



Case Study: Troubleshooting Cisco IP Phone-to-Cisco IOS Gateway Calls

The case study described in [Appendix A, “Opening a Case With TAC,”](#) described the call flow for an intracluster call. The case study in this appendix examines a Cisco IP Phone that is calling through a Cisco IOS Gateway to a phone that is connected through a local PBX or on the Public Switched Telephone Network (PSTN). Conceptually, when the call reaches the Cisco IOS Gateway, the gateway will forward the call to either a phone connected to an FXS port or to the PBX. If the call is forwarded to the PBX, it could terminate to a phone that is connected to a local PBX, or the PBX forwards it over the PSTN, and the call will terminate somewhere on the PSTN.

This appendix contains the following topics:

- [Call Flow Traces](#)
- [Debug Messages and Show Commands on the Cisco IOS Gatekeeper](#)
- [Debug Messages and Show Commands on the Cisco IOS Gateway](#)
- [Cisco IOS Gateway with T1/PRI Interface](#)
- [Cisco IOS Gateway with T1/CAS Interface](#)

Call Flow Traces

This section discusses call flow through examples from the Cisco CallManager trace file CCM000000000. The traces in this case study focus only on the call flow itself because [Appendix A, “Opening a Case With TAC”](#) (for example, initialization, registration, and KeepAlive mechanism) already explained the more detailed trace information.

In this call flow, a Cisco IP Phone (directory number 1001) that is located in cluster 2 is calling a phone (directory number 3333) that is located somewhere on the PSTN. Remember that you can follow a device through the trace by looking at the TCP handle value, time stamp, or name of the device. The TCP handle value for the device remains the same until the device is rebooted or goes off line.

In the following traces, the Cisco IP Phone (1001) has gone off hook. The trace shows the unique messages, TCP handle, and the calling number, which displays on the Cisco IP Phone. No called number appears at this point, because the user has not tried to dial any digits.

```
16:05:46.37515:20:18.390 CCM|StationInit - InboundStim -
OffHookMessageID tcpHandle=0x5138d98
```

```
15:20:18.390 CCM|StationD - stationOutputDisplayText
tcpHandle=0x5138d98, Display=1001
```

In the following traces, the user is dialing the DN 3333, one digit at a time. The number 3333 specifies the destination number of the phone, which is located somewhere on the PSTN network. The digit analysis process of the Cisco CallManager is currently active and is analyzing the digits to discover where the call needs to get routed. [Appendix A, “Opening a Case With TAC”](#) provides a more detailed explanation of the digit analysis

```
15:20:18.390 CCM|Digit analysis: match(fqcn="", cn="1001", pss="",
dd="")
15:20:19.703 CCM|Digit analysis: match(fqcn="", cn="1001", pss="",
dd="3")
15:20:20.078 CCM|Digit analysis: match(fqcn="", cn="1001", pss="",
dd="33")
15:20:20.718 CCM|Digit analysis: match(fqcn="", cn="1001", pss="",
dd="333")
15:20:21.421 CCM|Digit analysis: match(fqcn="", cn="1001", pss="",
dd="3333")
15:20:21.421 CCM|Digit analysis: analysis results
```

In the following traces, the digit analysis has completed, calling and called party are matched, and the information has been parsed.

```
|CallingPartyNumber=1001
|DialingPattern=3333
|DialingRoutePatternRegularExpression=(3333)
|PretransformDigitString=3333
|PretransformPositionalMatchList=3333
|CollectedDigits=3333
|PositionalMatchList=3333
```

In the following traces, the number 0 indicates the originating location, and the number 1 indicates the destination location. BW = -1 determines the bandwidth of the originating location. The value -1 implies that the bandwidth is infinite. The bandwidth is infinite because the call originated from a Cisco IP Phone located in a LAN environment. BW = 64 determines the bandwidth of the destination location. The call destination specifies a phone located in a PSTN, and the codec type that is used is G.711 (64 Kbps).

```
15:20:21.421 CCM|Locations:Orig=0 BW=-1 Dest=1 BW=64 (-1 implies
infinite bw available)
```

The following traces show the calling and called party information. In this example, the calling party name and number are the same because the administrator has not configured a display name, such as John Smith.

```
15:20:21.421 CCM|StationD - stationOutputCallInfo
CallingPartyName=1001, CallingParty=1001, CalledPartyName=,
CalledParty=3333, tcpHandle=0x5138d98
```

The following trace shows that the H.323 code has been initialized and is sending an H.225 setup message. You can also see the traditional HDLC SAPI messages, the IP address of the called side in hexadecimal, and the port numbers.

```
15:20:21.421 CCM|Out Message -- H225SetupMsg -- Protocol= H225Protocol
15:20:21.421 CCM|MMan_Id= 1. (iep= 0 dsl= 0 sapi= 0 ces= 0
IpAddr=e24610ac IpPort=47110)
```

The following trace shows the calling and called party information as well as the H.225 alerting message. Also shown is the mapping of a Cisco IP Phone hexadecimal value to the IP address. The IP address of the Cisco IP Phone (1001) is 172.16.70.231.

```
15:20:21.437 CCM|StationD - stationOutputCallInfo
CallingPartyName=1001, CallingParty=1001, CalledPartyName=,
CalledParty=3333, tcpHandle=0x5138d98
15:20:21.453 CCM|In Message -- H225AlertMsg -- Protocol= H225Protocol
```

```
15:20:21.953 CCM|StationD - stationOutputOpenReceiveChannel
tcpHandle=0x5138d98 myIP: e74610ac (172.16.70.231)
```

The following trace shows the compression type that is used for this call (G.711 mu-law).

```
15:20:21.953 CCM|StationD - ConferenceID: 0 msecPacketSize: 20
compressionType: (4)Media_Payload_G711Ulaw64k
```

After the H.225 alert message has been sent, H.323 initializes H.245. The following trace shows the calling and called party information, and the H.245 messages. Notice that the TCP handle value remains the same as before, indicating that this is the continuation of the same call.

```
15:20:22.062 CCM|H245Interface(3) paths established ip = e74610ac,
port = 23752
15:20:22.062 CCM|H245Interface(3) OLC outgoing confirm ip = e24610ac,
port = 16758
15:20:22.062 CCM|MediaManager - wait_AuConnectInfo - received
response, forwarding
```

The following trace shows the H.225 connection message, as well as other information. When the H.225 connection message is received, the call connected.

```
15:20:22.968 CCM|In Message -- H225ConnectMsg -- Protocol=
H225Protocol
15:20:22.968 CCM|StationD - stationOutputCallInfo
CallingPartyName=1001, CallingParty=1001, CalledPartyName=,
CalledParty=3333, tcpHandle=0x5138d98
15:20:22.062 CCM|MediaCoordinator - wait_AuConnectInfoInd
15:20:22.062 CCM|StationD - stationOutputStartMediaTransmission
tcpHandle=0x5138d98 myIP: e74610ac (172.16.70.231)
15:20:22.062 CCM|StationD - RemoteIpAddr: e24610ac (172.16.70.226)
RemoteRtpPortNumber: 16758 msecPacketSize: 20
compressionType: (4)Media_Payload_G711Ulaw64k
15:20:22.062 CCM|Locations:Orig=0 BW=-1Dest=1 BW=6(-1 implies infinite
bw available)
```

The following message shows that an on-hook message from the Cisco IP Phone (1001) is being received. As soon as an on-hook message is received, the H.225 and Skinny Station device disconnection messages are sent, and the entire H.225 message is seen. This final message indicates that the call terminated.

```
15:20:27.296 CCM|StationInit - InboundStim - OnHookMessageID
tcpHandle=0x5138d98
15:20:27.296 CCM|ConnectionManager -wait_AuDisconnectRequest
(16777247,16777248): STOP SESSION
```

```
15:20:27.296 CCM|MediaManager - wait_AuDisconnectRequest - StopSession
sending disconnect to (64,5) and remove connection from list
15:20:27.296 CCM| Device SEP003094C26105 , UnRegisters with SDL Link
to monitor NodeID= 1
15:20:27.296 CCM|StationD - stationOutputCloseReceiveChannel
tcpHandle=0x5138d98 myIP: e74610ac (172.16.70.231)
15:20:27.296 CCM|StationD - stationOutputStopMediaTransmission
tcpHandle=0x5138d98 myIP: e74610ac (172.16.70.231)
15:20:28.328 CCM|In Message -- H225ReleaseCompleteMsg -- Protocol=
H225Protocol
```

Debug Messages and Show Commands on the Cisco IOS Gatekeeper

In the [Call Flow Traces](#) discussion covers the Cisco CallManager SDI trace in detail. In the topology for this case study, the debug ras command has been turned on in the Cisco IOS Gatekeeper.

The following debug messages show that the Cisco IOS Gatekeeper is receiving the admission request (ARQ) for the Cisco CallManager (172.16.70.228), followed by other successful Remote Access Server (RAS) messages. Finally, the Cisco IOS Gatekeeper sends an admission confirmed (ACF) message to the Cisco CallManager.

```
*Mar 12 04:03:57.181: RASLibRASRecvData ARQ (seq# 3365) rcvd from
[172.16.70.228883] on sock [0x60AF038C]
*Mar 12 04:03:57.181: RASLibRAS_WK_TInit ipsock [0x60A7A68C] setup
successful
*Mar 12 04:03:57.181: RASLibRAS_sendto msg length 16 from
172.16.70.2251719 to 172.16.70.228883
*Mar 12 04:03:57.181: RASLibRASSendACF ACF (seq# 3365) sent to
172.16.70.228
```

The following debug messages show that the call is in progress.

```
*Mar 12 04:03:57.181: RASLibRASRecvData successfully rcvd message of
length 55 from 172.16.70.228883
```

The following debug messages show that the Cisco IOS Gatekeeper has received a disengaged request (DRQ) from the Cisco CallManager (172.16.70.228), and the Cisco IOS Gatekeeper has sent a disengage confirmed (DCF) to the Cisco CallManager.

```
*Mar 12 04:03:57.181: RASLibRASRecvData DRQ (seq# 3366) rcvd from
[172.16.70.228883] on sock [0x60AF038C]
*Mar 12 04:03:57.181: RASLibras_sendto msg length 3 from
172.16.70.2251719 to 172.16.70.228883
*Mar 12 04:03:57.181: RASLibRASSendDCF DCF (seq# 3366) sent to
172.16.70.228
*Mar 12 04:03:57.181: RASLibRASRecvData successfully rcvd message of
length 124 from 172.16.70.228883
```

The command `show gatekeeper endpoints` on the Cisco IOS Gatekeeper shows that all four Cisco CallManagers are registered with the Cisco IOS Gatekeeper. Remember that in the topology for this case study, four Cisco CallManagers exist, two in each cluster. This Cisco IOS Gatekeeper includes two zones and each zone includes two Cisco CallManagers.

R2514-1#show gatekeeper endpoints

```

                                GATEKEEPER ENDPOINT REGISTRATION
                                =====
CallSignalAddr  Port  RASignalAddr  Port  Zone Name          Type
-----
172.16.70.228   2     172.16.70.228   1493  gka.cisco.com
VOIP-GW
H323-ID: ac1046e4->ac1046f5
172.16.70.229   2     172.16.70.229   3923  gka.cisco.com
VOIP-GW
H323-ID: ac1046e5->ac1046f5
172.16.70.245   1     172.16.70.245   1041  gkb.cisco.com
VOIP-GW
H323-ID: ac1046f5->ac1046e4
172.16.70.243   1     172.16.70.243   2043  gkb.cisco.com
VOIP-GW
H323-ID: ac1046f5->ac1046e4
Total number of active registrations = 4
```

Debug Messages and Show Commands on the Cisco IOS Gateway

[Debug Messages and Show Commands on the Cisco IOS Gatekeeper](#), the Cisco IOS Gatekeeper show commands and debug outputs discusses in detail. This section focuses on the debug output and show commands on the Cisco IOS Gateway. In the topology for this case study, calls go through the

Cisco IOS Gateways. The Cisco IOS Gateway interfaces to the PSTN or PBX with either T1/CAS or T1/PRI interfaces. The following example shows debug output of commands such as debug voip ccapi inout, debug H225 events, and debug H225 asn1.

In the following debug output, the Cisco IOS Gateway accepts the TCP connection request from Cisco CallManager (172.16.70.228) on port 2328 for H.225.

```
*Mar 12 04:03:57.169: H225Lib::h225TAccept: TCP connection accepted
from 172.16.70.228:2328 on socket [1]
*Mar 12 04:03:57.169: H225Lib::h225TAccept: Q.931 Call State is
initialized to be [Null].
*Mar 12 04:03:57.177: Hex representation of the received
TPKTO3000065080000100
```

The following debug output shows that the H.225 data is coming from the Cisco CallManager on this TCP session. Notice the protocolIdentifier, which indicates the H.323 version being used, in this debug output. The following debug shows that H.323 version 2 is being used. The example also shows the called and calling party numbers.

```
- Source Address H323-ID
- Destination Address e164
*Mar 12 04:03:57.177: H225Lib::h225RecvData: Q.931 SETUP
received from socket [1]value H323-UserInformation ::=
*Mar 12 04:03:57.181: {
*Mar 12 04:03:57.181:   h323-uu-pdu
*Mar 12 04:03:57.181:   {
*Mar 12 04:03:57.181:     h323-message-body setup :
*Mar 12 04:03:57.181:       {
*Mar 12 04:03:57.181:         protocolIdentifier { 0 0 8 2250 0 2 },
*Mar 12 04:03:57.181:         sourceAddress
*Mar 12 04:03:57.181:         {
*Mar 12 04:03:57.181:           h323-ID : "1001"
*Mar 12 04:03:57.181:         },
*Mar 12 04:03:57.185:         destinationAddress
*Mar 12 04:03:57.185:         {
*Mar 12 04:03:57.185:           e164 : "3333"
*Mar 12 04:03:57.185:         },
*Mar 12 04:03:57.189:       H225Lib::h225RecvData: State changed to
[Call Present].
```

The following debug output shows Call Control Application Programming Interface (CCAPi). Call Control APi indicates an incoming call. You can also see called and calling party information in the following output. CCAPi matches the

dial peer 0, which is the default dial peer. It matches dial peer 0 because the CCAPi could not find any other dial peer for the calling number, so it is using the default dial peer.

```
*Mar 12 04:03:57.189: cc_api_call_setup_ind (vdbPtr=0x616C9F54,
callInfo={called=3333, calling=1001, fdest=1 peer_tag=0},
callID=0x616C4838)
*Mar 12 04:03:57.193: cc_process_call_setup_ind (event=0x617A2B18)
handed call to app "SESSION"
*Mar 12 04:03:57.193: sess_appl: ev(19=CC_EV_CALL_SETUP_IND), cid(17),
disp(0)
*Mar 12 04:03:57.193: ccCallSetContext (callID=0x11,
context=0x61782BBC)
Mar 12 04:03:57.193: ssaCallSetupInd finalDest cllng(1001),
clled(3333)
*Mar 12 04:03:57.193: ssaSetupPeer cid(17) peer list: tag(1)
*Mar 12 04:03:57.193: ssaSetupPeer cid(17), destPat(3333), matched(4),
prefix(), peer(6179E63C)
*Mar 12 04:03:57.193: ccCallSetupRequest (peer=0x6179E63C, dest=,
params=0x61782BD0 mode=0, *callID=0x617A87C0)
*Mar 12 04:03:57.193: callingNumber=1001, calledNumber=3333,
redirectNumber=
*Mar 12 04:03:57.193: accountNumber=,finalDestFlag=1,
guid=0098.89c8.9233.511d.0300.cddd.ac10.46e6
```

The CCAPi matches the dial-peer 1 with the destination pattern, which is the called number 3333. Keep in mind that peer_tag means dial peer. Notice the calling and called party number in the request packet.

```
*Mar 12 04:03:57.193: peer_tag=1
*Mar 12 04:03:57.197: ccIFCallSetupRequest: (vdbPtr=0x617BE064, dest=,
callParams={called=3333, calling=1001, fdest=1, voice_peer_tag=1},
mode=0x0)
```

The following debug output shows that the H.225 alerting messages are returning to the Cisco CallManager.

```
*Mar 12 04:03:57.197: ccCallSetContext (callID=0x12,
context=0x61466B30)
*Mar 12 04:03:57.197: ccCallProceeding (callID=0x11, prog_ind=0x0)
*Mar 12 04:03:57.197: cc_api_call_proceeding(vdbPtr=0x617BE064,
callID=0x12, prog_ind=0x0)
*Mar 12 04:03:57.197: cc_api_call_alert(vdbPtr=0x617BE064,
callID=0x12, prog_ind=0x8, sig_ind=0x1)
*Mar 12 04:03:57.201: sess_appl: ev(17=CC_EV_CALL_PROCEEDING),
cid(18), disp(0)
```

```

*Mar 12 04:03:57.201: ssa:
cid(18)st(1)oldst(0)cfid(-1)csize(0)in(0)fDest(0)-cid2(17)st2(1)oldst2
(0)
*Mar 12 04:03:57.201: ssaIgnore cid(18), st(1),oldst(1), ev(17)
*Mar 12 04:03:57.201: sess_appl: ev(7=CC_EV_CALL_ALERT), cid(18),
disp(0)
*Mar 12 04:03:57.201: ssa:
cid(18)st(1)oldst(1)cfid(-1)csize(0)in(0)fDest(0)-cid2(17)st2(1)oldst2
(0)
*Mar 12 04:03:57.201: ssaFlushPeerTagQueue cid(17) peer list: (empty)
*Mar 12 04:03:57.201: ccCallAlert (callID=0x11, prog_ind=0x8,
sig_ind=0x1)
*Mar 12 04:03:57.201: ccConferenceCreate (confID=0x617A8808,
callID1=0x11, callID2=0x12, tag=0x0)
*Mar 12 04:03:57.201: cc_api_bridge_done (confID=0x7,
srcIF=0x616C9F54, srcCallID=0x11, dstCallID=0x12, disposition=0,
tag=0x0)value H323-UserInformation
*Mar 12 04:03:57.201: {
*Mar 12 04:03:57.201:   h323-uu-pdu
*Mar 12 04:03:57.201:   {
*Mar 12 04:03:57.201:     h323-message-body alerting :
*Mar 12 04:03:57.201:     {
*Mar 12 04:03:57.201:       protocolIdentifier { 0 0 8 2250 0 2 },
*Mar 12 04:03:57.205:       destinationInfo
*Mar 12 04:03:57.205:       {
*Mar 12 04:03:57.205:         mc FALSE,
*Mar 12 04:03:57.205:         undefinedNode FALSE
*Mar 12 04:03:57.205:       },

```

Notice in this packet that Cisco IOS is also sending the H.245 address and port number to Cisco CallManager. Sometimes the Cisco IOS Gateway will send the unreachable address, which could cause either no audio or one-way audio.

```

*Mar 12 04:03:57.205:   h245Address ipAddress :
*Mar 12 04:03:57.205:   {
*Mar 12 04:03:57.205:     ip 'AC1046E2'H,
*Mar 12 04:03:57.205:     port 011008
*Mar 12 04:03:57.205:   },
*Mar 12 04:03:57.213: Hex representation of the ALERTING TPKT to
send.0300003D0100
*Mar 12 04:03:57.213:
*Mar 12 04:03:57.213:   H225Lib:h225AlertRequest: Q.931 ALERTING
sent from socket [1]. Call state changed to [Call Received].
*Mar 12 04:03:57.213: cc_api_bridge_done (confID=0x7,
srcIF=0x617BE064, srcCallID=0x12, dstCallID=0x11, disposition=0,
tag=0x0)

```

The following debug output shows that the H.245 session is coming up. You can see the capability indication for codec negotiation, as well as how many bytes will be present in each voice packet.

```
*Mar 12 04:03:57.217: cc_api_caps_ind (dstVdbPtr=0x616C9F54,
dstCallId=0x11, srcCallId=0x12, caps={codec=0xEBFB, fax_rate=0x7F,
vad=0x3, modem=0x617C5720 codec_bytes=0, signal_type=3})
*Mar 12 04:03:57.217: sess_appl: ev(23=CC_EV_CONF_CREATE_DONE),
cid(17), disp(0)
*Mar 12 04:03:57.217: ssa:
cid(17)st(3)oldst(0)cfid(7)csize(0)in(1)fDest(1)-cid2(18)st2(3)oldst2(
1)
*Mar 12 04:03:57.653: cc_api_caps_ind (dstVdbPtr=0x617BE064,
dstCallId=0x12, srcCallId=0x11, caps={codec=0x1, fax_rate=0x2,
vad=0x2, modem=0x1, codec_bytes=160, signal_type=0})
```

The following debug output shows that both parties negotiated correctly and agreed on G.711 codec with 160 bytes of data.

```
*Mar 12 04:03:57.653: cc_api_caps_ack (dstVdbPtr=0x617BE064,
dstCallId=0x12, srcCallId=0x11, caps={codec=0x1, fax_rate=0x2,
vad=0x2, modem=0x1, codec_bytes=160, signal_type=0})
*Mar 12 04:03:57.653: cc_api_caps_ind (dstVdbPtr=0x617BE064,
dstCallId=0x12, srcCallId=0x11, caps={codec=0x1, fax_rate=0x2,
vad=0x2, modem=0x, codec_bytes=160, signal_type=0})
*Mar 12 04:03:57.653: cc_api_caps_ack (dstVdbPtr=0x617BE064,
dstCallId=0x12, srcCallId=0x11, caps={codec=0x1, fax_rate=0x2,
vad=0x2, modem=0x1, codec_bytes=160, signal_type=0})
*Mar 12 04:03:57.657: cc_api_caps_ack (dstVdbPtr=0x616C9F54,
dstCallId=0x11, srcCallId=0x12, caps={codec=0x1, fax_rate=0x2,
vad=0x2, modem=0x1, codec_bytes=160, signal_type=0})
*Mar 12 04:03:57.657: cc_api_caps_ack (dstVdbPtr=0x616C9F54,
dstCallId=0x11, srcCallId=0x12, caps={codec=0x1, fax_rate=0x2,
vad=0x2, modem=0x1, codec_bytes=160, signal_type=0})
```

The H.323 connect and disconnect messages follow.

```
*Mar 12 04:03:59.373: cc_api_call_connected(vdbPtr=0x617BE064,
callID=0x12)
*Mar 12 04:03:59.373: sess_appl: ev(8=CC_EV_CALL_CONNECTED), cid(18),
disp(0)
*Mar 12 04:03:59.373: ssa:
cid(18)st(4)oldst(1)cfid(7)csize(0)in(0)fDest(0)-cid2(17)st2(4)oldst2(
3)
*Mar 12 04:03:59.373: ccCallConnect (callID=0x11)
*Mar 12 04:03:59.373: {
*Mar 12 04:03:59.373:   h323-uu-pdu
*Mar 12 04:03:59.373:   {
*Mar 12 04:03:59.373:     h323-message-body connect :
```

```
*Mar 12 04:03:59.373:      {
*Mar 12 04:03:59.373:      protocolIdentifier { 0 0 8 2250 0 2 },
*Mar 12 04:03:59.373:      h245Address ipAddress :
*Mar 12 04:03:59.373:      {
*Mar 12 04:03:59.377:      ip 'AC1046E2'H,
*Mar 12 04:03:59.377:      port 011008
*Mar 12 04:03:59.377:      },
*Mar 12 04:03:59.389: Hex representation of the CONNECT TPKT to
send.03000052080
*Mar 12 04:03:59.393: H225Lib::h225SetupResponse: Q.931 CONNECT sent
from socket [1]
*Mar 12 04:03:59.393: H225Lib::h225SetupResponse: Q.931 Call State
changed to [Active].
*Mar 12 04:04:08.769: cc_api_call_disconnected(vdbPtr=0x617BE064,
callID=0x12, cause=0x10)
*Mar 12 04:04:08.769: sess_appl: ev(12=CC_EV_CALL_DISCONNECTED),
cid(18), disp(0)
```

Cisco IOS Gateway with T1/PRI Interface

As explained earlier, two types of calls go through the Cisco IOS Gateways: the Cisco IOS Gateway interfaces to the PSTN or PBX with either T1/CAS or T1/PRI interfaces. The following example shows the debug outputs when the Cisco IOS Gateways use T1/PRI interface.

The debug isdn q931 command on the Cisco IOS Gateway has been turned on, enabling Q.931, a Layer Three signaling protocol for D-channel in the ISDN environment. Each time a call is placed out of the T1/PRI interface, a setup packet must be sent. The setup packet always has (protocol descriptor) pd = 8, and it generates a random hexadecimal value for the callref. The callref tracks the call. For example, if two calls are placed, the callref value can determine the call for which the RX (received) message is intended. Bearer capability 0x8890 means a 64 Kbps data call. If it were a 0x8890218F, it would be a 56 Kbps data call and 0x8090A3 if it is a voice call. In the debug output below, the bearer capability is 0x8090A3, which is for voice. The example shows called and calling party numbers.

The callref uses a different value for the first digit (to differentiate between TX and RX) and the second value is the same (SETUP had a 0 for the last digit and CONNECT_ACK also has a 0). The router completely depends upon the PSTN or PBX to assign a Bearer channel (B-channel). If the PSTN or PBX does not assign a channel to the router, the call will not be routed. In this case, a CONNECT

message is received from the switch with the same reference number as was received for ALERTING (0x800B). Finally, you can see the exchange of the DISCONNECT message followed by RELEASE and RELEASE_COMP messages as the call is being disconnected. A cause ID for the call rejection follows RELEASE_COMP messages. The cause ID is a hexadecimal value. The meaning of the cause can be found by decoding the hexadecimal value and following up with your provider.

```
*Mar 1 225209.694 ISDN Se115 TX -> SETUP pd = 8 callref = 0x000B
*Mar 1 225209.694 Bearer Capability i = 0x8090A3
*Mar 1 225209.694 Channel ID i = 0xA98381
*Mar 1 225209.694 Calling Party Number i = 0x2183, '1001'
*Mar 1 225209.694 Called Party Number i = 0x80, '3333'
*Mar 1 225209.982 ISDN Se115 RX <- ALERTING pd = 8 callref =
0x800B
*Mar 1 225209.982 Channel ID i = 0xA98381
*Mar 1 225210.674 ISDN Se115 RX <- CONNECT pd = 8 callref = 0x800B
*Mar 1 225210.678 ISDN Se115 TX -> CONNECT_ACK pd = 8 callref =
0x000B
*Mar 1 225215.058 ISDN Se115 RX <- DISCONNECT pd = 8 callref =
0x800B
*Mar 1 225215.058 Cause i = 0x8090 - Normal call clearing
225217 %ISDN-6
DISCONNECT Int S10 disconnected from unknown , call lasted 4 sec
*Mar 1 225215.058 ISDN Se115 TX -> RELEASE pd = 8 callref = 0x000B
*Mar 1 225215.082 ISDN Se115 RX <- RELEASE_COMP pd = 8 callref =
0x800B
*Mar 1 225215.082 Cause i = 0x829F - Normal, unspecified or Special
intercept, call blocked group restriction
```

Cisco IOS Gateway with T1/CAS Interface

Two types of calls go through the Cisco IOS Gateways: the Cisco IOS Gateway interface to the PSTN or PBX with either T1/CAS or T1/PRI interfaces. The following debug outputs occur when the Cisco IOS Gateways has T1/CAS interface. The debug cas on the Cisco IOS Gateway has been turned on.

The following debug message shows that the Cisco IOS Gateway is sending an off-hook signal to the switch.

```
Apr 5 17:58:21.727: from NEAT(0): (0/15): Tx LOOP_CLOSURE (ABCD=1111)
```

The following debug message indicates that the switch is sending wink after receiving the loop closure signal from the Cisco IOS Gateway.

```
Apr 5 17:58:21.859: from NEAT(0): (0/15): Rx LOOP_CLOSURE (ABCD=1111)
Apr 5 17:58:22.083: from NEAT(0): (0/15): Rx LOOP_OPEN (ABCD=0000)
```

The following debug message indicates that the Cisco IOS Gateway is going off-hook.

```
Apr 5 17:58:23.499: from NEAT(0): (0/15): Rx LOOP_CLOSURE (ABCD=1111)
```

The following output shows the show call active voice brief on the Cisco IOS Gateway when the call is in progress. The output also shows the called and calling party number and other useful information.

```
R5300-5#show call active voice brief
<ID>: <start>hs.<index> +<connect> pid:<peer_id> <dir> <addr> <state>
tx:<packets>/<bytes> rx:<packets>/<bytes> <state>
  IP <ip>:<udp> rtt:<time>ms pl:<play>/<gap>ms
lost:<lost>/<early>/<late> delay:<last>/<min>/<max>ms <codec>
  FR <protocol> [int dlci cid] vad:<y/n> dtmf:<y/n> seq:<y/n>
sig:<on/off> <codec> (payload size)
  Tele <int>: tx:<tot>/<v>/<fax>ms <codec> noise:<l> acom:<l>
i/o:<l>/<l> dBm
511D : 156043737hs.1 +645 pid:0 Answer 1001 active
  tx:1752/280320 rx:988/158080
  IP172.16.70.228:18888 rtt:0ms pl:15750/80ms lost:0/0/0
delay:25/25/65ms g711ulaw
511D : 156043738hs.1 +644 pid:1 Originate 3333 active
  tx:988/136972 rx:1759/302548
  Tele 1/0/0 (30): tx:39090/35195/0ms g711ulaw noise:-43 acom:0
i/o:-36/-42 dBm
```

