_CDAPI_SEND_MSG_008
9	INDIVIDUAL_TSP_DEBUG_CDAPI_SEND_INFO_MSG_009
10	INDIVIDUAL_TSP_DEBUG_ALLOC_CDB_010
11	INDIVIDUAL_TSP_DEBUG_DEALLOC_CDB_011
12	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_012
13	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_013
14	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_014
15	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_015
16	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_016
17	INDIVIDUAL_TSP_DEBUG_CONNECT_IND_EXIT_017
18	INDIVIDUAL_TSP_DEBUG_CDAPI_SETUP_ACK_018
19	INDIVIDUAL_TSP_DEBUG_CDAPI_PROCEEDING_019
20	INDIVIDUAL_TSP_DEBUG_CDAPI_ALERT_020
21	INDIVIDUAL_TSP_DEBUG_CDAPI_CONNECT_021
22	INDIVIDUAL_TSP_DEBUG_CDAPI_INFO_022
23	INDIVIDUAL_TSP_DEBUG_CDAPI_PROGRESS_023
24	INDIVIDUAL_TSP_DEBUG_CDAPI_FACILITY_024

Value	TSP Debug Function
25	INDIVIDUAL_TSP_DEBUG_CDAPI_FACILITY_025
26	INDIVIDUAL_TSP_DEBUG_CDAPI_PRE_CONN_DISC_REQ_026
27	INDIVIDUAL_TSP_DEBUG_CDAPI_DISC_PROG_IND_027
28	INDIVIDUAL_TSP_DEBUG_CDAPI_DISCONNECT_REQ_028
29	INDIVIDUAL_TSP_DEBUG_CDAPI_DISCONNECT_REQ_029
30	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_SS_RESP_030
31	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_INFO_IND_031
32	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_PROCEEDING_032
33	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_ALERT_033
34	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_ALERT_EXIT_034
35	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_ALERT_EXIT_035
36	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_PROGRESS_036
37	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_INFO_037
38	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_CONNECT_038
39	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_CONNECT_CONF_039
40	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_DISC_PROG_IND_040
41	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_PROG_IND_PROGRESS_041
42	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_IND_042
43	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_IND_EXIT_043
44	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_COMP_044
45	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RELEASE_COMP_CLEAR_045
46	INDIVIDUAL_TSP_DEBUG_SETUP_REQ_EXIT_046
47	INDIVIDUAL_TSP_DEBUG_SETUP_REQ_EXIT_047
48	INDIVIDUAL_TSP_DEBUG_SETUP_REQ_EXIT_048

Value	TSP Debug Function
49	INDIVIDUAL_TSP_DEBUG_TSP_SET_TRANSFER_INFO_049
50	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_050
51	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_051
52	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_052
53	INDIVIDUAL_TSP_DEBUG_TSP_CALL_VOICE_CUT_THROUGH_053
54	INDIVIDUAL_TSP_DEBUG_TSP_MAIN_054
55	INDIVIDUAL_TSP_DEBUG_DO_GLOBAL_END_TO_END_DISC_055
56	INDIVIDUAL_TSP_DEBUG_TSP_CDAPI_MSG_DUMP_056
57	INDIVIDUAL_TSP_DEBUG_TSP_COT_TIMER_START_057
58	INDIVIDUAL_TSP_DEBUG_TSP_COT_TIMER_STOP_058
59	INDIVIDUAL_TSP_DEBUG_TSP_COT_RESULT_059
60	INDIVIDUAL_TSP_DEBUG_TSP_COT_DONE_060
61	INDIVIDUAL_TSP_DEBUG_TSP_COT_TIMEOUT_061
62	INDIVIDUAL_TSP_DEBUG_TSP_COT_REQ_062
63	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_COT_SETUP_ACK_063
64	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RCV_COT_MSG_064
65	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RCV_COT_MSG_065
66	INDIVIDUAL_TSP_DEBUG_TSP_CDAPI_PUT_CAUSE_IE_066
67	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_SETUP_ACK_067
68	INDIVIDUAL_TSP_DEBUG_CDAPI_TSP_RCV_MSG_068

Command Default Debugging is not enabled.

Command Modes Privileged EXEC

Command History

Release	Modification
12.3(8)T	This command replaces the **debug tsp** command.

Examples

The following is sample output from the **debug voip tsp**command:

```
Router# debug voip tsp

Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_voice_call_check:Query#9
   Overlap=FALSE, Called Number=222, Calling Number=4321
*Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_check_call_type:Query#9

   Peer Search Type=Voice
*Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_check_call_type:Query#9

   Matched Incoming Dialpeer With=Port, Peer=299
*Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_check_call_type:Query#9

   DID=TRUE
*Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_check_call_type:Query#9

   Incoming Dial-peer=SUCCESS
*Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_check_call_type:Query#9

   Matched Outgoing Dialpeer=221
*Apr  4 2002 14:04:11.034 UTC://-1/xxxxxxxxxxxx/TSP:():-1/FFFF/tsp_voice_call_check:Query#9

   Call Type=VOICE, Result=ACCEPT
```

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

Table 7: debug voip tsp dialpeer Field Descriptions

Field	Description
//-1/xxxxxxxxxxxx/TSP:():-1/FFFF/ tsp_voice_call_check:	The format of this message is //callid/GUID/DMSP/function name: • CallEntry ID is -1. This indicates that a call leg has not been identified. • GUID is xxxxxxxxxxxx. This indicates that the call has not been specified. • TSP:():-1/FFFFis the module name and module-specific parameters. • Thetsp_voice_call_checkfield shows that the accounting for an onramp fax is active.
Called Number=222, Calling Number=4321	Shows the calling and called numbers for the call.
Matched Incoming Dialpeer With=Port, Peer=299	Shows that the incoming dial peer was matched and identifies the dial peer.

Field	Description
DID=TRUE	Indicates that the call is a direct-inward dial (DID) call.
Matched Outgoing Dialpeer=221	Shows that the outgoing dial peer was matched and identifies the dial peer.

Related Commands

Command	Description
debug track	Displays information about the telephony service provider.
debug voip rawmsg	Displays the raw message owner, length, and pointer.

debug voip vtsp

To display information about the voice telephony service provider (VTSP), use the **debug voip vtsp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip vtsp [**all**| **default**| **error**| **event**| **function**| **individual** *range*| **inout**| **session**| **tone**]

no debug voip vtsp

Syntax Description

all	(Optional) Displays all VTSP debugging messages.
default	(Optional) Displays VTSP inout, error, and event debugging messages. This option also runs if no keywords are added.
error	(Optional) Displays VTSP error messages.
event	(Optional) Displays VTSP events.
function	(Optional) Displays VTSP functions.
individual	(Optional) Enables individual VTSP debugs.
range	For the **individual** keyword, the range is an integer value from 1 to 102. For specific range values, see the table below.
inout	(Optional) Displays VTSP function entry/exit debugs.
session	(Optional) Traces how the router interacts with the digital signal processor (DSP) based on the signaling indications from the signaling stack and requests from the application.
tone	(Optional) Displays the VTSP messages showing the types of tones generated by the Voice over IP (VoIP) gateway.

Table 8: VTSP Individual Debug Values

Value	VTSP Debug Function
1	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_PEND_DEFER_001
2	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_WAIT_PEND_SUCCESS_002
3	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_WAIT_PEND_FAIL_003

Value	VTSP Debug Function
4	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_COMPLETE_004
5	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_COMPLETE_EXIT_005
6	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_CHECK_006
7	INDIVIDUAL_VTSP_DEBUG_TDM_HPM_CHECK_EXIT_007
8	INDIVIDUAL_VTSP_DEBUG_GENERATE_DISC_008
9	INDIVIDUAL_VTSP_DEBUG_GENERATE_DISC_EXIT_009
10	INDIVIDUAL_VTSP_DEBUG_SETUP_IND_ACK_010
11	INDIVIDUAL_VTSP_DEBUG_SETUP_IND_ACK_EXIT_011
12	INDIVIDUAL_VTSP_DEBUG_PROCEEDING_012
13	INDIVIDUAL_VTSP_DEBUG_PRE_CON_DISCONNECT_013
14	INDIVIDUAL_VTSP_DEBUG_PRE_CON_DISCONNECT_EXIT_014
15	INDIVIDUAL_VTSP_DEBUG_SET_DIGIT_TIMEOUTS_015
16	INDIVIDUAL_VTSP_DEBUG_CONNECT_016
17	INDIVIDUAL_VTSP_DEBUG_LOOPBACK_017
18	INDIVIDUAL_VTSP_DEBUG_RING_NOAN_TIMER_018
19	INDIVIDUAL_VTSP_DEBUG_ALERT_CONNECT_019
20	INDIVIDUAL_VTSP_DEBUG_PRE_CON_DISC_REL_EXIT_020
21	INDIVIDUAL_VTSP_DEBUG_HOST_DISC_CLEANUP_021
22	INDIVIDUAL_VTSP_DEBUG_HOST_DISC_CLEANUP_EXIT_022
23	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_023
24	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_EXIT_024
25	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_EXIT_025
26	INDIVIDUAL_VTSP_DEBUG_CONNECT_DIAL_026
27	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_DIAL_027

Value	VTSP Debug Function
28	INDIVIDUAL_VTSP_DEBUG_PRE_DISC_CAUSE_028
29	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_CONNECT_029
30	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_PEND_FAIL_030
31	INDIVIDUAL_VTSP_DEBUG_SETUP_REQ_DISC_031
32	INDIVIDUAL_VTSP_DEBUG_RELEASE_TIMEOUT_032
33	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROCEEDING_EXIT_033
34	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROCEEDING_EXIT_034
35	INDIVIDUAL_VTSP_DEBUG_PEND_RELEASE_IND_035
36	INDIVIDUAL_VTSP_DEBUG_PEND_RELEASE_IND_EXIT_036
37	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_NO_DSP_CHAN_037
38	INDIVIDUAL_VTSP_DEBUG_DISCONNECT_NO_DSP_CHAN_EXIT_038
39	INDIVIDUAL_VTSP_DEBUG_CALL_FEATURE_IND_039
40	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROGRESS_040
41	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_041
42	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_EXIT_042
43	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_FIRST_PROGRESS_043
44	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_FIRST_PROGRESS_EXIT_044
45	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_FIRST_PROGRESS_EXIT_045
46	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_PROG_PROCEEDING_046
47	INDIVIDUAL_VTSP_DEBUG_PROCEEDING_R2_PEND_DIAL_047
48	INDIVIDUAL_VTSP_DEBUG_ALERT_R2_PEND_DIAL_048
49	INDIVIDUAL_VTSP_DEBUG_CONN_R2_PEND_DIAL_049
50	INDIVIDUAL_VTSP_DEBUG_SETUP_R2_PEND_DIAL_050
51	INDIVIDUAL_VTSP_DEBUG_R2_PEND_DIAL_ALL_051

Value	VTSP Debug Function
52	INDIVIDUAL_VTSP_DEBUG_INFO_IND_052
53	INDIVIDUAL_VTSP_DEBUG_ALERT_053
54	INDIVIDUAL_VTSP_DEBUG_ALERT_EXIT_054
55	INDIVIDUAL_VTSP_DEBUG_PROGRESS_055
56	INDIVIDUAL_VTSP_DEBUG_DISC_PROG_IND_056
57	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_DISC_PI_IND_057
58	INDIVIDUAL_VTSP_DEBUG_INFO_058
59	INDIVIDUAL_VTSP_DEBUG_FEATURE_059
60	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_NO_TIMEOUT_060
61	INDIVIDUAL_VTSP_DEBUG_SETUP_PEND_ALERT_NO_TIMEOUT_EXIT_061
62	INDIVIDUAL_VTSP_DEBUG_CALL_FEATURE_ENABLE_062
63	INDIVIDUAL_VTSP_DEBUG_XCCSM_COT_TEST_DONE_063
64	INDIVIDUAL_VTSP_DEBUG_XCCSM_COT_TEST_TIMEOUT_064
65	INDIVIDUAL_VTSP_DEBUG_XCCSM_COT_TEST_065
66	INDIVIDUAL_VTSP_DEBUG_CALL_FEATURE_066
67	INDIVIDUAL_VTSP_DEBUG_TCSM_COT_TEST_DONE_067
68	INDIVIDUAL_VTSP_DEBUG_TCSM_COT_TEST_TIMEOUT_068
69	INDIVIDUAL_VTSP_DEBUG_TCSM_ACT_COT_TEST_069
70	INDIVIDUAL_VTSP_DEBUG_PLAY_BUSY_TIMER_START_070
71	INDIVIDUAL_VTSP_DEBUG_PLAY_BUSY_TIMER_STOP_071
72	INDIVIDUAL_VTSP_DEBUG_RING_NOAN_TIMER_START_072
73	INDIVIDUAL_VTSP_DEBUG_RING_NOAN_TIMER_STOP_073
74	INDIVIDUAL_VTSP_DEBUG_VTSP_TIMER_074
75	INDIVIDUAL_VTSP_DEBUG_VTSP_TIMER_STOP_075

Value	VTSP Debug Function
76	INDIVIDUAL_VTSP_DEBUG_VTSP_ALLOCATE_CDB_076
77	INDIVIDUAL_VTSP_DEBUG_VTSP_DO_CALL_SETUP_IND_077
78	INDIVIDUAL_VTSP_DEBUG_VTSP_DO_CALL_SETUP_IND_EXIT_078
79	INDIVIDUAL_VTSP_DEBUG_VTSP_REQUEST_CALL_079
80	INDIVIDUAL_VTSP_DEBUG_VTSP_REQUEST_CALL_EXIT_080
81	INDIVIDUAL_VTSP_DEBUG_VTSP_REALLOC_CDB_081
82	INDIVIDUAL_VTSP_DEBUG_VTSP_OG_CALL_REQ_EXIT_082
83	INDIVIDUAL_VTSP_DEBUG_VTSP_FREE_CDB_083
84	INDIVIDUAL_VTSP_DEBUG_TGRM_DISC_REL_084
85	INDIVIDUAL_VTSP_DEBUG_VTSP_CC_CALL_DISCONNECTED_085
86	INDIVIDUAL_VTSP_DEBUG_SIGO_BDROP_086
87	INDIVIDUAL_VTSP_DEBUG_SIGO_PRE_CON_DISCONNECT_087
88	INDIVIDUAL_VTSP_DEBUG_SIGO_PROCEEDING_088
89	INDIVIDUAL_VTSP_DEBUG_SIGO_GENERATE_DISC_089
90	INDIVIDUAL_VTSP_DEBUG_SIGO_ALERT_090
91	INDIVIDUAL_VTSP_DEBUG_SIGO_ALERT_CONNECT_091
92	INDIVIDUAL_VTSP_DEBUG_SIGO_SETUP_PEND_CONNECT_092
93	INDIVIDUAL_VTSP_DEBUG_DO_SIGO_CALL_SETUP_REQ_093
94	INDIVIDUAL_VTSP_DEBUG_DO_SIGO_CALL_SETUP_REQ_SESSION_094
95	INDIVIDUAL_VTSP_DEBUG_DSM_MEDIA_EVENT_CB_095
96	INDIVIDUAL_VTSP_DEBUG_DSM_PEER_EVENT_CB_096
97	INDIVIDUAL_VTSP_DEBUG_DSM_FEATURE_NOTIFY_CB_097
98	INDIVIDUAL_VTSP_DEBUG_DSM_BRIDGE_CHECK_CB_098
99	INDIVIDUAL_VTSP_DEBUG_DSM_BRIDGE_STATUS_EXIT_099

Value	VTSP Debug Function
100	INDIVIDUAL_VTSP_DEBUG_DSM_SET_FAX_FEAT_EXIT_100
101	INDIVIDUAL_VTSP_DEBUG_DS_DO_DIAL_101
102	INDIVIDUAL_VTSP_DEBUG_DS_DIALING_DEFAULT_102

Command Default

Debugging is not enabled.

Command Modes

Privileged EXEC

Command History

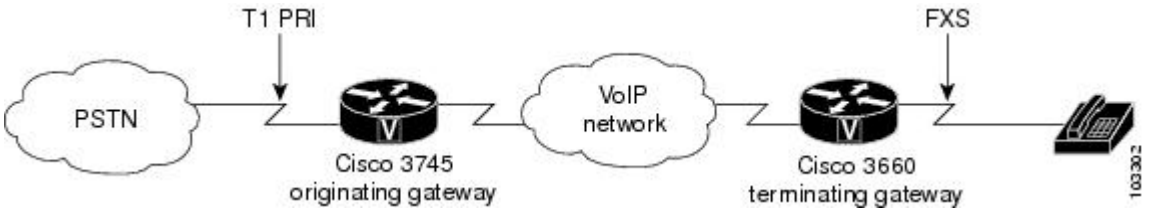
Release	Modification
12.3(8)T	This command replaces the **debug vtsp** command.
12.3(14)T	T.38 fax relay call statistics were made available to Call Detail Records (CDRs) through Vendor-Specific Attributes (VSAs) and added to the call log.

Examples

The following examples show output for variations of the **debug voip vtsp** command:

For these examples, the topology shown in the figure below is used.

Figure 5: Network Topology for debug voip vtsp Examples



Examples

```
Router# debug voip vtsp event
voip vtsp event debugging is on
*May  1 20:03:47.703: //-1/xxxxxxxxxxxx/VTSP:(4/0/0):-1:-1:-1/vtsp_process_event:
   [state:INVALID STATE MACHINE, event:E_CC_SETUP_REQ]
```

At the setup request, the CallEntry ID and GUID are set. The remainder of the output follows the progress of the call.

```
*May  1 20:03:47.707: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_SETUP_REQUEST, event:E_TSP_PROCEEDING]
*May  1 20:03:47.707: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_SETUP_REQ_PROC, event:E_TSP_PROGRESS]
*May  1 20:03:49.955: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
```

```
   [state:S_SETUP_REQ_PROC, event:E_TSP_CONNECT]
*May  1 20:03:49.959: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_CONNECT, event:E_CC_FEATURE]
*May  1 20:04:14.851: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_CONNECT, event:E_CC_DISCONNECT]
*May  1 20:04:14.855: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_WAIT_STATS, event:E_VTSP_DSM_STATS_COMPLETE]
*May  1 20:04:15.759: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_WAIT_RELEASE, event:E_TSP_CALL_FEATURE_IND]
*May  1 20:04:15.811: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_WAIT_RELEASE, event:E_TSP_CALL_FEATURE_IND]
*May  1 20:04:15.811: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:1:1/vtsp_process_event:
   [state:S_WAIT_RELEASE, event:E_TSP_DISCONNECT_CONF]
*May  1 20:04:15.811: //78/CDDFE7FF8029/VTSP:(4/0/0):-1:-1:-1/vtsp_process_event:
   [state:S_CLOSE_DSPRM, event:E_VTSP_DSM_CLOSE_COMPLETE]
```

Examples

```
Router# debug voip vtsp function

voip vtsp function debugging is on
*Apr 18 21:48:25.671: //-1/xxxxxxxxxxxx/VTSP:(2/1:23):-1:-1:-1/vtsp_do_call_setup_ind:
```

At the setup request, the CallEntry ID and GUID are set. The call setup functions are shown.

```
*Apr 18 21:48:25.671: //-1/D87794B9802B/VTSP:(2/1:23):0:-1:-1/vtsp_do_normal_call_setup_ind:
*Apr 18 21:48:25.671: //-1/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_indicate_call:
*Apr 18 21:48:25.675: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_insert_cdb:
*Apr 18 21:48:25.675: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_proceeding:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_progress:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_bridge_check_cb:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_bridge_check_cb:exit@1066
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_bridge_status_cb:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_save_fax_config:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_set_fax_feat_param:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_feature_notify_cb:
*Apr 18 21:48:25.687: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_reactivate_ringback:
*Apr 18 21:48:25.687:
//88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_reactivate_ringback:exit@871
*Apr 18 21:48:27.451: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
```

At this point, the ringback to the caller has occurred and the next event shows a connection.

```
*Apr 18 21:48:28.635: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_connect:
*Apr 18 21:48:29.003: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
*Apr 18 21:48:34.059: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
*Apr 18 21:48:36.587: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_generate_disc:
*Apr 18 21:48:36.587: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_cc_call_disconnected:
*Apr 18 21:48:36.587: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_bridge_status_cb:
```

The next event shows the call disconnect. There are several VTSP functions that follow the call disconnection to release and terminate the call.

```
*Apr 18 21:48:36.587: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_disconnect:
*Apr 18 21:48:36.587: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_save_fax_config:
*Apr 18 21:48:36.587: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_set_fax_feat_param:
*Apr 18 21:48:36.595: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_dsm_dsp_stats_complete:
*Apr 18 21:48:36.595: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/act_wrelease_release:
*Apr 18 21:48:36.595: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_do_call_history:
*Apr 18 21:48:36.595: //88/D87794B9802B/VTSP:(2/1:23):0:8:4/vtsp_dsm_closed_cb:
*Apr 18 21:48:36.595: //88/D87794B9802B/VTSP:(2/1:23):0:-1:-1/act_terminate:
*Apr 18 21:48:36.595: //-1/D87794B9802B/VTSP:(2/1:23):0:-1:-1/vtsp_free_cdb:
```

Examples

```
Router# debug voip vtsp inout

voip vtsp inout debugging is on
*Apr 18 21:48:59.239: //-1/xxxxxxxxxxxx/VTSP:(2/1:23):-1:-1:-1/vtsp_allocate_cdb:
```

```
   CDB=0x65289878
*Apr 18 21:48:59.239: //-1/xxxxxxxxxxxx/VTSP:(2/1:23):-1:-1:-1/vtsp_do_call_setup_ind:
   Event=E_TSP_SETUP_IND
   Progress Indication=0, CarrierIDCode=, Info Trans Capability=0, Source Carrier ID=,
tg_label_flag=0
```

The following two events show the calling number, called number, and related parameters:

```
*Apr 18 21:48:59.239: //-1/xxxxxxxxxxxx/VTSP:(2/1:23):-1:-1:-1/vtsp_do_call_setup_ind:
   Calling Number=4085550111, TON=National, NPI=ISDN, Screening=User, Passed,
Presentation=Allowed
   CLIR=FALSE, CLID Transparent=FALSE, Null Originating Calling Number=FALSE, Calling
Translated=FALSE
*Apr 18 21:48:59.239: //-1/xxxxxxxxxxxx/VTSP:(2/1:23):-1:-1:-1/vtsp_do_call_setup_ind:
   Called Number=83103, TON=Unknown, NPI=Unknown
*Apr 18 21:48:59.239: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_insert_cdb:
*Apr 18 21:48:59.243: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_proceeding:
   Progress Indication=0
*Apr 18 21:48:59.255: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_progress:
   Progress Indication=8
*Apr 18 21:48:59.255: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_timer_stop:
   Timer Stop Time=538706
```

The following event shows fax parameters associated with the call:

```
*Apr 18 21:48:59.255: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_dsm_save_fax_config:
   Fax Relay=DISABLED - 'fax rate disabled' set (dial-peer)
   Primary Fax Protocol=IGNORE_FAX_RELAY, Fallback Fax Protocol=IGNORE_FAX_RELAY
   Fax Parameters Set By=Dialpeer, Peer=3600
*Apr 18 21:48:59.255: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_CAPS_IND
*Apr 18 21:48:59.255: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_CAPS_ACK
*Apr 18 21:48:59.255: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_dsm_feature_notify_cb:
   Feature ID=0, Feature Status=1
```

The following event shows the call connection:

```
*Apr 18 21:49:03.779: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_connect:
   Progress Indication=2
*Apr 18 21:49:03.779: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_ring_noan_timer_stop:
   Timer Stop Time=539158
Router#
```

The following event shows the call disconnect:

```
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_generate_disc:
   Cause Value=16
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_ring_noan_timer_stop:
   Timer Stop Time=541374
```

The following event shows that it was the calling party that initiated the call disconnect:

```
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_set_release_source:
   Release Direction=PSTN, Release Source=Calling Party-PSTN
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_timer:
   Timer Start Time=541374, Timer Value=15000(ms)
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_generate_disc:
   Return Code=0
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_disconnect:
   Cause Value=16, Previous Cause Value=16
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_ring_noan_timer_stop:
   Timer Stop Time=541374
*Apr 18 21:49:25.943: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_dsm_save_fax_config:
   Fax Relay=DISABLED - 'fax rate disabled' set (dial-peer)
   Primary Fax Protocol=IGNORE_FAX_RELAY, Fallback Fax Protocol=IGNORE_FAX_RELAY
   Fax Parameters Set By=Dialpeer, Peer=3600
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_dsm_dsp_stats_complete:
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_timer:
   Timer Start Time=541375, Timer Value=60000(ms)
```

The following two events show the call being released and the timer stopping:

```
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/act_wrelease_release:
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_play_busy_timer_stop:
   Timer Stop Time=541375
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:8:4/vtsp_timer_stop:
   Timer Stop Time=541375
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:-1:-1/act_terminate:
*Apr 18 21:49:25.951: //90/EC79A754802C/VTSP:(2/1:23):0:-1:-1/vtsp_timer_stop:
   Timer Stop Time=541375
*Apr 18 21:49:25.951: //-1/EC79A754802C/VTSP:(2/1:23):0:-1:-1/vtsp_free_cdb:
   CDB=0x65289878
Router#
```

Examples

```
Router# debug voip vtsp tone
voip vtsp tone debugging is on
```

This output shows a wrong number dialed in the two-stage call to see the tone generated by the router.

```
*Apr 18 21:52:26.595: //98/657C0B9C8030/VTSP:(2/1:23):0:8:4/act_pre_con_disconnect:
   [Number Unobtainable]-Tone Played In Direction [Network]
```

Examples

This output shows the fax relay statistics.

```
Router# debug voip vtsp
VTSP:
  debug voip vtsp event is ON (filter is OFF)
  debug voip vtsp error software is ON
  debug voip vtsp error call is ON (filter is OFF)
  debug voip vtsp inout is ON (filter is OFF)
May  7 21:37:35.322 UTC: //-1/xxxxxxxxxxxx/VTSP:(3/1:D):-1:-1:-1/vtsp_allocate_cdb:
   CDB=0x63088050
May  7 21:37:35.322 UTC: //-1/xxxxxxxxxxxx/VTSP:(3/1:D):-1:-1:-1/vtsp_do_call_setup_ind:
   Event=E_TSP_SETUP_IND
   Progress Indication=3, CarrierIDCode=, Info Trans Capability=16, Source Carrier ID=,
tg_label_flag=0
May  7 21:37:35.322 UTC: //-1/xxxxxxxxxxxx/VTSP:(3/1:D):-1:-1:-1/vtsp_do_call_setup_ind:
   Called Number=41021, TON=National, NPI=ISDN
May  7 21:37:35.326 UTC: //-1/96A4C0C48006/VTSP:(3/1:D):0:-1:-1/vtsp_timer:
   Timer Start Time=1019501, Timer Value=180000(ms)
May  7 21:37:35.326 UTC: //-1/96A4C0C48006/VTSP:(3/1:D):0:-1:-1/vtsp_insert_cdb:
May  7 21:37:35.326 UTC: //-1/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_SETUP_IND_PEND, event:E_VTSP_DSM_OPEN_SUCCESS]
May  7 21:37:35.326 UTC: //-1/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_setup_ind_pend_success:
May  7 21:37:35.326 UTC: //-1/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_realloc_cdb:
   CDB=0x63088050
May  7 21:37:35.326 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_insert_cdb:
May  7 21:37:35.326 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer:
   Timer Start Time=1019501, Timer Value=180000(ms)
May  7 21:37:35.330 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_SETUP_INDICATED, event:E_CC_PROCEEDING]
May  7 21:37:35.330 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_proceeding:
   Progress Indication=0
May  7 21:37:35.330 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer_stop:
   Timer Stop Time=1019502
May  7 21:37:35.394 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_PROCEEDING, event:E_CC_ALERT]
May  7 21:37:35.394 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_alert:
   Progress Indication=0, Signal Indication=1, Setup Progress Indication=3
May  7 21:37:35.394 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer_stop:
   Timer Stop Time=1019508
May  7 21:37:35.398 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_alert:
   Progress Indication=0, Tone=
May  7 21:37:37.422 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_save_fax_config:
   Fax Relay=ENABLED
   Primary Fax Protocol=T38_FAX_RELAY, Fallback Fax Protocol=NONE_FAX_RELAY
```

```
   Fax Parameters Set By=Dialpeer, Peer=2
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_save_fax_config:
   LS Red=0, HS Red=0
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_ALERTING, event:E_CC_DO_CAPS_IND]
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_CAPS_IND
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_ALERTING, event:E_CC_CAPS_IND]
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_CAPS_ACK
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_ALERTING, event:E_CC_SERVICE_MSG]
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_service_msg_down:
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer_stop:
   Timer Stop Time=1019711
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_ALERTING, event:E_CC_CONNECT]
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_alert_connect:
   Progress Indication=0
May  7 21:37:37.426 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_ring_noan_timer_stop:
   Timer Stop Time=1019711
May  7 21:37:37.598 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_feature_notify_cb:
   Feature ID=0, Feature Status=1
May  7 21:37:37.598 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_feature_notify_cb:
   Feature ID=0, Feature Status=1
May  7 21:37:44.123 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_MC_LOCAL_DNLD_DONE
May  7 21:37:44.123 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_SET_FAX_MODE
May  7 21:37:44.123 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_MC_LOCAL_DNLD_DONE
May  7 21:37:44.123 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_SET_FAX_MODE
May  7 21:38:02.911 UTC: %ALIGN-3-SPURIOUS: Spurious memory access made at 0x6040A40C reading
 0x1
May  7 21:38:02.911 UTC: %ALIGN-3-TRACE: -Traceback= 6040A40C 60409198 603F8338 603F85F8
613EA398 619B369C 619B40BC 613DFEE4
May  7 21:38:02.915 UTC: %ALIGN-3-TRACE: -Traceback= 6040A54C 60409198 603F8338 603F85F8
613EA398 619B369C 619B40BC 613DFEE4
May  7 21:38:37.483 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_CONNECT, event:E_CC_CAPS_IND]
May  7 21:38:37.483 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_peer_event_cb:
   Event=E_DSM_CC_CAPS_ACK
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_CONNECT, event:E_TSP_DISCONNECT_IND]
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_generate_disc:
   Cause Value=16
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer_stop:
   Timer Stop Time=1025735
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_ring_noan_timer_stop:
   Timer Stop Time=1025735
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_set_release_source:
   Release Direction=PSTN, Release Source=Calling Party-PSTN
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer:
   Timer Start Time=1025735, Timer Value=15000(ms)
May  7 21:38:37.663 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_generate_disc:
   Return Code=0
May  7 21:38:37.667 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_CONNECT, event:E_CC_DISCONNECT]
May  7 21:38:37.667 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_disconnect:
   Cause Value=16, Previous Cause Value=16
May  7 21:38:37.667 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_ring_noan_timer_stop:
   Timer Stop Time=1025735
May  7 21:38:37.667 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_save_fax_config:
   Fax Relay=ENABLED
   Primary Fax Protocol=T38_FAX_RELAY, Fallback Fax Protocol=NONE_FAX_RELAY
   Fax Parameters Set By=Dialpeer, Peer=2
May  7 21:38:37.667 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_dsm_save_fax_config:
   LS Red=0, HS Red=0
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_WAIT_STATS, event:E_VTSP_DSM_STATS_COMPLETE]
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_dsm_dsp_stats_complete:
```

```
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer:
   Timer Start Time=1025738, Timer Value=60000(ms)
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_WAIT_RELEASE, event:E_TSP_DISCONNECT_CONF]
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/act_wrelease_release:
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_play_busy_timer_stop:
   Timer Stop Time=1025738
May  7 21:38:37.691 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_timer_stop:
   Timer Stop Time=1025738
May  7 21:38:37.695 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   [state:S_CLOSE_DSPRM, event:E_VTSP_DSM_STATS_COMPLETE]
May  7 21:38:37.695 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:0:0/vtsp_process_event:
   Unexpected EVENT [E_VTSP_DSM_STATS_COMPLETE] Received For STATE [S_CLOSE_DSPRM];
   Previous STATE [0.17 ]
May  7 21:38:37.695 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:-1:-1/vtsp_process_event:
   [state:S_CLOSE_DSPRM, event:E_VTSP_DSM_CLOSE_COMPLETE]
May  7 21:38:37.695 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:-1:-1/act_terminate:
May  7 21:38:37.695 UTC: //9/96A4C0C48006/VTSP:(3/1:D):0:-1:-1/vtsp_timer_stop:
   Timer Stop Time=1025738
May  7 21:38:37.695 UTC: //-1/96A4C0C48006/VTSP:(3/1:D):0:-1:-1/vtsp_free_cdb:
   CDB=0x63088050
```

Related Commands

Command	Description
debug voip dsm	Displays information about the DSM.
show debug	Displays which debug commands are enabled.
voice call debug	Allows configuration of the voice call debug output.

debug voip xcodemsp

To display debugging information from the Transcoding Media Service Processor and its related applications, use the **debug voip xcodemsp** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug voip xcodemsp

no debug voip xcodemsp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
12.3(8)T	This command was introduced.
12.2SX	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.

Examples

The following is sample output from the **debug voip xcodemsp** command:

```
Router# debug voip xcodemsp

XCODEMSP debugging is on
Router#
Router#
*Jul  8 18:36:53: xcmsp_call_setup_request:
*Jul  8 18:36:53: xcmsp_call_setup_request: callID 20, rscID 1 srvcDef.srvc_id 2
*Jul  8 18:36:53: xcmsp_bridge:
*Jul  8 18:36:53: xc_add_streams:
*Jul  8 18:36:53: xc_add_streams: stream id 1 added
*Jul  8 18:36:53: xc_add_streams: stream id 2 added
*Jul  8 18:36:53: xcmsp_bridge:
*Jul  8 18:36:53: xc_add_streams:
*Jul  8 18:36:53: xc_add_streams: stream id 5 added
*Jul  8 18:36:53: xc_add_streams: stream id 6 added
*Jul  8 18:36:53: xc_connect_bridges:
*Jul  8 18:36:53: xcmsp_dsmp_response
Router#
Router#
Router# show sccp connections

sess_id    conn_id    stype mode     codec   ripaddr         rport sport
16777223   16777905   xcode sendrecv g711a   1.4.177.1       16628 18870
16777223   16777921   xcode sendrecv g729ab  1.4.177.5       31318 18146
Total number of active session(s) 1, and connection(s) 2
Router#
*Jul  8 18:37:22: xcmsp_bridge_drop:
*Jul  8 18:37:22: xcmsp_bridge_drop: deleting stream id 5
*Jul  8 18:37:22: xcmsp_bridge_drop: deleting stream id 6
```

```
*Jul  8 18:37:22: xcmsp_dsmp_response
*Jul  8 18:37:22: xcmsp_dsmp_response: DSMP_DISCONNECTED
*Jul  8 18:37:22: xcmsp_bridge_drop:
*Jul  8 18:37:22: xcmsp_bridge_drop: deleting stream id 1
*Jul  8 18:37:22: xcmsp_bridge_drop: deleting stream id 2
*Jul  8 18:37:22: xcmsp_call_disconnect:
```

debug vpdn

To troubleshoot Layer 2 Forwarding (L2F) or Layer 2 Tunnel Protocol (L2TP) virtual private dial-up network (VPDN) tunneling events and infrastructure, use the **debug vpdn** command in privileged EXEC mode. To disable the debugging of L2TP VPDN tunneling events and infrastructure, use the **no** form of this command.

Note Effective with Cisco IOS Release 12.4(11)T, the L2F protocol is not supported in Cisco IOS software.

Cisco IOS Release 12.2(33)XNA and Later Releases

debug vpdn {**call** {**event**| **fsm**}| **authorization** {**error**| **event**}| **error**| **event** [**disconnect** [**traceback**]]| **l2tp-sequencing**| **l2x-data**| **l2x-errors**| **l2x-events**| **l2x-packets**| **message**| **packet** [**detail**| **errors**]| **sss** {**error**| **event**| **fsm**}| **subscriber** {**error**| **event**| **fsm**}}

no debug vpdn {**call** {**event**| **fsm**}| **authorization** {**error**| **event**}| **error**| **event** [**disconnect** [**traceback**]]| **l2tp-sequencing**| **l2x-data**| **l2x-errors**| **l2x-events**| **l2x-packets**| **message**| **packet** [**detail**| **errors**]| **sss** {**error**| **event**| **fsm**}| **subscriber** {**error**| **event**| **fsm**}}

Cisco IOS Releases Prior to 12.2(33)XNA

debug vpdn {**call** {**event**| **fsm**}| **authorization** {**error**| **event**}| **error**| **event** [**disconnect**]| **l2tp-sequencing**| **l2x-data**| **l2x-errors**| **l2x-events**| **l2x-packets**| **message**| **packet** [**detail**| **errors**]| **sss** {**error**| **event**| **fsm**}| **subscriber** {**error**| **event**| **fsm**}}

no debug vpdn {**call** {**event**| **fsm**}| **authorization** {**error**| **event**}| **error**| **event** [**disconnect**]| **l2tp-sequencing**| **l2x-data**| **l2x-errors**| **l2x-events**| **l2x-packets**| **message**| **packet** [**detail**| **errors**]| **sss** {**error**| **event**| **fsm**}| **subscriber** {**error**| **event**| **fsm**}}

Syntax Description

call event	Displays significant events in the VPDN call manager.
call fsm	Displays significant events in the VPDN call manager finite state machine (FSM).
authorization error	Displays authorization errors.
authorization event	Displays authorization events.
error	Displays VPDN errors.
event	Displays VPDN events.
disconnect	(Optional) Displays VPDN disconnect events. **Note** The disconnect keyword is required in Cisco IOS Release 12.2(33)XNA and later releases.
traceback	(Optional) Displays traceback messages that provide reasons for VPDN disconnect.

l2tp-sequencing	Displays significant events related to L2TP sequence numbers such as mismatches, resend queue flushes, and drops.
l2x-data	Displays errors that occur in data packets.
l2x-errors	Displays errors that occur in protocol-specific conditions.
l2x-events	Displays events resulting from protocol-specific conditions.
l2x-packets	Displays detailed information about control packets in protocol-specific conditions.
message	Displays VPDN interprocess messages.
packet	Displays information about VPDN packets.
detail	(Optional) Displays detailed packet information, including packet dumps.
errors	(Optional) Displays errors that occur in packet processing.
sss error	Displays debug information about VPDN Subscriber Service Switch (SSS) errors.
sss event	Displays debug information about VPDN SSS events.
sss fsm	Displays debug information about the VPDN SSS FSM.
subscriber error	Displays debug information about VPDN Subscriber errors.
subscriber event	Displays debug information about VPDN Subscriber events.
subscriber fsm	Displays debug information about the VPDN Subscriber FSM.

Command Modes

Privileged EXEC (#)

Command History

Release	Modification
11.2 T	This command was introduced.

Release	Modification
12.0(5)T	This command was modified. Support was added for L2TP debugging messages. The **l2tp-sequencing** and **error** keywords were added. The **l2f-errors**, **l2f-events**, and **l2f-packets** keywords were changed to **l2x-errors**, **l2x-events**, and **l2x-packets**.
12.2(4)T	This command was modified. The **call**, **event**, **fsm**, and **message** keywords were added.
12.2(11)T	This command was modified. The **detail** keyword was added.
12.0(23)S	This command was integrated into Cisco IOS Release 12.0(23)S.
12.2(13)T	This command was modified. The **sss**, **error**, **event**, and **fsm** keywords were added.
12.3(14)T	This command was modified. Support was added to decode the outbound control channel authentication events.
12.0(31)S	This command was modified. The output was enhanced to display messages about control channel authentication events.
12.2(27)SBC	This command was modified. Support for enhanced display of messages about control channel authentication events was added.
12.2(28)SB	This command was modified. Support for the display of messages about congestion avoidance events was added.
12.2(31)SB	This command was modified. Support was added to decode the outbound control channel authentication events.
12.4(15)T	This command was modified. The **authorization**, **error**, and **event** keywords were added.
12.2(33)XNA	This command was modified. The **traceback** keyword was added.
12.4(20)T	This command was modified. The **subscriber** keyword was added and the **sss** keyword was removed.
Cisco IOS XE Release 2.6	This command was modified. Authentication failure messages for L2TPv3 were added.

Usage Guidelines

The **debug vpdn packet** and **debug vpdn packet detail** commands generate several debug operations per packet. Depending on the L2TP traffic pattern, these commands may cause the CPU load to increase to a high level that impacts performance.

Examples

Examples

The following example shows the VPDN configuration on a network access server (NAS):

```
vpdn-group 1
 request-dialin
  protocol l2f
  domain example.com
 initiate-to ip 172.17.33.125
username nas1 password nas1
```

The following is sample output from the **debug vpdn event** command on a NAS when an L2F tunnel is brought up and Challenge Handshake Authentication Protocol (CHAP) authentication of the tunnel succeeds:

```
Device# debug vpdn event

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:26:05.537: looking for tunnel — example.com —
*Mar 2 00:26:05.545: Async6 VPN Forwarding...
*Mar 2 00:26:05.545: Async6 VPN Bind interface direction=1
*Mar 2 00:26:05.553: Async6 VPN vpn_forward_user user6@example.com is forwarded
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:26:06.289: L2F: Chap authentication succeeded for nas1.
```

The following is sample output from the **debug vpdn event** command on a NAS when the L2F tunnel is brought down normally:

```
Device# debug vpdn event

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:27:18.865: Async6 VPN cleanup
*Mar 2 00:27:18.869: Async6 VPN reset
*Mar 2 00:27:18.873: Async6 VPN Unbind interface
%LINK-3-UPDOWN: Interface Async6, changed state to down
```

The table below describes the significant fields shown in the two previous displays. The output describes normal operations when an L2F tunnel is brought up or down on a NAS.

Table 9: debug vpdn event Field Descriptions for the NAS

Field	Description
Asynchronous interface coming up	
%LINK-3-UPDOWN: Interface Async6, changed state to up	Asynchronous interface 6 came up.
looking for tunnel — example.com — Async6 VPN Forwarding...	Domain name is identified.

Field	Description
Async6 VPN Bind interface direction=1	Tunnel is bound to the interface. These are the direction values: • 1—From the NAS to the tunnel server • 2—From the tunnel server to the NAS
Async6 VPN vpn_forward_user user6@example.com is forwarded	Tunnel for the specified user and domain name is forwarded.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol is up.
L2F: Chap authentication succeeded for nas1.	Tunnel was authenticated with the tunnel password nas1.
Virtual access interface coming down	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down	Normal operation when the virtual access interface is taken down.
Async6 VPN cleanup Async6 VPN reset Async6 VPN Unbind interface	Normal cleanup operations performed when the line or virtual access interface goes down.

Examples

The following example shows the VPDN configuration on a tunnel server, which uses *nas1* as the tunnel name and the tunnel authentication name. The tunnel authentication name can be entered in a user's file on an authentication, authorization, and accounting (AAA) server and used to define authentication requirements for the tunnel.

```
vpdn-group 1
 accept-dialin
  protocol l2f
  virtual-template 1
 terminate-from hostname nas1
```

The following is sample output from the **debug vpdn event** command on a tunnel server when an L2F tunnel is brought up successfully:

```
Device# debug vpdn event

L2F: Chap authentication succeeded for nas1.
Virtual-Access3 VPN Virtual interface created for user6@example.com
Virtual-Access3 VPN Set to Async interface
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up
Virtual-Access3 VPN Bind interface direction=2
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up
```

The following is sample output from the **debug vpdn event** command on a tunnel server when an L2F tunnel is brought down normally:

```
Device# debug vpdn event

%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down
Virtual-Access3 VPN cleanup
Virtual-Access3 VPN reset
Virtual-Access3 VPN Unbind interface
Virtual-Access3 VPN reset
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down
```

The table below describes the fields shown in two previous outputs. The output describes normal operations when an L2F tunnel is brought up or down on a tunnel server.

Table 10: debug vpdn event Field Descriptions

Field	Description
L2F: Chap authentication succeeded for nas1.	PPP CHAP authentication status for the tunnel named *nas1*.
Virtual-Access3 VPN Virtual interface created for user6@example.com	Virtual access interface was set up on a tunnel server for the user user6@example.com.
Virtual-Access3 VPN Set to Async interface	Virtual access interface 3 was set to asynchronous for character-by-character transmission.
Virtual-Access3 VPN Clone from Vtemplate 1 block=1 filterPPP=0	Virtual template 1 was applied to virtual access interface 3.
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to up	Link status is set to up.
Virtual-Access3 VPN Bind interface direction=2	Tunnel is bound to the interface. These are the direction values: • 1—From the NAS to the tunnel server • 2—From the tunnel server to the NAS
Virtual-Access3 VPN PPP LCP accepted sent & rcv CONFACK	PPP link control protocol (LCP) configuration settings (negotiated between the remote client and the NAS) were copied to the tunnel server and acknowledged.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to up	Line protocol is up; the line can be used.
%LINK-3-UPDOWN: Interface Virtual-Access3, changed state to down	Virtual access interface is coming down.

Field	Description
Virtual-Access3 VPN cleanup Virtual-Access3 VPN reset Virtual-Access3 VPN Unbind interface Virtual-Access3 VPN reset	Device is performing normal cleanup operations when a virtual access interface used for an L2F tunnel comes down.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access3, changed state to down	Line protocol is down for virtual access interface 3; the line cannot be used.

Examples

The following is sample output from the **debug vpdn event disconnect traceback** command on a tunnel server when an L2TP Network Server (LNS) tunnel session is disconnected:

```
Device# debug vpdn event disconnect traceback

*Aug  8 07:13:56.795: VPDN Vi2.1 disconnect (L2X) IETF: 18/host-request Ascend: 66/VPDN
Local PPP Disconnect
*Aug  8 07:13:56.795: VPDN Vi2.1 vpdn shutdown session, result=2, error=6, vendor_err=0,
syslog_error_code=2, syslog_key_type=1
*Aug  8 07:13:56.795: VPDN Vi2.1 VPDN/AAA: accounting stop sent
*Aug  8 07:13:56.795: VPDN Vi2.1 Unbinding session from idb, informational traceback:
*Aug  8 07:13:56.795: -Traceback= DFFFE7z 30EE221z 30DFBA8z 30E2F26z 30DF1DCz 30DF12Fz
1F0170Fz 1F015A1z 31E695Bz 31E674Dz 1F019F6z
*Aug  8 07:13:56.795: Vi2.1 VPDN: Resetting interface, informational traceback below:
LNS#
*Aug  8 07:13:56.795: -Traceback= DFFFE7z 30EDE74z 30EE2D4z 37996B7z 37A3019z 30EE408z
30DFBB3z 30E2F26z 30DF1DCz 30DF12Fz 1F0170Fz 1F015A1z 31E695Bz 31E674Dz 1F019F6z
```

Examples

The following is sample output from the **debug vpdn event** command on the NAS when an L2TP tunnel is brought up successfully:

```
Device# debug vpdn event

20:19:17: L2TP: I SCCRQ from ts1 tnl 8
20:19:17: L2X: Never heard of ts1
20:19:17: Tnl 7 L2TP: New tunnel created for remote ts1, address 172.21.9.4
20:19:17: Tnl 7 L2TP: Got a challenge in SCCRQ, ts1
20:19:17: Tnl 7 L2TP: Tunnel state change from idle to wait-ctl-reply
20:19:17: Tnl 7 L2TP: Got a Challenge Response in SCCCN from ts1
20:19:17: Tnl 7 L2TP: Tunnel Authentication success
20:19:17: Tnl 7 L2TP: Tunnel state change from wait-ctl-reply to established
20:19:17: Tnl 7 L2TP: SM State established
20:19:17: Tnl/Cl 7/1 L2TP: Session FS enabled
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from idle to wait-for-tunnel
20:19:17: Tnl/Cl 7/1 L2TP: New session created
20:19:17: Tnl/Cl 7/1 L2TP: O ICRP to ts1 8/1
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from wait-for-tunnel to wait-connect
20:19:17: Tnl/Cl 7/1 L2TP: Session state change from wait-connect to established
20:19:17: Vi1 VPDN: Virtual interface created for example1@example.com
20:19:17: Vi1 VPDN: Set to Async interface
20:19:17: Vi1 VPDN: Clone from Vtemplate 1 filterPPP=0 blocking
20:19:18: %LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
20:19:18: Vi1 VPDN: Bind interface direction=2
20:19:18: Vi1 VPDN: PPP LCP accepting rcv CONFACK
20:19:19: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
```

Examples

The following is sample output from the **debug vpdn event** command on a tunnel server when an L2TP tunnel is brought up successfully:

```
Device# debug vpdn event

20:47:33: %LINK-3-UPDOWN: Interface Async7, changed state to up
20:47:35: As7 VPDN: Looking for tunnel — example.com —
20:47:35: As7 VPDN: Get tunnel info for example.com with NAS nas1, IP 172.21.9.13
20:47:35: As7 VPDN: Forward to address 172.21.9.13
20:47:35: As7 VPDN: Forwarding...
20:47:35: As7 VPDN: Bind interface direction=1
20:47:35: Tnl/Cl 8/1 L2TP: Session FS enabled
20:47:35: Tnl/Cl 8/1 L2TP: Session state change from idle to wait-for-tunnel
20:47:35: As7 8/1 L2TP: Create session
20:47:35: Tnl 8 L2TP: SM State idle
20:47:35: Tnl 8 L2TP: Tunnel state change from idle to wait-ctl-reply
20:47:35: Tnl 8 L2TP: SM State wait-ctl-reply
20:47:35: As7 VPDN: example1@example.com is forwarded
20:47:35: Tnl 8 L2TP: Got a challenge from remote peer, nas1
20:47:35: Tnl 8 L2TP: Got a response from remote peer, nas1
20:47:35: Tnl 8 L2TP: Tunnel Authentication success
20:47:35: Tnl 8 L2TP: Tunnel state change from wait-ctl-reply to established
20:47:35: Tnl 8 L2TP: SM State established
20:47:35: As7 8/1 L2TP: Session state change from wait-for-tunnel to wait-reply
20:47:35: As7 8/1 L2TP: Session state change from wait-reply to established
20:47:36: %LINEPROTO-5-UPDOWN: Line protocol on Interface Async7, changed state to up
```

Examples

The following is sample output from the **debug vpdn l2x-events** command on the NAS when an L2F tunnel is brought up successfully:

```
Device# debug vpdn l2x-events

%LINK-3-UPDOWN: Interface Async6, changed state to up
*Mar 2 00:41:17.365: L2F Open UDP socket to 172.21.9.26
*Mar 2 00:41:17.385: L2F_CONF received
*Mar 2 00:41:17.389: L2F Removing resend packet (type 1)
*Mar 2 00:41:17.477: L2F_OPEN received
*Mar 2 00:41:17.489: L2F Removing resend packet (type 2)
*Mar 2 00:41:17.493: L2F building nas2gw_mid0
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up
*Mar 2 00:41:18.613: L2F_OPEN received
*Mar 2 00:41:18.625: L2F Got a MID management packet
*Mar 2 00:41:18.625: L2F Removing resend packet (type 2)
*Mar 2 00:41:18.629: L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6
```

The following is sample output from the **debug vpdn l2x-events** command on a NAS when an L2F tunnel is brought down normally:

```
Device# debug vpdn l2x-events

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down
%LINK-5-CHANGED: Interface Async6, changed state to reset
*Mar 2 00:42:29.213: L2F_CLOSE received
*Mar 2 00:42:29.217: L2F Destroying mid
*Mar 2 00:42:29.217: L2F Removing resend packet (type 3)
*Mar 2 00:42:29.221: L2F Tunnel is going down!
*Mar 2 00:42:29.221: L2F Initiating tunnel shutdown.
*Mar 2 00:42:29.225: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F_CLOSE received
*Mar 2 00:42:29.229: L2F Got closing for tunnel
*Mar 2 00:42:29.233: L2F Removing resend packet
*Mar 2 00:42:29.233: L2F Closed tunnel structure
%LINK-3-UPDOWN: Interface Async6, changed state to down
*Mar 2 00:42:31.793: L2F Closed tunnel structure
*Mar 2 00:42:31.793: L2F Deleted inactive tunnel
```

The table below describes the fields shown in the displays.

Table 11: debug vpdn l2x-events Field Descriptions—NAS

Field	Descriptions
%LINK-3-UPDOWN: Interface Async6, changed state to up	Asynchronous interface came up normally.
L2F Open UDP socket to 172.21.9.26	L2F opened a User Datagram Protocol (UDP) socket to the tunnel server IP address.
L2F_CONF received	L2F_CONF signal was received. When sent from the tunnel server to the NAS, an L2F_CONF indicates the tunnel server's recognition of the tunnel creation request.
L2F Removing resend packet (type ...)	Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the tunnel server accepted the NAS configuration of an L2F tunnel.
L2F building nas2gw_mid0	L2F is building a tunnel between the NAS and the tunnel server using the multiplex ID (MID) MID0.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to up	Line protocol came up. Indicates whether the software processes that handle the line protocol regard the interface as usable.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the tunnel server accepted the NAS configuration of an L2F tunnel.
L2F Got a MID management packet	MID management packets are used to communicate between the NAS and the tunnel server.
L2F MID synced NAS/HG Clid=7/15 Mid=1 on Async6	L2F synchronized the client IDs on the NAS and the tunnel server, respectively. An MID is assigned to identify this connection in the tunnel.
Tunnel coming down	
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async6, changed state to down	Line protocol came down. Indicates whether the software processes that handle the line protocol regard the interface as usable.

Field	Descriptions
%LINK-5-CHANGED: Interface Async6, changed state to reset	Interface was marked as reset.
L2F_CLOSE received	NAS received a request to close the tunnel.
L2F Destroying mid	Connection identified by the MID is being taken down.
L2F Tunnel is going down!	Advisory message about impending tunnel shutdown.
L2F Initiating tunnel shutdown.	Tunnel shutdown has started.
L2F_CLOSE received	NAS received a request to close the tunnel.
L2F Got closing for tunnel	NAS began tunnel closing operations.
%LINK-3-UPDOWN: Interface Async6, changed state to down	Asynchronous interface was taken down.
L2F Closed tunnel structure	NAS closed the tunnel.
L2F Deleted inactive tunnel	Now-inactivated tunnel was deleted.

Examples

The following is sample output from the **debug vpdn l2x-events** command on a tunnel server when an L2F tunnel is created:

```
Device# debug vpdn l2x-events

L2F_CONF received
L2F Creating new tunnel for nas1
L2F Got a tunnel named nas1, responding
L2F Open UDP socket to 172.21.9.25
L2F_OPEN received
L2F Removing resend packet (type 1)
L2F_OPEN received
L2F Got a MID management packet
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up
```

The following is sample output from the **debug vpdn l2x-events** command on a tunnel server when the L2F tunnel is brought down normally:

```
Device# debug vpdn l2x-events

L2F_CLOSE received
L2F Destroying mid
L2F Removing resend packet (type 3)
L2F Tunnel is going down!
L2F Initiating tunnel shutdown.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down
L2F_CLOSE received
L2F Got closing for tunnel
L2F Removing resend packet
L2F Removing resend packet
L2F Closed tunnel structure
```

```
L2F Closed tunnel structure
L2F Deleted inactive tunnel
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down
```

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

Table 12: debug vpdn l2x-events Field Descriptions—Tunnel Server

Field	Description
L2F_CONF received	L2F configuration is received from the NAS. When sent from a NAS to a tunnel server, the L2F_CONF is the initial packet in the conversation.
L2F Creating new tunnel for nas1	Tunnel named nas1 is being created.
L2F Got a tunnel named nas1, responding	Tunnel server is responding.
L2F Open UDP socket to 172.21.9.25	Opening a socket to the NAS IP address.
L2F_OPEN received	L2F_OPEN management message was received, indicating that the NAS is opening an L2F tunnel.
L2F Removing resend packet (type 1)	Removing the resend packet for the L2F management packet. The two resend packet types have different meanings in different states of the tunnel.
L2F Got a MID management packet	L2F MID management packets are used to communicate between the NAS and the tunnel server.
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to up	Tunnel server is bringing up virtual access interface 1 for the L2F tunnel.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to up	Line protocol is up. The line can be used.
Tunnel coming down	
L2F_CLOSE received	NAS or tunnel server received a request to close the tunnel.
L2F Destroying mid	Connection identified by the MID is being taken down.
L2F Removing resend packet (type 3)	Removing the resend packet for the L2F management packet. There are two resend packets that have different meanings in different states of the tunnel.
L2F Tunnel is going down! L2F Initiating tunnel shutdown.	Device is performing normal operations when a tunnel is coming down.

Field	Description
%LINK-3-UPDOWN: Interface Virtual-Access1, changed state to down	The virtual access interface is coming down.
L2F_CLOSE received L2F Got closing for tunnel L2F Removing resend packet L2F Removing resend packet L2F Closed tunnel structure L2F Closed tunnel structure L2F Deleted inactive tunnel	Device is performing normal cleanup operations when the tunnel is being brought down.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access1, changed state to down	Line protocol is down; virtual access interface 1 cannot be used.

Examples

The following partial example of the **debug vpdn l2x-events** command is useful for monitoring a network running the L2TP Congestion Avoidance feature. The report shows that the congestion window (Cwnd) has been reset to 1 because of packet retransmissions:

```
Device# debug vpdn l2x-events
.
.
.
*Jul 15 19:02:57.963:  Tnl 47100 L2TP: Congestion Control event received is retransmission
*Jul 15 19:02:57.963:  Tnl 47100 L2TP: Congestion Window size, Cwnd 1
*Jul 15 19:02:57.963:  Tnl 47100 L2TP: Slow Start threshold, Ssthresh 2
*Jul 15 19:02:57.963:  Tnl 47100 L2TP: Remote Window size, 500
*Jul 15 19:02:57.963:  Tnl 47100 L2TP: Control channel retransmit delay set to 4 seconds
*Jul 15 19:03:01.607:  Tnl 47100 L2TP: Update ns/nr, peer ns/nr 2/5, our ns/nr 5/2
```

The following partial example shows that traffic has been restarted with L2TP congestion avoidance throttling traffic:

```
Device# debug vpdn l2x-events
.
.
.
*Jul 15 14:45:16.123:  Tnl 30597 L2TP: Control channel retransmit delay set to 2 seconds
*Jul 15 14:45:16.123:  Tnl 30597 L2TP: Tunnel state change from idle to wait-ctl-reply
*Jul 15 14:45:16.131:  Tnl 30597 L2TP: Congestion Control event received is positive
acknowledgement
*Jul 15 14:45:16.131:  Tnl 30597 L2TP: Congestion Window size, Cwnd 2
*Jul 15 14:45:16.131:  Tnl 30597 L2TP: Slow Start threshold, Ssthresh 500
*Jul 15 14:45:16.131:  Tnl 30597 L2TP: Remote Window size, 500
*Jul 15 14:45:16.131:  Tnl 30597 L2TP: Congestion Ctrl Mode is Slow Start
```

The table below describes the significant fields shown in the displays. See RFC 2661 for more details about the information in the reports for L2TP congestion avoidance.

Table 13: debug vpdn l2x-events Field Descriptions—L2TP Congestion Avoidance

Field	Description
Control channel retransmit delay set to ...	Indicates the current value set for the retransmit delay.
Tunnel state...	Indicates the tunnel's current Control Connection State, per RFC 2661.
Congestion Control event received is...	Indicates the received congestion control event. • Retransmission—Indicates packet retransmission has been detected in the resend queue. • Positive acknowledgement—Indicates that a packet was received and acknowledged by the peer tunnel endpoint.
Congestion Window size, Cwnd 2	Current size of the Cwnd.
Slow Start threshold, Ssthresh 500	Current value of the slow start threshold (Ssthresh).
Remote Window size, 500	Size of the advertised receive window configured on the remote peer with the **l2tp tunnel receive-window** command.
Congestion Ctrl Mode is...	Indicates whether the device is operating in Slow Start or Congestion Avoidance mode.
Update ns/nr, peer ns/nr 2/5, our ns/nr 5/2	See RFC 2661.

Examples

The following is sample output from the **debug vpdn error** command on a NAS when the L2F tunnel is not set up:

```
Device# debug vpdn error

%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down
%LINK-5-CHANGED: Interface Async1, changed state to reset
%LINK-3-UPDOWN: Interface Async1, changed state to down
%LINK-3-UPDOWN: Interface Async1, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up
VPDN tunnel management packet failed to authenticate
VPDN tunnel management packet failed to authenticate
```

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

Table 14: debug vpdn error Field Descriptions for the NAS

Field	Description
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to down	Line protocol on the asynchronous interface went down.
%LINK-5-CHANGED: Interface Async1, changed state to reset	Asynchronous interface 1 was reset.
%LINK-3-UPDOWN: Interface Async1, changed state to down %LINK-3-UPDOWN: Interface Async1, changed state to up	Link from asynchronous interface 1 link went down and then came back up.
%LINEPROTO-5-UPDOWN: Line protocol on Interface Async1, changed state to up	Line protocol on the asynchronous interface came back up.
VPDN tunnel management packet failed to authenticate	Tunnel authentication failed. This is the most common VPDN error. **Note** Verify the password for the NAS and the tunnel server name. If you store the password on an AAA server, you can use the **debug aaa authentication** command.

The following is sample output from the **debug vpdn l2x-errors** command:

```
Device# debug vpdn l2x-errors

%LINK-3-UPDOWN: Interface Async1, changed state to up
L2F Out of sequence packet 0 (expecting 0)
L2F Tunnel authentication succeeded for example.com
 L2F Received a close request for a non-existent mid
 L2F Out of sequence packet 0 (expecting 0)
 L2F packet has bogus1 key 1020868 D248BA0F
L2F packet has bogus1 key 1020868 D248BA0F
```

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

Table 15: debug vpdn l2x-errors Field Descriptions

Field	Description
%LINK-3-UPDOWN: Interface Async1, changed state to up	The line protocol on the asynchronous interface came up.
L2F Out of sequence packet 0 (expecting 0)	Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received.
L2F Tunnel authentication succeeded for example.com	Tunnel was established from the NAS to the tunnel server, example.com.

Field	Description
L2F Received a close request for a non-existent mid	Multiplex ID was not used previously; cannot close the tunnel.
L2F Out of sequence packet 0 (expecting 0)	Packet was expected to be the first in a sequence starting at 0, but an invalid sequence number was received.
L2F packet has bogus1 key 1020868 D248BA0F	Value based on the authentication response given to the peer during tunnel creation. This packet, in which the key does not match the expected value, must be discarded.
L2F packet has bogus1 key 1020868 D248BA0F	Another packet was received with an invalid key value. The packet must be discarded.

Examples

The following is sample output from the **debug vpdn l2x-packets** command on a NAS. This example displays a trace for a **ping** command.

```
Device# debug vpdn l2x-packets

L2F SENDING (17): D0 1 1 10 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 16 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 10 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F SENDING (17): D0 1 1 11 0 0 0 4 0 11 0 0 81 94 E1 A0 4
L2F header flags: 53249 version 53249 protocol 1 sequence 17 mid 0 cid 4
length 17 offset 0 key 1701976070
L2F RECEIVED (17): D0 1 1 11 0 0 0 4 0 11 0 0 65 72 18 6 5
L2F header flags: 57345 version 57345 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key 1701976070
L2F-IN Output to Async1 (16): FF 3 C0 21 9 F 0 C 0 1D 41 AD FF 11 46 87
L2F-OUT (16): FF 3 C0 21 A F 0 C 0 1A C9 BD FF 11 46 87
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 32 offset 0 key -2120949344
L2F-OUT (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 3 1 0 0 1 8 0 62 B1
0 0 C A8 0 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key -2120949344
L2F header flags: 49153 version 49153 protocol 2 sequence 0 mid 1 cid 4
length 120 offset 3 key 1701976070
L2F-IN Output to Async1 (101): 21 45 0 0 64 0 10 0 0 FF 1 B9 85 1 0 0 1 1 0
0 3 0 0 6A B1 0 0 C A8 0 0 0 0 0 11 E E0 AB CD AB CD AB CD AB CD AB CD
AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD
CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB CD AB
```

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

Table 16: debug vpdn l2x-packets Field Descriptions

Field	Description
L2F SENDING (17)	Number of bytes being sent. The first set of "SENDING"..."RECEIVED" lines displays L2F keepalive traffic. The second set displays L2F management data.
L2F header flags:	Version and flags, in decimal.
version 53249	Version number.
protocol 1	Protocol for negotiation of the point-to-point link between the NAS and the tunnel server is always 1, indicating L2F management.
sequence 16	Sequence numbers start at 0. Each subsequent packet is sent with the next increment of the sequence number. The sequence number is thus a free running counter represented modulo 256. There is a distinct sequence counter for each distinct MID value.
mid 0	MID, which identifies a particular connection within the tunnel. Each new connection is assigned a MID currently unused within the tunnel.
cid 4	Client ID used to assist endpoints in demultiplexing tunnels.
length 17	Size in octets of the entire packet, including header, all fields pre-sent, and payload. Length does not reflect the addition of the checksum, if present.
offset 0	Number of bytes past the L2F header at which the payload data is expected to start. If it is 0, the first byte following the last byte of the L2F header is the first byte of payload data.
key 1701976070	Value based on the authentication response given to the peer during tunnel creation. During the life of a session, the key value serves to resist attacks based on spoofing. If a packet is received in which the key does not match the expected value, the packet must be silently discarded.
L2F RECEIVED (17)	Number of bytes received.
L2F-IN Output to Async1 (16)	Payload datagram. The data came in to the VPDN code.

Field	Description
L2F-OUT (16):	Payload datagram sent out from the VPDN code to the tunnel.
L2F-OUT (101)	Ping payload datagram. The value 62 in this line is the ping packet size in hexadecimal (98 in decimal). The three lines that follow this line show ping packet data.

Examples

The following example shows output from the **debug vpdn l2x-events** command for an L2TP version 3 (L2TPv3) xconnect session on an Ethernet interface:

```
Device# debug vpdn l2x-events

23:31:18: L2X: l2tun session [1669204400], event [client request], old state [open], new
state [open]
 23:31:18: L2X: L2TP: Received L2TUN message <Connect>
 23:31:18: Tnl/Sn58458/28568 L2TP: Session state change from idle to wait-for-tunnel
 23:31:18: Tnl/Sn58458/28568 L2TP: Create session
 23:31:18: Tnl58458 L2TP: SM State idle
 23:31:18: Tnl58458 L2TP: O SCCRQ
 23:31:18: Tnl58458 L2TP: Control channel retransmit delay set to 1 seconds
 23:31:18: Tnl58458 L2TP: Tunnel state change from idle to wait-ctl-reply
 23:31:18: Tnl58458 L2TP: SM State wait-ctl-reply
 23:31:18: Tnl58458 L2TP: I SCCRP from router
 23:31:18: Tnl58458 L2TP: Tunnel state change from wait-ctl-reply to established
 23:31:18: Tnl58458 L2TP: O SCCCN to router tnlid 8012
 23:31:18: Tnl58458 L2TP: Control channel retransmit delay set to 1 seconds
 23:31:18: Tnl58458 L2TP: SM State established
 23:31:18: Tnl/Sn58458/28568 L2TP: O ICRQ to router 8012/0
 23:31:18: Tnl/Sn58458/28568 L2TP: Session state change from wait-for-tunnel to wait-reply

 23:31:19: Tnl58458 L2TP: Control channel retransmit delay set to 1 seconds
 23:31:20: %LINK-3-UPDOWN: Interface Ethernet2/1, changed state to up
 23:31:21: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet2/1, changed state to
up
 23:31:25: L2X: Sending L2TUN message <Connect OK>
 23:31:25: Tnl/Sn58458/28568 L2TP: O ICCN to router 8012/35149
 23:31:25: Tnl58458 L2TP: Control channel retransmit delay set to 1 seconds
 23:31:25: Tnl/Sn58458/28568 L2TP: Session state change from wait-reply to established
 23:31:25: L2X: l2tun session [1669204400], event [server response], old state [open], new
 state [open]
 23:31:26: Tnl58458 L2TP: Control channel retransmit delay set to 1 seconds
```

Examples

The following example shows debug messages for control channel authentication failure events in Cisco IOS Release 12.0(31)S:

```
Device# debug vpdn l2x-events

Tnl41855 L2TP: Per-Tunnel auth counter, Overall Failed, now 1
Tnl41855 L2TP: Tunnel auth counter, Overall Failed, now 219
```

Related Commands

Command	Description
debug aaa authentication	Displays information on AAA/TACACS+ authentication.
debug acircuit	Displays events and failures related to attachment circuits.
debug pppoe	Displays debugging information for PPPoE sessions.
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established sessions to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
debug xconnect	Displays errors and events related to an xconnect configuration.

debug vpdn pppoe-data

Note Effective with Cisco IOS Release 12.2(13)T, the **debug vpdn pppoe-data** command is replaced by the **debug pppoe** command. See the **debug pppoe** command for more information.

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

debug vpdn pppoe-data

no debug vpdn pppoe-data

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.2(13)T	This command was replaced by the **debug pppoe** command.

Usage Guidelines

The **debug vpdn pppoe-data**command displays a large number of debug messages and should generally be used only on a debug chassis with a single active session.

Examples

The following is sample output from the **debug vpdn pppoe-data**command:

```
Router# debug vpdn pppoe-data
6d20h:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up
6d20h:PPPoE:OUT
 contiguous pak, size 19
     FF 03 C0 21 01 01 00 0F 03 05 C2 23 05 05 06 D3
     FF 2B DA
6d20h:PPPoE:IN
 particle pak, size 1240
     C0 21 01 01 00 0A 05 06 39 53 A5 17 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
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 C0 21 02 01 00 0A 05 06 39 53 A5 17
6d20h:PPPoE:OUT
 contiguous pak, size 19
     FF 03 C0 21 01 02 00 0F 03 05 C2 23 05 05 06 D3
     FF 2B DA
6d20h:PPPoE:IN
 particle pak, size 1740
     C0 21 02 02 00 0F 03 05 C2 23 05 05 06 D3 FF 2B
```

```
     DA 00 80 C2 00 07 00 00 00 10 7B 01 2C D9 00 B0
     C2 EB 10 38 88 64 11 00
6d20h:PPPoE:OUT
 contiguous pak, size 30
     FF 03 C2 23 01 06 00 1A 10 99 1E 6E 8F 8C F2 C6
     EE 91 0A B0 01 CB 89 68 13 47 61 6E 67 61
6d20h:PPPoE:IN
 particle pak, size 3840
     C2 23 02 06 00 24 10 E6 84 FF 3A A4 49 19 CE D7
     AC D7 D5 96 CC 23 B3 41 6B 61 73 68 40 63 69 73
     63 6F 2E 63 6F 6D 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 8
     FF 03 C2 23 03 06 00 04
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 80 21 01 01 00 0A 03 06 65 65 00 66
6d20h:PPPoE:IN
 particle pak, size 1240
     80 21 01 01 00 0A 03 06 00 00 00 00 49 19 CE D7
     AC D7 D5 96 CC 23 B3 41 6B 61 73 68 40 63 69 73
     63 6F 2E 63 6F 6D 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 80 21 03 01 00 0A 03 06 65 65 00 67
6d20h:PPPoE:IN
 particle pak, size 1240
     80 21 02 01 00 0A 03 06 65 65 00 66 00 04 AA AA
     03 00 80 C2 00 07 00 00 00 10 7B 01 2C D9 00 B0
     C2 EB 10 38 88 64 11 00
6d20h:PPPoE:IN
 particle pak, size 1240
     80 21 01 02 00 0A 03 06 65 65 00 67 49 19 CE D7
     AC D7 D5 96 CC 23 B3 41 6B 61 73 68 40 63 69 73
     63 6F 2E 63 6F 6D 00 00
6d20h:PPPoE:OUT
 contiguous pak, size 14
     FF 03 80 21 02 02 00 0A 03 06 65 65 00 67
6d20h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access1,
changed state to up
6d20h:PPPoE:OUT
 contiguous pak, size 16
     FF 03 C0 21 09 01 00 0C D3 FF 2B DA 4C 4D 49 A4
6d20h:PPPoE:IN
 particle pak, size 1440
     C0 21 0A 01 00 0C 39 53 A5 17 4C 4D 49 A4 AA AA
     03 00 80 C2 00 07 00 00 00 10 7B 01 2C D9 00 B0
     C2 EB 10 38 88 64 11 00
6d20h:PPPoE:IN
 particle pak, size 1440
     C0 21 09 01 00 0C 39 53 A5 17 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
```

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

Table 17: debug vpdn pppoe-data Field Descriptions

Field	Descriptions
6d20h:%LINK-3-UPDOWN:Interface Virtual-Access1, changed state to up	Virtual access interface 1 came up.
6d20h:PPPoE:OUT	The host delivered a PPPoE session packet to the access concentrator.
6d20h:PPPoE:IN	The access concentrator received a PPPoE session packet.

Field	Descriptions
6d20h:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access1, changed state to up	Line protocol is up; the line can be used.
contiguous pak, size 19	Size 19 contiguous packet.
particle pak, size 1240	Size 1240 particle packet.

Related Commands

Command	Description
debug pppoe	Displays debugging information for PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established session to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is present.

debug vpdn pppoe-error

Note Effective with Cisco IOS Release 12.2(13)T, the **debug vpdn pppoe-error** command is replaced by the **debug pppoe** command. See the **debug pppoe** command for more information.

To display PPP over Ethernet (PPPoE) protocol errors that prevent a session from being established or errors that cause an established sessions to be closed, use the **debug vpdn pppoe-error**command in privileged EX**EC**mode. To disable debugging output, use the **no** form of this command.

debug vpdn pppoe-error

no debug vpdn pppoe-error

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.2(13)T	This command was replaced by the **debug pppoe** command.

Examples

The following is a full list of error messages displayed by the **debug vpdn pppoe-error**command:

```
PPPOE:pppoe_acsys_err cannot grow packet
PPPoE:Cannot find PPPoE info
PPPoE:Bad MAC address:00b0c2eb1038
PPPOE:PADI has no service name tag
PPPoE:pppoe_handle_padi cannot add AC name/Cookie.
PPPoE:pppoe_handle_padi cannot grow packet
PPPoE:pppoe_handle_padi encap failed
PPPoE cannot create virtual access.
PPPoE cannot allocate session structure.
PPPoE cannot store session element in tunnel.
PPPoE cannot allocate tunnel structure.
PPPoE cannot store tunnel
PPPoE:VA221:No Session, Packet Discarded
PPPOE:Tried to shutdown a null session
PPPoE:Session already open, closing
PPPoE:Bad cookie:src_addr=00b0c2eb1038
PPPoE:Max session count on mac elem exceeded:mac=00b0c2eb1038
PPPoE:Max session count on vc exceeded:vc=3/77
PPPoE:Bad MAC address - dropping packet
PPPoE:Bad version or type - dropping packet
```

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

Table 18: debug vpdn pppoe-error Field Descriptions

Field	Descriptions
PPPOE:pppoe_acsys_err cannot grow packet	Asynchronous PPPoE packet initialization error.
PPPoE:Cannot find PPPoE info	The access concentrator sends a PADO to the host.
PPPoE:Bad MAC address:00b0c2eb1038	The host was unable to identify the Ethernet MAC address.
PPPOE:PADI has no service name tag	PADI requires a service name tag.
PPPoE:pppoe_handle_padi cannot add AC name/Cookie.	pppoe_handle_padi could not append AC name.
PPPoE:pppoe_handle_padi cannot grow packet	pppoe_handle_padi could not append packet.
PPPoE:pppoe_handle_padi encap failed	pppoe_handle_padi could not specify PPPoE on ATM encapsulation.
PPPoE cannot create virtual access.	PPPoE session unable to verify virtual access interface.
PPPoE cannot allocate session structure.	PPPoE session unable to allocate Stage Protocol.
PPPoE cannot store session element in tunnel.	PPPoE tunnel cannot allocate session element.
PPPoE cannot allocate tunnel structure.	PPPoE tunnel unable to allocate Stage Protocol.
PPPoE cannot store tunnel	PPPoE configuration settings unable to initialize a tunnel.
PPPoE:VA221:No Session, Packet Discarded	No sessions created. All packets dropped.
PPPOE:Tried to shutdown a null session	Null session shutdown.
PPPoE:Session already open, closing	PPPoE session already open.
PPPoE:Bad cookie:src_addr=00b0c2eb1038	PPPoE session unable to append new cookie.
PPPoE:Max session count on mac elem exceeded:mac=00b0c2eb1038	The maximum number of sessions exceeded the Ethernet MAC address.
PPPoE:Max session count on vc exceeded:vc=3/77	The maximum number of sessions exceeded the PVC connection.
PPPoE:Bad MAC address - dropping packet	The host was unable to identify the MAC address. Packet dropped.

Field	Descriptions
PPPoE:Bad version or type - dropping packet	The host was unable to identify the encapsulation type.

Related Commands

Command	Description
debug pppoe	Displays debugging information for PPPoE sessions.
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpdn pppoe-events

Note Effective with Cisco IOS Release 12.2(13)T, the **debug vpdn pppoe-events**command is replaced by the **debug pppoe** command. See the **debug pppoe** command for more information.

To display PPP over Ethernet (PPPoE) protocol messages about events that are part of normal session establishment or shutdown, use the **debug vpdn pppoe-events**command in privileged E**XEC**mode. To disable debugging output, use the **no** form of this command.

debug vpdn pppoe-events

no debug vpdn pppoe-events

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.2(13)T	This command was replaced by the **debug pppoe** command.

Examples

The following is sample output from the **debug vpdn pppoe-events**command:

```
1w5d:IN PADI from PPPoE tunnel
1w5d:OUT PADO from PPPoE tunnel
1w5d:IN PADR from PPPoE tunnel
1w5d:PPPoE:VPN session created.
1w5d:%LINK-3-UPDOWN:Interface Virtual-Access2, changed state to up
1w5d:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access2, changed state to up
```

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

Table 19: debug vpdn pppoe-events Field Descriptions

Field	Descriptions
1w5d:IN PADI from PPPoE tunnel	The access concentrator receives an Active Discovery Initiation (PADI) packet from the PPPoE tunnel.
1w5d:OUT PADO from PPPoE tunnel	The access concentrator sends an Active Discovery Offer (PADO) to the host.

Field	Descriptions
1w5d:IN PADR from PPPoE tunnel	The host sends a single Active Discovery Request (PADR) to the access concentrator that it has chosen.
1w5d:PPPoE:VPN session created.	The access concentrator receives the PADR packet and creates a virtual private network (VPN) session.
1w5d:%LINK-3-UPDOWN:Interface Virtual-Access2, changed state to up	Virtual access interface 2 came up.
1w5d:%LINEPROTO-5-UPDOWN:Line protocol on Interface Virtual-Access2, changed state to up	Line protocol is up. The line can be used.

Related Commands

Command	Description
debug pppoe	Displays debugging information for PPPoE sessions.
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established session to be closed.
debug vpdn pppoe-packet	Displays each PPPoE protocol packet exchanged.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpdn pppoe-packet

Note Effective with Cisco IOS Release 12.2(13)T, the **debug vpdn pppoe-packet**command is replaced by the **debug pppoe** command. See the **debug pppoe** command for more information.

To display each PPP over Ethernet (PPPoE) protocol packet exchanged, use the **debug vpdn pppoe-packet**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpdn pppoe-packet

no debug vpdn pppoe-packet

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Command History

Release	Modification
12.1(1)T	This command was introduced.
12.2(13)T	This command was replaced by the **debug pppoe** command.

Usage Guidelines

The **debug vpdn pppoe-packet**command displays a large number of debug messages and should generally only be used on a debug chassis with a single active session.

Examples

The following is sample output from the **debug vpdn pppoe-packet**command:

```
PPPoE control packets debugging is on
1w5d:PPPoE:discovery packet
 contiguous pak, size 74
     FF FF FF FF FF FF 00 10 7B 01 2C D9 88 63 11 09
     00 00 00 04 01 01 00 00 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...
1w5d:OUT PADO from PPPoE tunnel
 contiguous pak, size 74
     00 01 09 00 AA AA 03 00 80 C2 00 07 00 00 00 10
     7B 01 2C D9 00 90 AB 13 BC A8 88 63 11 07 00 00
     00 20 01 01 00 00 01 02 00 04 41 67 6E 69 01 ...
1w5d:PPPoE:discovery packet
 contiguous pak, size 74
     00 90 AB 13 BC A8 00 10 7B 01 2C D9 88 63 11 19
     00 00 00 20 01 01 00 00 01 02 00 04 41 67 6E 69
     01 04 00 10 B7 4B 86 5B 90 A5 EF 11 64 A9 BA ...
```

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

Table 20: debug vpdn pppoe-packet Field Descriptions

Field	Descriptions
PPPoE control packets debugging is on	PPPoE debugging of packets is enabled.
1w5d:PPPoE:discovery packet	The host performs a discovery to initiate a PPPoE session.
1w5d:OUT PADO from PPPoE tunnel	The access concentrator sends a PADO to the host.
1w5d:PPPoE:discovery packet	The host performs a discovery to initiate a PPPoE session.
contiguous pak, size 74	Size 74 contiguous packet.

Related Commands

Command	Description
debug pppoe	Displays debugging information for PPPoE sessions.
debug vpdn pppoe-data	Displays data packets of PPPoE sessions.
debug vpdn pppoe-error	Displays PPPoE protocol errors that prevent a session from being established or errors that cause an established session to be closed.
debug vpdn pppoe-events	Displays PPPoE protocol messages about events that are part of normal session establishment or shutdown.
protocol (VPDN)	Specifies the L2TP that the VPDN subgroup will use.
show vpdn	Displays information about active L2F protocol tunnel and message identifiers in a VPDN.
vpdn enable	Enables virtual private dialup networking on the router and informs the router to look for tunnel definitions in a local database and on a remote authorization server (home gateway), if one is pre-sent.

debug vpdn redundancy

To debug virtual private dial-up network (VPDN) sessions that contain redundancy status, use the **debug vpdn redundancy**command in user or privileged EXEC mode. To disable this debugging, use the **no** form of this command.

debug vpdn redundancy {**cf**| **detail**| **error**| **event**| **fsm**| **resync**| **rf**}

no debug vpdn redundancy

Syntax Description

cf	Displays VPDN redundancy-facility (cf) events.
detail	Displays VPDN redundancy details.
error	Displays VPDN redundancy errors.
event	Displays VPDN redundancy events.
fsm	Displays VPDN redundancy forwarding-service manager (fsm) events.
resync	Displays VPDN redundancy resynchronizations.
rf	Displays VPDN redundancy-facility (rf) events.

Command Modes

User EXEC (>) Privileged EXEC (#)

Command History

Release	Modification
Cisco IOS XE Release 2.2.	This command was introduced in Cisco IOS XE Release 2.2.

Usage Guidelines

Use the **debug vpdn redundancy** command in privileged EXEC mode to display a list of VPDN sessions that have redundancy events and errors.

Use the **show vpdn redundancy**command in privileged EXEC mode to display information on the state of the VPDN session redundancy data.

Examples

The following example shows how to display a debug of redundancy events during the setup and termination of an tunnel for an LNS active Route Processor (RP):

```
LNS1> debug
 enable
LNS1# debug
```

```
 vpdn
 redundancy
 cf
L2TP redundancy cf debugging is on
LNS1# debug
 vpdn
 redundancy
 detail
L2TP redundancy details debugging is on
LNS1# debug
 vpdn
 redundancy
 error
L2TP redundancy errors debugging is on
LNS1# debug
 vpdn
 redundancy
 event
L2TP redundancy events debugging is on
LNS1# debug
 vpdn
 redundancy
 fsm
L2TP redundancy fsm debugging is on
LNS1# debug
 vpdn
 redundancy
 resync
L2TP redundancy resync debugging is on
LNS1# debug
 vpdn
 redundancy
 rf
L2TP redundancy rf debugging is on
LNS1#
*Aug 26 18:00:00.467: %SYS-5-CONFIG_I: Configured from console by console
LNS1#
*Aug 26 18:00:45.631: L2TP tnl   01000:________: CCM initialized CCM session
*Aug 26 18:00:45.631: : L2TP HA:CC playback chkpt skipped, CC not doing HA
*Aug 26 18:00:45.711: : L2TP HA FSM:Receive proto FSM event 19
*Aug 26 18:00:45.711: : L2TP HA FSM:Receive RxSCCRQ
*Aug 26 18:00:45.711: : L2TP HA:lcm_cc alloc: l2tp_cc 070B45B8, lcm_cc 02FE55E8
*Aug 26 18:00:45.711: : L2TP HA FSM:FSM-CC ev Rx-SCCRQ
*Aug 26 18:00:45.711: : L2TP HA FSM:FSM-CC    Idle->Wt-ChkptSidRmt
*Aug 26 18:00:45.711: : L2TP HA FSM:FSM-CC do Block-Tx-AckSCCRQ
*Aug 26 18:00:45.711: : L2TP HA FSM:Checkpoint Two Cc IDs
*Aug 26 18:00:45.711: L2TP HA CF: Chkpt send: s/c id 0/52631, BothCcId, seq 0, ns/nr 0/0,
rid 51583, len 52; flush = 1, ctr 1
*Aug 26 18:00:45.711: 01000:0000CD97: L2TP HA:Enqueue peer Ns 0 to ns_q, seq 1 (q sz 0)
*Aug 26 18:00:45.711: L2TP tnl   01000:0000CD97: Encoding SCCRQ-IN CHKPT
*Aug 26 18:00:45.711: L2TP tnl   01000:0000CD97: Tx CHKPT
*Aug 26 18:00:45.739: L2TP tnl   01000:0000CD97: Encoding SCCRP-OUT CHKPT
*Aug 26 18:00:45.739: L2TP tnl   01000:0000CD97: Tx CHKPT
*Aug 26 18:00:45.739: : L2TP HA:Adjust local window size to 10
*Aug 26 18:00:45.739: 01000:0000CD97: L2TP HA FSM:Receive proto TxCM event SCCRP
*Aug 26 18:00:45.739: : L2TP HA FSM:Receive TxSCCRP
LNS1#
*Aug 26 18:00:45.739: : L2TP HA FSM:FSM-CC ev Tx-SCCRP
*Aug 26 18:00:45.739: : L2TP HA FSM:FSM-CC    Wt-ChkptSidRmt->WtCcIdRmt2
*Aug 26 18:00:45.739: : L2TP HA FSM:FSM-CC do Block-Tx-SCCRP
*Aug 26 18:00:45.739: 01000:0000CD97: L2TP HA FSM:Found blocked RxSCCRQ, seq_num 1
*Aug 26 18:00:45.739: 01000:0000CD97: L2TP HA FSM:Queued SCCRP to CC hold_q
*Aug 26 18:00:46.863: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:46.863: : L2TP HA FSM:Context s/c id 0/52631, BothCcId, seq 1, ns/nr 0/0, rid
 51583, len 52
*Aug 26 18:00:46.863: L2TP HA CF: Rcvd status s/c id 0/52631, BothCcId, seq 1, ns/nr 0/0,
rid 51583, len 52
*Aug 26 18:00:46.863: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:46.863: L2TP HA CF: Status content s/c id 0/52631, BothCcId, seq 1, ns/nr
0/0, rid 51583, len 52
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 0/52631, BothCcId,
 seq 1, ns/nr 0/0, rid 51583, len 52
```

```
*Aug 26 18:00:46.863: : L2TP HA FSM:Receive CC-ChkptAck
*Aug 26 18:00:46.863: : L2TP HA FSM:FSM-CC ev Rx-CkpACK-CcID-Rmt
*Aug 26 18:00:46.863: : L2TP HA FSM:FSM-CC    WtCcIdRmt2->Wt-RxSccn
*Aug 26 18:00:46.863: : L2TP HA FSM:FSM-CC do Allow-Tx-SCCRP2
*Aug 26 18:00:46.863: : L2TP HA FSM:Received Chkpt of local + remote CC ID
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:Try to remove from CC's ns_q: seq num 1
(current Ns 1)
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:Ns entry to remove: found (current Ns 1)
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:Advance peer Nr to 1 (ns_q sz 0)
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:CC send all unblocked if can
LNS1#
*Aug 26 18:00:46.863: 01000:0000CD97: L2TP HA:CC send one blocked CM (SCCRP): ns 0 (0), nr
 1
*Aug 26 18:00:46.863: L2TP HA CF: O SCCRP 51583/0 ns/nr 0/1
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Receive Cm-Ack
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Receive CC Cm-Ack
*Aug 26 18:00:47.867: : L2TP HA FSM:FSM-CC ev Rx-CmACK
*Aug 26 18:00:47.867: : L2TP HA FSM:FSM-CC    in Wt-RxSccn
*Aug 26 18:00:47.867: : L2TP HA FSM:FSM-CC do Ignore
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Ignore event
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 1, peer 1
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (1,0/1,1, int
1, rx 1, 1) (ns_q sz 0)
*Aug 26 18:00:47.867: 01000:0000CD97: L2TP HA FSM:Peer Ns 1 (1), Nr 1 (ns_q sz 0)
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 1, peer 1
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (1,0/1,1, int
1, rx 1, 1) (ns_q sz 0)
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Peer Ns 1 (2), Nr 1 (ns_q sz 0)
*Aug 26 18:00:48.087: : L2TP HA FSM:Receive proto FSM event 21
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Receive RxSCCCN
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC ev Rx-SCCCN
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC    Wt-RxSccn->WtCcsUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC do Allow-Tx-AckSCCCN
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Allow TxSCCCN-ACK
*Aug 26 18:00:48.087: 01000:0000CD97: L2TP HA FSM:Receive CcUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC ev Proto CcUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC    WtCcsUp->Wt-CkptCcUp
*Aug 26 18:00:48.087: : L2TP HA FSM:FSM-CC do Chkpt-CcUp2
*Aug 26 18:00:48.087: : L2TP HA FSM:Checkpoint CcUp
*Aug 26 18:00:48.087: L2TP HA CF: Chkpt send: s/c id 0/52631, CcUp, seq 0, ns/nr 1/1, rid
0, len 52; flush = 1, ctr 2
*Aug 26 18:00:48.091: L2TP tnl   01000:0000CD97: CCM added sync data
*Aug 26 18:00:48.095: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 2, peer 1
*Aug 26 18:00:48.095: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (2,1/1,1, int
2, rx 1, 2) (ns_q sz 0)
*Aug 26 18:00:48.095: 01000:0000CD97: L2TP HA FSM:Peer Ns 2 (3), Nr 1 (ns_q sz 0)
*Aug 26 18:00:48.095: L2TP _____:01000:000036F8: Encoding ICRQ-IN CHKPT
*Aug 26 18:00:48.095: L2TP _____:01000:000036F8: Tx CHKPT
*Aug 26 18:00:48.095: : L2TP HA FSM:Receive proto FSM event 3
*Aug 26 18:00:48.095: : L2TP HA FSM:Receive RxICRQ
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:  Using ICRQ FSM
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev created
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Init->Idle
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do none
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-xCRQ
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Idle->Wt-ChkptSidRmt
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Block-Tx-AckXCRQ
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA FSM:Checkpoint TwoSessionIDs
*Aug 26 18:00:48.095: L2TP HA CF: Chkpt send: s/c id 14072/52631, BothSesId, seq 0, ns/nr
1/2, rid 40276, len 52; flush = 1, ctr 3
*Aug 26 18:00:48.095: _____:01000:000036F8: L2TP HA:Enqueue peer Ns 2 to ns_q, seq 3 (q sz
 0)
*Aug 26 18:00:48.131: : L2TP HA:Try to buffer sock msg type 19
*Aug 26 18:00:48.131: : L2TP HA:Buffering skipped
*Aug 26 18:00:48.131: L2TP _____:01000:000036F8: Encoding ICRP-OUT CHKPT
*Aug 26 18:00:48.131: L2TP _____:01000:000036F8: Tx CHKPT
*Aug 26 18:00:48.131: 01000:0000CD97: L2TP HA FSM:Receive proto TxCM event ICRP
*Aug 26 18:00:48.131: _____:_____:000036F8: L2TP HA FSM:Receive TxICRP
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Tx-xCRP
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-ChkptSidRmt->Wt-SesIdRmt2
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Block-Tx-xCRP
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:Found blocked RxICRQ, seq_num 3
LNS1#
```

```
*Aug 26 18:00:48.131: _____:01000:000036F8: L2TP HA FSM:Queued xCRP to session hold_q
*Aug 26 18:00:48.131: : L2TP HA:Try to buffer sock msg type 23
*Aug 26 18:00:48.131: : L2TP HA:CC not in resync state, buffering skipped
*Aug 26 18:00:49.115: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 2, peer 1
*Aug 26 18:00:49.115: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (2,2/1,1, int
3, rx 1, 3) (ns_q sz 1)
*Aug 26 18:00:49.211: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:49.211: : L2TP HA FSM:Context s/c id 0/52631, CcUp, seq 2, ns/nr 1/1, rid 0,
 len 52
*Aug 26 18:00:49.211: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:49.211: : L2TP HA FSM:Context s/c id 14072/52631, BothSesId, seq 3, ns/nr
1/2, rid 40276, len 52
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status s/c id 0/52631, CcUp, seq 2, ns/nr 1/1, rid
0, len 52
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:49.211: L2TP HA CF: Status content s/c id 0/52631, CcUp, seq 2, ns/nr 1/1,
rid 0, len 52
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 0/52631, CcUp, seq
 2, ns/nr 1/1, rid 0, len 52
*Aug 26 18:00:49.211: : L2TP HA FSM:Receive CC-ChkptAck
*Aug 26 18:00:49.211: : L2TP HA FSM:FSM-CC ev Rx-CkpACK-CcUp
*Aug 26 18:00:49.211: : L2TP HA FSM:FSM-CC    Wt-CkptCcUp->ProcCcsUp
*Aug 26 18:00:49.211: : L2TP HA FSM:FSM-CC do Proc-ChpACK-CcUp2
*Aug 26 18:00:49.211: : L2TP HA FSM:Received chkpt ACK of CcUp
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status s/c id 14072/52631, BothSesId, seq 3, ns/nr
1/2, rid 40276, len 52
*Aug 26 18:00:49.211: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:49.211: L2TP HA CF: Status content s/c id 14072/52631, BothSesId, seq 3, ns/nr
 1/2, rid 40276, len 52
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 14072/52631,
BothSesId, seq 3, ns/nr 1/2, rid 40276, len 52
*Aug 26 18:00:49.211: _____:_____:000036F8: L2TP HA FSM:Receive Session-ChkptAck
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-CktACK-SesID-Rmt
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-SesIdRmt2->Wt-RxXccn
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Allow-Tx-xCRP
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA:Try to remove from CC's ns_q: seq num 3
(current Ns 3)
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA:Ns entry to remove: found (current Ns
3)
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA:Advance peer Nr to 3 (ns_q sz 0)
*Aug 26 18:00:49.211: _____:01000:000036F8: L2TP HA:Session send all unblocked
*Aug 26 18:00:49.211: 01000:0000CD97: L2TP HA:CC send if can (ICRP): ns 1 (1, 1), nr 3 (3)
*Aug 26 18:00:49.211: L2TP HA CF: O ICRP 51583/40276 ns/nr 1/3
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Receive Cm-Ack
*Aug 26 18:00:49.231: _____:_____:000036F8: L2TP HA FSM:Receive session Cm-Ack
LNS1#
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-CmACK
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    in Wt-RxXccn
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Ignore
*Aug 26 18:00:49.231: _____:01000:000036F8: L2TP HA FSM:Ignore event
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 3, peer 2
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (3,2/2,2, int
3, rx 2, 3) (ns_q sz 0)
*Aug 26 18:00:49.231: 01000:0000CD97: L2TP HA FSM:Peer Ns 3 (3), Nr 2 (ns_q sz 0)
LNS1#
*Aug 26 18:00:50.407: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 3, peer 2
*Aug 26 18:00:50.407: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (3,2/2,2, int
3, rx 2, 3) (ns_q sz 0)
*Aug 26 18:00:50.407: 01000:0000CD97: L2TP HA FSM:Peer Ns 3 (4), Nr 2 (ns_q sz 0)
*Aug 26 18:00:50.407: : L2TP HA FSM:Receive proto FSM event 5
*Aug 26 18:00:50.407: _____:_____:000036F8: L2TP HA FSM:Receive RxICCN
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-xCCN
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-RxXccn->Wt-SessUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Allow-Tx-AckXCCN
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:Allow TxICCN-ACK
*Aug 26 18:00:50.407: L2TP _____:01000:000036F8: Encoding ICCN-IN CHKPT
*Aug 26 18:00:50.407: L2TP _____:01000:000036F8: Tx CHKPT
*Aug 26 18:00:50.407: _____:_____:000036F8: L2TP HA FSM:Receive SessionUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Proto SessUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-SessUp->Wt-CkptSesUp
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Chkpt-SesUp2
*Aug 26 18:00:50.407: _____:01000:000036F8: L2TP HA FSM:Checkpoint SessionUP
*Aug 26 18:00:50.407: L2TP HA CF: Chkpt send: s/c id 14072/52631, SesUp, seq 0, ns/nr 2/3,
```

```
 rid 0, len 52; flush = 1, ctr 4
*Aug 26 18:00:51.055: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:00:51.055: : L2TP HA FSM:Context s/c id 14072/52631, SesUp, seq 4, ns/nr 2/3,
rid 0, len 52
*Aug 26 18:00:51.055: L2TP HA CF: Rcvd status s/c id 14072/52631, SesUp, seq 4, ns/nr 2/3,
 rid 0, len 52
*Aug 26 18:00:51.055: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:00:51.055: L2TP HA CF: Status content s/c id 14072/52631, SesUp, seq 4, ns/nr
2/3, rid 0, len 52
*Aug 26 18:00:51.055: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 14072/52631, SesUp,
 seq 4, ns/nr 2/3, rid 0, len 52
*Aug 26 18:00:51.055: _____:_____:000036F8: L2TP HA FSM:Receive Session-ChkptAck
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:FSM-Sn ev Rx-CktACK-SesUp
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:FSM-Sn    Wt-CkptSesUp->Proc-SessUp
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:FSM-Sn do Proc-ChpACK-SesUp
*Aug 26 18:00:51.055: _____:01000:000036F8: L2TP HA FSM:Received chkpt ACK of SessionUP
*Aug 26 18:00:51.347: %LINK-3-UPDOWN: Interface Virtual-Access2, changed state to up
LNS1#
*Aug 26 18:00:51.635: : L2TP HA:Try to buffer sock msg type 26
*Aug 26 18:00:51.635: : L2TP HA:CC not in resync state, buffering skipped
*Aug 26 18:00:51.659: : L2TP HA:Try to buffer sock msg type 26
*Aug 26 18:00:51.659: : L2TP HA:CC not in resync state, buffering skipped
LNS1#
*Aug 26 18:00:52.363: %LINEPROTO-5-UPDOWN: Line protocol on Interface Virtual-Access2,
changed state to up
LNS1#
LNS1# clear
 vpdn
 all
Proceed with clearing all tunnels? [confirm]
LNS1#
*Aug 26 18:01:21.271: 00001:_____:000036F8: L2TP HA FSM:Receive Session-CC-Rm
*Aug 26 18:01:21.271: 00001:_____:000036F8: L2TP HA FSM:Receive SessionRm
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Receive proto TxCM event StopCCN
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Receive TxSTOPCCN
*Aug 26 18:01:21.271: : L2TP HA FSM:FSM-CC ev Tx-STOPCCN
*Aug 26 18:01:21.271: : L2TP HA FSM:FSM-CC    ProcCcsUp->Wt-CkptCcDn
*Aug 26 18:01:21.271: : L2TP HA FSM:FSM-CC do Chkpt-CcDwn
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Receive TxSTOPCCN while CC up
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA:CC ns_q cleanup: overall head Ns old/new =
4/4 (Q sz 0)
LNS1#
*Aug 26 18:01:21.271: : L2TP HA FSM:Checkpoint CCDown
*Aug 26 18:01:21.271: L2TP HA CF: Chkpt send: s/c id 0/52631, CcDwn, seq 0, ns/nr 2/3, rid
 0, len 52; flush = 1, ctr 5
*Aug 26 18:01:21.271: 01000:0000CD97: L2TP HA FSM:Queued STOPCCN to cc hold_q
*Aug 26 18:01:21.295: : L2TP HA:Try to buffer sock msg type 22
*Aug 26 18:01:21.295: : L2TP HA:Buffering skipped
*Aug 26 18:01:22.423: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:01:22.423: : L2TP HA FSM:Context s/c id 0/52631, CcDwn, seq 5, ns/nr 2/3, rid
0, len 52
*Aug 26 18:01:22.423: L2TP HA CF: Rcvd status s/c id 0/52631, CcDwn, seq 5, ns/nr 2/3, rid
 0, len 52
*Aug 26 18:01:22.423: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:01:22.423: L2TP HA CF: Status content s/c id 0/52631, CcDwn, seq 5, ns/nr 2/3,
rid 0, len 52
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA FSM:Recv chkpt ack: s/c id 0/52631, CcDwn,
seq 5, ns/nr 2/3, rid 0, len 52
*Aug 26 18:01:22.423: : L2TP HA FSM:Receive CC-ChkptAck
*Aug 26 18:01:22.423: : L2TP HA FSM:FSM-CC ev Rx-CkpACK-CcDwn
*Aug 26 18:01:22.423: : L2TP HA FSM:FSM-CC    Wt-CkptCcDn->Wt-RxStopAck
*Aug 26 18:01:22.423: : L2TP HA FSM:FSM-CC do Allow-Tx-STOPCCN4
*Aug 26 18:01:22.423: : L2TP HA FSM:Received Chkpt of CC removal
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:Try to remove from CC's ns_q: seq num 5
(current Ns 4)
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:Ns entry to remove: not found (current Ns 4)
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:CC send all unblocked if can
*Aug 26 18:01:22.423: 01000:0000CD97: L2TP HA:CC send one blocked CM (SCCRP): ns 2 (2), nr
 4
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive Cm-Ack
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive CC Cm-Ack
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC ev Rx-CmACK
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC    Wt-RxStopAck->Wt-CkptCcRm
```

```
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC do ChkptCcRm3
*Aug 26 18:01:22.451: : L2TP HA FSM:Received STOPCCN-ACK while waiting for it, checkpoint
CCRm and remove cc
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA:CC ns_q cleanup: overall head Ns old/new =
4/4 (Q sz 0)
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Checkpoint CcRm
*Aug 26 18:01:22.451: L2TP HA CF: Chkpt send: s/c id 0/52631, CcRm, seq 0, ns/nr 3/3, rid
0, len 52; flush = 1, ctr 6
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Check for Ns/Nr update 4, peer 3
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive peer Ns/Nr update (4,3/3,3, int
4, rx 3, 4) (ns_q sz 0)
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Peer Ns 4 (4), Nr 3 (ns_q sz 0)
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:Receive CC-Rm
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC ev Proto CcRm
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC    Wt-CkptCcRm->End
*Aug 26 18:01:22.451: : L2TP HA FSM:FSM-CC do RmCc3
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA FSM:CC destruction after Tx/Rx StopCCN
LNS1#
*Aug 26 18:01:22.451: 01000:0000CD97: L2TP HA:CC ns_q cleanup: overall head Ns old/new =
4/4 (Q sz 0)
*Aug 26 18:01:22.451: : L2TP HA FSM:Checkpoint CCRm
*Aug 26 18:01:22.451: L2TP HA CF: Chkpt send: s/c id 0/52631, CcRm, seq 0, ns/nr 3/3, rid
0, len 52; flush = 1, ctr 7
*Aug 26 18:01:22.451: : L2TP HA:lcm_cc free: l2tp_cc 070B45B8, lcm_cc 02FE55E8
*Aug 26 18:01:22.451: L2TP tnl _____:________: CCM setting state to DOWN
*Aug 26 18:01:23.571: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:01:23.571: : L2TP HA FSM:Context s/c id 0/52631, CcRm, seq 6, ns/nr 3/3, rid 0,
 len 52
*Aug 26 18:01:23.571: : L2TP HA FSM:CHKPT status callback: status 0, len 56
*Aug 26 18:01:23.571: : L2TP HA FSM:Context s/c id 0/52631, CcRm, seq 7, ns/nr 3/3, rid 0,
 len 52
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status s/c id 0/52631, CcRm, seq 6, ns/nr 3/3, rid
0, len 52
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:01:23.571: L2TP HA CF: Status content s/c id 0/52631, CcRm, seq 6, ns/nr 3/3,
rid 0, len 52
*Aug 26 18:01:23.571: : L2TP HA FSM:Ignore chkpt ACK: CC not found.
LNS1#
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status s/c id 0/52631, CcRm, seq 7, ns/nr 3/3, rid
0, len 52
*Aug 26 18:01:23.571: L2TP HA CF: Rcvd status 0: len 56
*Aug 26 18:01:23.571: L2TP HA CF: Status content s/c id 0/52631, CcRm, seq 7, ns/nr 3/3,
rid 0, len 52
*Aug 26 18:01:23.571: : L2TP HA FSM:Ignore chkpt ACK: CC not found.
LNS1#
*Aug 26 18:01:35.771: %REDUNDANCY-3-STANDBY_LOST: Standby processor fault
(PEER_DOWN_INTERRUPT)
```

The table below describes significant fields shown in the **debug vpdn redundancy** command output.

Table 21: debug vpdn redundancy Command Field Descriptions

Field	Description
cf	Number of L2TP checkpointing-facility events (cf-events).
error	Number of L2TP checkpointing errors.
event	Number of L2TP checkpointing events.
fsm	Number of L2TP checkpointing fsm events.
resync	Number of L2TP checkpointing resynchronized events.

Field	Description
rf	Number of L2TP checkpointing redundancy-facility events (rf-events).

Related Commands

Command	Description
debug l2tp redundancy	Displays information about L2TP sessions that have redundancy events and errors.
l2tp sso enable	Enables L2TP High Availability (HA).
l2tp tunnel resync	Specifies the number of packets sent before waiting for an acknowledgement message.
show l2tp redundancy	Displays L2TP sessions containing redundancy data.
show vpdn redundancy	Displays VPDN sessions containing redundancy data.
sso enable	Enables L2TP HA for VPDN groups.

debug vpm all

To enable all voice port module (VPM) debugging, use the **debug vpm all** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm all

no debug vpm all

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(1)T	This command was introduced for the Cisco 3600 series.
12.0(7)XK	This command was updated for the Cisco 2600, Cisco 3600, and Cisco MC3810 series devices.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Use the **debug vpm all** command to enable the complete set of VPM debugging commands: **debug vpm dsp**, **debug vpm error**, **debug vpm port**, **debug vpm spi**, and **debug vpm trunk_sc**.

Execution of **no debug all** will turn off all port level debugging. It is usually a good idea to turn off all debugging and then enter the **debug** commands you are interested in one by one. This will help to avoid confusion about which ports you are actually debugging.

Examples

For sample outputs, refer to the documentation of the other **debup vpm** commands.

Related Commands

Command	Description
debug vpm port	Limits the **debug vpm all** command to a specified port.
show debug	Displays which debug commands are enabled.
debug vpm error	Enables DSP error tracing.

Command	Description
debug vpm voaal2 all	Enables the display of trunk conditioning supervisory component trace information.

debug vpm dsp

To show messages from the digital signal processor (DSP) on the voice port module VPM) to the router, use the **debug vpm dsp**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm dsp

no debug vpm dsp

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

The **debug vpm dsp** command shows messages from the DSP on the VPM to the router; this command can be useful if you suspect that the VPM is not functional. It is a simple way to check if the VPM is responding to off-hook indications and to evaluate timing for signaling messages from the interface.

Examples

The following output shows the DSP time stamp and the router time stamp for each event. For SIG_STATUS, the state value shows the state of the ABCD bits in the signaling message. This sample shows a call coming in on an FXO interface.

The router waits for ringing to terminate before accepting the call. State=0x0 indicates ringing; state 0x4 indicates not ringing.

```
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x0 timestamp=58172 systime=40024
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x4 timestamp=59472 systime=40154
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0x4 timestamp=59589 systime=40166
```

The following output shows the digits collected:

```
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=4
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=1
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
vcsm_dsp_message: MSG_TX_DTMF_DIGIT: digit=0
```

This shows the disconnect indication and the final call statistics reported by the DSP (which are then populated in the call history table):

```
ssm_dsp_message: SEND/RESP_SIG_STATUS: state=0xC timestamp=21214 systime=42882
vcsm_dsp_message: MSG_TX_GET_TX_STAT: num_tx_pkts=1019 num_signaling_pkts=0
num_comfort_noise_pkts=0 transmit_durtation=24150 voice_transmit_duration=20380
fax_transmit_duration=0
```

debug vpm error

To enable digital signal processor (DSP) error tracing in voice port modules (VPMs), use the **debug vpm error** command in privileged EXEC mode. To disable DSP error tracing, use the **no** form of this command.

debug vpm error

no debug vpm error

Syntax Description

This command has no arguments or keywords.

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced on the Cisco 2600, 3600, and MC3810 series devices.
12.1(2)T	This command was integrated into Cisco IOS Release 12.1(2)T.

Usage Guidelines

Execution of **no debug all** will turn off all port level debugging. You should turn off all debugging and then enter the **debug** commands you are interested in one by one. This will help avoid confusion about which ports you are actually debugging.

Examples

The following example shows **debug vpm error**messages for Cisco 2600 or Cisco 3600 series router or a Cisco MC3810 series concentrator:

```
Router# debug vpm error
00:18:37:[1:0.1, FXSLS_NULL, E_DSP_SIG_0100] -> ERROR:INVALID INPUT
Router#
```

The following example turns off **debug vpm error**debugging messages:

```
Router# no debug vpm error
```

Related Commands

Command	Description
debug vpm all	Enables all VPM debugging.
debug vpm port	Limits the **debug vpm error**command to a specified port.

Command	Description
show debug	Displays which debug commands are enabled.

debug vpm port

To observe the behavior of the Holst state machine, use the **debug vpm port** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm port [*slot-number*| *subunit-number*| *port*]

no debug vpm port [*slot-number*| *subunit-number*| *port*]

Syntax Description

slot-number	(Optional) Specifies the slot number in the Cisco router where the voice interface card is installed. Valid entries are from 0 to 3, depending on the router being used and the slot where the voice interface card has been installed.
subunit-number	(Optional) Specifies the subunit on the voice interface card where the voice port is located. Valid entries are 0 or 1.
port	(Optional) Specifies the voice port. Valid entries are 0 or 1.

Command Modes

Privileged EXEC

Command History

Release	Modification
11.3(1)	This command was introduced.

Usage Guidelines

This command is not supported on Cisco 7200 series routers or on the Cisco MC3810.

Use this command to limit the debug output to a particular port. The debug output can be quite voluminous for a single channel. A 12-port box might create problems. Use this **debug** command with any or all of the other debug modes.

Execution of **no debug vpm all** will turn off all port level debugging. We recommend that you turn off all debugging and then enter the **debug** commands you are interested in one by one. This process helps to avoid confusion about which ports you are actually debugging.

Examples

The following is sample output from the **debug vpm port 1/1/0** command during trunk establishment after the **no shutdown** command has been executed on the voice port:

```
Router# debug vpm port 1/1/0
*Mar 1 03:21:39.799: htsp_process_event: [1/1/0, 0.1 , 2]act_down_inserve
```

```
*Mar 1 03:21:39.807: htsp_process_event: [1/1/0, 0.0 , 14]
   act_go_trunkhtsp_trunk_createhtsp_trunk_sig_linkfxols_trunk
*Mar 1 03:21:39.807: htsp_process_event: [1/1/0, 1.0 , 1]trunk_offhookfxols_trunk_down
*Mar 1 03:21:39.807: dsp_sig_encap_config: [1/1/0] packet_len=28 channel_id=128
   packet_id=42 transport_protocol=1 playout_delay=100 signaling_mode=0
   t_ssrc=0 r_ssrc=0 t_vpxcc=0 r_vpxcc=0
*Mar 1 03:21:39.811: dsp_set_sig_state: [1/1/0] packet_len=12
   channel_id=128 packet_id=39 state=0xC timestamp=0x0
*Mar 1 03:21:39.811: trunk_offhook: Trunk Retry Timer Enabled
*Mar 1 03:22:13.095: htsp_process_event: [1/1/0, 1.1, 39]act_trunk_setuphtsp_setup_ind
*Mar 1 03:22:13.095: htsp_process_event: [1/1/0, 1.2 , 8]
*Mar 1 03:22:13.099: hdsprm_vtsp_codec_loaded_ok: G726 firmware needs download
*Mar 1 03:22:13.103: dsp_download: p=0x60E73844 size=34182 (t=1213310):39 FA 6D
*Mar 1 03:22:13.103: htsp_process_event: [1/1/0, 1.2 , 6]act_trunk_proc_connect
*Mar 1 03:22:13.191: dsp_receive_packet: MSG_TX_RESTART_INDICATION: code=0 t=1213319
*Mar 1 03:22:13.191: dsp_download: p=0x60EA8924 size=6224 (t=1213319): 8 55 AE
*Mar 1 03:22:13.207: dsp_receive_packet: MSG_TX_RESTART_INDICATION: code=0 t=1213320
*Mar 1 03:22:13.207: htsp_process_event: [1/1/0, 1.3 , 11] trunk_upfxols_trunk_up
*Mar 1 03:22:13.207: dsp_set_sig_state: [1/1/0] packet_len=12
   channel_id=128 packet_id=39 state=0x4 timestamp=0x0
*Mar 1 03:22:13.207: dsp_sig_encap_config: [1/1/0] packet_len=28 channel_id=128
   packet_id=42 transport_protocol=3 playout_delay=100 headerbytes = 0xA0
```

Note in the above display that “transport_protocol = 3” indicates Voice-over-Frame Relay. Also note that the second line of the display indicates that a **shutdown/no shutdown** command sequence was executed on the voice port.

Related Commands

Command	Description
debug vpdn pppoe-data	Enables debugging of all VPM areas.
debug vpm dsp	Shows messages from the DSP on the VPM to the router.
debug vpm signal	Collects debug information only for signaling events.
debug vpm spi	Displays information about how each network indication and application request is handled.

debug vpm signal

To collect debug information only for signaling events, use the **debug vpm signal**command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm signal

no debug vpm signal

Syntax Description

This command has no arguments or keywords.

Command Modes

Privileged EXEC

Usage Guidelines

The **debug vpm signal** command collects debug information only for signaling events. This command can also be useful in resolving problems with signaling to a PBX.

Examples

The following output shows that a ring is detected, and that the router waits for the ringing to stop before accepting the call:

```
ssm_process_event: [1/0/1, 0.2, 15] fxols_onhook_ringing
ssm_process_event: [1/0/1, 0.7, 19] fxols_ringing_not
ssm_process_event: [1/0/1, 0.3, 6]
ssm_process_event: [1/0/1, 0.3, 19] fxols_offhook_clear
```

The following output shows that the call is connected:

```
ssm_process_event: [1/0/1, 0.3, 4] fxols_offhook_proc
ssm_process_event: [1/0/1, 0.3, 8] fxols_proc_voice
ssm_process_event: [1/0/1, 0.3, 5] fxols_offhook_connect
```

The following output confirms a disconnect from the switch and release with higher layer code:

```
ssm_process_event: [1/0/1, 0.4, 27] fxols_offhook_disc
ssm_process_event: [1/0/1, 0.4, 33] fxols_disc_confirm
ssm_process_event: [1/0/1, 0.4, 3] fxols_offhook_release
```

debug vpm signaling

To see information about the voice port module signaling, use the **debug vpm signaling** command in privileged EXEC mode. To disable debugging output, use the **no** form of this command.

debug vpm signaling

no debug vpm signaling

Syntax Description

This command has no arguments or keywords

Command Default

Disabled

Command Modes

Privileged EXEC

Command History

Release	Modification
12.0(7)XK	This command was introduced.
12.1(2)T	This command was integrated into Release 12.1(2)T.

Examples

The following is sample output from the **debug vpm signaling** command:

```
Router# debug vpm signaling
01:52:55: [1:1.1, S_TRUNK_BUSYOUT, E_HTSP_OUT_BUSYOUT]
01:52:55: htsp_timer - 0 msec
01:52:55: [1:1.1, S_TRUNK_PEND, E_HTSP_EVENT_TIMER]
01:52:55: htsp_timer_stop htsp_setup_ind
01:52:55: htsp_timer - 2000 msec
01:52:55: [1:1.1, S_TRUNK_PROC, E_HTSP_SETUP_ACK]
01:52:55: htsp_timer_stop
01:52:55: htsp_timer - 20000 msec
01:52:55: [1:6.6, S_TRUNK_PROC, E_HTSP_SETUP_ACK]
01:52:55: htsp_timer_stop
01:52:55: htsp_timer - 20000 msec
01:52:55: [1:1.1, S_TRUNK_PROC, E_HTSP_VOICE_CUT_THROUGH]
01:52:55: %HTSP-5-UPDOWN: Trunk port(channel) [1:1.1] is up
```