An H.323 gateway is an endpoint on the LAN that provides real-time communications between H.323 terminals on the LAN and other ITU terminals on a WAN or to other H.323 gateways.

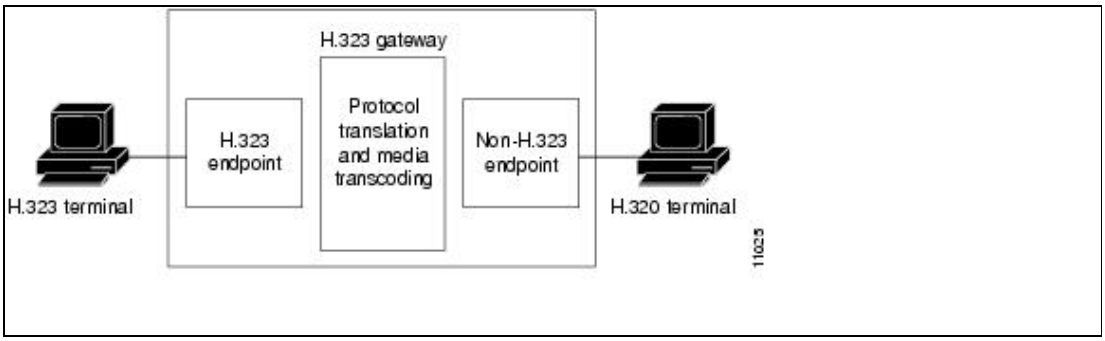
Guide Contents
Troubleshooting Cisco IOS Voice Overview
Debug Command Output on Cisco IOS Voice Gateways
Filtering Troubleshooting Output
Cisco VoIP Internal Error Codes
Troubleshooting Cisco IOS Voice Telephony
Troubleshooting Cisco IOS Voice Protocols
Troubleshooting Cisco IOS Telephony Applications
Monitoring the Cisco IOS Voice Network
Cause Codes and Debug Values

Contents

H.323 Gateway Overview

Gateways allow H.323 terminals to communicate with devices that are running other protocols. They provide protocol conversion between the devices that are running different types of protocols. For example, Figure: Gateway Between an H.323 Terminal and an H.320 Terminal shows a gateway between an H.323 terminal and a non-H.323 terminal.

Figure: Gateway Between an H.323 Terminal and an H.320 Terminal



Troubleshooting H.323 Gateway Call Routing and Dial Peers

To troubleshoot H.323 call routing, see the following sections:

Verifying Digits Received and Sent on the POTS Call Leg

Once the on-hook and off-hook signaling are verified to be working correctly, the next step in troubleshooting and debugging a VoIP call is to verify that the correct digits are being received or sent on the voice-port (digital or analog). A dial peer is not matched or the switch (CO or PBX) is not able to ring the correct station if incomplete or incorrect digits are being sent or received. Some commands that can be used to verify the digits received/sent are:

- **show dialplan number**-This command is used to show which dial peer is reached when a particular telephone number is dialed.
- **debug vtsp session**-This command displays information on how each network indication and application request is processed, signaling indications, and DSP control messages.
- **debug vtsp dsp** -This command displays the digits as they are received by the voice-port.
- **debug vtsp all**-This command enables the following debug voice telephony service provider (VTSP) commands: **debug vtsp session**, **debug vtsp error**, and **debug vtsp dsp**.

show dialplan number

The **show dialplan number** *digit_string* command displays the dial peer that is matched by a string of digits. If multiple dial peers can be matched, they are all shown in the order in which they are matched. The output of this command looks like this:

```
Router# show dialplan number 5000
Macro Exp.: 5000
VoiceOverIpPeer2
        information type = voice,
        tag = 2, destination-pattern = `5000',
        answer-address = `', preference=0,
        group = 2, Admin state is up, Operation
        state is up,
        incoming called-number = `',
        connections/maximum = 0/unlimited,
        application associated:
        type = voip, session-target =
        `ipv4:192.168.10.2',
        technology prefix:
        ip precedence = 5, UDP checksum =
        disabled, session-protocol = cisco,
        req-qos = best-effort,
        acc-qos = best-effort,
dtmf-relay = cisco-rtp,
        fax-rate = voice,
        payload size =  20 bytes
        codec = g729r8,
        payload size =  20 bytes,
        Expect factor = 10, Icpif = 30,
        signaling-type = cas,
        VAD = enabled, Poor QOV Trap = disabled,
        Connect Time = 25630, Charged Units = 0,
        Successful Calls = 25, Failed Calls = 0,
```

```
Accepted Calls = 25, Refused Calls = 0,
Last Disconnect Cause is "10  ",
Last Disconnect Text is "normal call
clearing.",
Last Setup Time = 84427934.
Matched: 5000   Digits: 4
Target: ipv4:192.168.10.2
```

debug vtsp dsp

debug vtsp dsp shows the digits as they are received by the voice port. The following output example shows the collection of DTMF digits from the DSP:

```
Router# debug vtsp dsp
Voice telephony call control dsp debugging is on
!-- ACTION: Caller picked up handset and dialed
!-- digits 5000.
!-- The DSP detects DTMF digits. Digit 5 was
!-- detected with ON time of 130msec.
*Mar 10 17:57:08.505: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=5,
*Mar 10 17:57:08.585: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=5,
duration=130
*Mar 10 17:57:09.385: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:09.485: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=150
*Mar 10 17:57:10.697: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:10.825: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=180
*Mar 10 17:57:12.865: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_BEGIN: digit=0
*Mar 10 17:57:12.917: vtsp_process_dsp_message:
MSG_TX_DTMF_DIGIT_OFF: digit=0,
duration=100
```

debug vtsp session

debug vtsp session shows the activity on the voice port. The following output example shows the handset being picked up and dial tone generated:

```
Router# debug vtsp session
Voice telephony call control session debugging is on
!--- <some output have been omitted>
!-- ACTION: Caller picked up handset.
!-- The DSP is allocated, jitter buffers, VAD
!-- thresholds, and signal levels are set.
*Mar 10 18:14:22.865: dsp_set_playout: [1/0/0 (69)]
packet_len=18 channel_id=1 packet_id=76 mode=1
initial=60 min=4 max=200 fax_nom=300
*Mar 10 18:14:22.865: dsp_echo_canceller_control:
[1/0/0 (69)] packet_len=10 channel_id=1 packet_id=66
flags=0x0
*Mar 10 18:14:22.865: dsp_set_gains: [1/0/0 (69)]
packet_len=12 channel_id=1 packet_id=91
in_gain=0 out_gain=65506
*Mar 10 18:14:22.865: dsp_vad_enable: [1/0/0 (69)]
```

```
packet_len=10 channel_id=1 packet_id=78
thresh=-38act_setup_ind_ack
*Mar 10 18:14:22.869: dsp_voice_mode: [1/0/0 (69)]
packet_len=24 channel_id=1 packet_id=73 coding_type=1
voice_field_size=80
VAD_flag=0 echo_length=64 comfort_noise=1
inband_detect=1 digit_relay=2
AGC_flag=0act_setup_ind_ack(): dsp_dtmf_mod
e()act_setup_ind_ack: passthru_mode = 0,
no_auto_switchover = 0dsp_dtmf_mode
(VTSP_TONE_DTMF_MODE)
!-- The DSP is put into "voice mode" and dial-tone is
!-- generated.
*Mar 10 18:14:22.873: dsp_cp_tone_on: [1/0/0 (69)]
packet_len=30 channel_id=1 packet_id=72 tone_id=4
n_freq=2 freq_of_first=350 freq_of_second=440 amp_of_first=
4000 amp_of_second=4000 direction=1 on_time_first=65535
off_time_first=0 on_time
_second=65535 off_time_second=0
```

If you find that the digits are not being sent or received properly, you might need to use either a digit-grabber (test tool) or T1 tester to verify the digits are being sent at the correct frequency and timing interval. If they are being sent incorrectly for the switch (CO or PBX), some values on the router or switch (CO or PBX) might need to be adjusted so that they match and can interoperate. These are usually digit duration and interdigit duration values. If the digits appear to be sent correctly you might want to examine any number translation tables in the switch (CO or PBX) that might add or remove digits.

Verifying End-to-End VoIP Signaling on the VoIP Call Leg

After verifying that voice-port signaling is working properly and the correct digits have been received, verify that the end-to-end VoIP signaling is set up on the VoIP call leg.VoIP signaling involves call control. The following factors explain why call control debugging can become a complex job:

- Cisco VoIP gateways use H.323 signaling to complete calls. H.323 is made up of three layers of call-negotiation and call-establishment: H.225, H.245, and H.323. These protocols use a combination of TCP and UDP to set up and establish a call.
- End-to-end VoIP debugging shows a number of Cisco IOS state machines, and problems with any state machine can cause a call to fail.
- End-to-end VoIP debugging can be very verbose and can create a lot of debug output.

debug voip ccapi inout

The primary command to debug end-to-end VoIP calls is **debug voip ccapi inout**. The output from a sample call debug is shown below.

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

```
bject[0x63401148], nxtElem[0x0], magic[0xFACE0FFF] tagID[5], dataLen[276], modif
[1]
*Mar  1 15:35:53.592: //-1/xxxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToContainer: Addin
g 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_labe
l= 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 , p
rot 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/xxxxxxxxxxx/CCAPI/ccTDPvtUpdateProfileTabFromContainer:
obtained key[0] for the tag[5]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxx/CCAPI/ccTDPvtAddObjectToProfileBucket: profileTable
[0x6380E11C], tdObject[0x63401148]
*Mar  1 15:35:53.600: //-1/xxxxxxxxxxx/CCAPI/ccTDPvtProfileTableBuildManager:
*Mar  1 15:35:53.600: ccTDUtilDumpAllElemInProfileTab: profileTable[0x6380E11C],
 numBuckets[11], numEntries[2]
*Mar  1 15:35:53.600: Bucket { 0 } ------> </nowiki>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 } ------> </nowiki>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/xxxxxxxxxxx/CCAPI/ccTDDestructTDUsrContainer: Container[0x638C1BF0]
*Mar  1 15:35:53.604: //-1/xxxxxxxxxxx/CCAPI/cc_incr_if_call_volume: not the VoIP or MMoIP
*Mar  1 15:35:53.608: //-1/xxxxxxxxxxx/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: >>>>CCA
PI handed cid 44 with tag 10002 to app "DEFAULT"
*Mar  1 15:35:53.608: //44/xxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(24=CC_EV_CALL_
SETUP_IND), cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxx/SSAPP:-1:-1/sess_appl: ev(SSA_EV_CALL_SE
TUP_IND), cid(44), disp(0)
*Mar  1 15:35:53.608: //44/xxxxxxxxxxx/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/xxxxxxxxxxx/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 rout
e label=, tgt route label= tg_label_flag 0x0
*Mar  1 15:35:53.608: //44/45F2AAE28044/SSAPP:10002:-1/ssaCallSetupInd: finalDes
t 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 ta
gs 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), destPat(34999),
 matched(5), prefix(), peer(637B0984), peer->encapType (2)
*Mar  1 15:35:53.612: //-1/xxxxxxxxxxx/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, prog_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_present 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,c
alling_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 la
bel tg_label_flag 0x0
```

```
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:  vdbP
tr type = 1
*Mar  1 15:35:53.620: //-1/xxxxxxxxxxxx/CCAPI/ccIFCallSetupRequestPrivate:
*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,  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 inco
ming 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, d
ialpeer_tag=10001)
*Mar  1 15:35:53.624: //45/xxxxxxxxxxxx/CCAPI/ccCallSetContext: (callID=0x2D, co
ntext=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: (v
dbPtr=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: ssaReportD
igitsDone 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=0
x10082
*Mar  1 15:35:53.628: //45/xxxxxxxxxxxx/CCAPI/cc_api_get_ic_leg_obtained_numbers
: callID=0x2D
```

The next two lines show the IP address of the terminating gateway and indicate 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 tagID[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. The line after that 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_in d=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=0x62EC61A4,
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 that message 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=0x2D,
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: v29=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=0x637EC1E0,
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=0x637EC1E0,
 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
(SSA_CS_ACTIVE),oldst(5), ev(52)
Router#
Router#! digit punched
Router#
```

The phone at the terminating gateway enters digit **1**.

```
*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: (dstVdbPtr=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 - 0x628B6
330
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=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=0x15,
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=0x15,
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:v12=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: ccFreeRawMsgInfo
(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:
ev12=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:
```

If the call is failing and the cause appears to be in the VoIP portion of the call setup, you might need to look at the H.225 or H.245 TCP part of the call setup, as opposed to just the UDP portion of the H.323 setup. The commands that can be used to debug the H.225 or H.245 call setup are:

- **debug ip tcp transaction** and **debug ip tcp packet** - These examine the TCP portion of the H.225 and H.245 negotiation. They returns the IP addresses, TCP ports and states of the TCP connections.
- **debug cch323 h225** -This examines the H.225 portion of the call negotiation. Think of this as the Layer1 part of the 3 part H.323 call setup.
- **debug ch323 h245**-This examines the H.245 portion of the call negotiation. Think of this as the Layer2 part of the 3 part H.323 call setup.

For more information about voice troubleshooting, refer to Troubleshooting and Debugging VoIP Call Basics, document 14081.

Troubleshooting H.323 Gateway Dial Tone

A common problem encountered in a VoIP network is being unable to break dial tone. The router puts a seizure on the local PBX but when digits are dialed, the dial tone stays. The calling party is unable to pass the DTMF tones or digits to the terminating device, resulting in callers being unable to dial the desired extension or interact with the device that needs DTMF tones such as a voice mail or interactive voice response (IVR) application. This problem can result from a number of reasons such as:

- DTMF tones not being passed
- DTMF tones not being understood
- DTMF tones being passed too distorted to be understood
- Other signaling and cabling issues

In the case of a VoIP call from an originating gateway (OGW) to a terminating gateway (TGW), terminating the call to a telephony device might not be understood. When you are passing DTMF tones through a compressed VoIP audio path, some or part of the dual-tones could become slightly distorted because DSP codecs are designed to interpret human speech, not machine tones. Usually, this does not occur with lesser compression codecs such as G.732 or G.711. But the higher compression codecs might result in distortion of in-band tones. However, Cisco IOS allows the DTMF tones to be passed out-of-band between VoIP

gateways using three different techniques. All of these techniques use the H.245 capabilities-exchange (part of H.323v2) to signal to the remote VoIP gateway that a DTMF-tone has been received and the remote VoIP gateway should regenerate it.

Make sure the **dtmf-relay** command is configured under the VoIP dial-peer on both sides. Three different types of DTMF relays can be configured.

```
Router(config-dial-peer)#dtmf-relay ?
  cisco-rtp Cisco Proprietary RTP
  h245-alphanumeric DTMF Relay via H245 Alphanumeric IE
  h245-signal DTMF Relay via H245 Signal IE
```

If the problem persists, try a different setting of the **dtmf-relay** command.

For more information, refer to Inability To Break Dialtone in a Voice over IP Network, document 22376.

Troubleshooting H.323 Gateway Busy Tone

This section addresses call progress in-band related issues that arise when you are interworking ISDN and H.323 signaling between VoIP and the PSTN. Challenges arise when Cisco VoIP gateways exchange signaling capabilities with the telco switch. The following are some common problem scenarios and symptoms:

- No DTMF Digits or Audio Passed on VoIP Calls to PSTN or PBX-This occurs when the IP phone user makes a call, is able to hear announcement messages (for example "enter your account number.") but cannot pass DTMF digits. This symptom applies for both VoIP toll-bypass calls and Cisco IP phone to PSTN/PBX calls.
- No Busy Tone or Announcement Message Received When Placing VoIP Outbound Calls-This occurs when Cisco IP phone (CallManager scenario) or POTS phone (VoIP toll-bypass scenario) does not hear a busy tone or announcement message from the PSTN. This symptom applies for both VoIP toll-bypass calls and Cisco IP phone to PSTN/PBX calls.
- No Busy Tone on Inbound Call from Telephony (ISDN) to Cisco CallManager IP Phone, Cisco IOS Gateway, or Third-Party H.323 Device-This occurs when a call from PSTN through gateway to a Cisco CallManager IP phone, Cisco IOS gateway or third party H.323 device does not hear busy tone when running either an application or two-stage dialing on the originating gateway.

These scenarios are described in the following sections.

No DTMF Digits or Audio Passed on VoIP Calls to PSTN or PBX

Symptom

Caller makes a call, hears announcement messages (for example "enter your account number...") but cannot pass DTMF digits. This symptom applies for both VoIP toll-bypass and Cisco IP phone calls to PSTN/PBX calls.

Problem Description

A Cisco IP phone (using Cisco CallManager) or POTS phone (VoIP) call leaves through a Cisco IOS gateway, where the called number is usually an interactive voice response (IVR). The IVR system sends back an ISDN progress message but does not connect until some account information is entered. By default, the audio path is cut through in the backward direction (toward the Cisco IP phone or originating gateway), but not in the forward direction, until the terminating gateway receives a connect message. Therefore, there is no

voice path for the transmission of DTMF tones or speech towards the terminating switch.

Solution

Configure the Cisco IOS global configuration command **voice rtp send-recv** to establish (cut through) the audio path in both directions prior to receiving an ISDN connect message from the PSTN. For more information on this command refer to the Cisco IOS Voice Command Reference.

No Busy Tone or Announcement Message Received When Placing VoIP Outbound Calls

Symptom

A Cisco IP phone (using CallManager) or POTS phone (VoIP) does not hear a busy tone or announcement message from the PSTN.

Solution

Configure the **voice call convert-discpi-to-prog** command. This command converts an inbound ISDN disconnect message with a progress indicator (PI) to an H.225 progress message with the same progress indicator (PI) value. This command can help when an announcement is played on the terminating PSTN side, but the calling party does not hear the response. In the VoIP toll-bypass scenario, most of these issues are resolved by upgrading the gateways to Cisco IOS Release 12.2(1) or later. However, if the originating device or originating ISDN switch does not keep the call active when an H.225/ISDN disconnect message is received, try using the **voice call convert-discpi-to-prog** command.

This might come up when the announcement in-band is a busy tone, as well. Beyond that, the busy signal should be provided by either the terminating device, the originating device, or the network. Some aspects of this can be controlled.

No Busy Tone on Inbound Call from Telephony (ISDN) to Cisco CallManager IP Phone, Cisco IOS Gateway, or Third-Party H.323 Device

Symptom

You cannot hear busy tone on a call from PSTN through a gateway to a Cisco CallManager IP phone, Cisco IOS gateway, or third-party H.323 device when running either an application or two-stage dialing on the originating gateway.

Solution

This can occur when the originating gateway is either running a voice application such as debit-card, or is running two-stage dialing. The latter refers to the calling party dialing the number to the gateway first, receiving the dial tone and then dialing the called party. In either case, the call has connected in terms of the PSTN network once it terminates on the originating gateway. If the IP call leg comes back with a release with the cause user-busy, the busy state cannot be indicated back towards the telephony session that is in connect state.

This problem has been addressed by having the originating gateway generate a busy tone when the release from the IP call leg is received with a cause code of user busy. The telephony leg is released either by the calling party or by the gateway after several minutes, with the cause code of normal call clearing.

For more information, refer to Troubleshooting No Busy Tone and No Announcement Messages on ISDN-VoIP (H.323) Calls, document 14066.

Troubleshooting H.323 Gateway Ringback

This section addresses call progress in-band related issues when you are interworking ISDN and H.323 signaling between VoIP and PSTN networks. Challenges arise when Cisco VoIP gateways exchange signaling capabilities with the telco switch. The following are common problem symptoms:

- No Ringback Tone on VoIP Toll-Bypass Calls-This occurs when a POTS (PSTN/PBX) user places a call (through Cisco gateways) and does not hear ringback tone before the call is answered.
- No Ringback Tone on VoIP Inbound Calls to Cisco CallManager (or Third-Party VoIP Devices) Through Cisco IOS Gateway-This occurs when a POTS (PSTN/PBX) user places a call to an IP phone (through a Cisco gateway) and does not hear ringback tone before the call is answered.
- No Ringback Tone on VoIP Outbound Calls from Cisco CallManager (or Third-Party Device) Through Cisco IOS Gateway-This occurs when a user places a call from an IP phone or third-party device to an outside number through a Cisco gateway and does not hear ringback tone.
- No Ringback to PSTN When IP Phones Initiate a Call Transfer (Cisco CallManager or Cisco Unity Voice Mail)-This occurs when an incoming call from a Cisco gateway that is transferred to Cisco CallManager or to Unity Voice Mail does not hear ringback.

No Ringback Tone on VoIP Toll-Bypass Calls

Symptom

A POTS user (from the PSTN or a PBX) places a call through a Cisco gateway and does not hear ringback tone before the call is answered.

Problem Description

The call terminating switch is sending the ringback tone, and signals a PI=8 to the terminating Cisco gateway. The PI information is then forwarded to the originating gateway via an H.225 progress message. The originating gateway is unable to decode the progress message and does not cut through the backward audio path to permit the transmission of the ringback tones. Some common scenarios for this problem are as follows:

- Terminating gateway running Cisco IOS software Release 12.1(5)T or later with originating gateway running Cisco IOS software Release 12.1T. The originating gateway does not understand the H.225 progress message and does not cut through the audio path until the connect message is received.
- Terminating Cisco gateway is connected to a CAS or analog interface and sends the PI information in an H.225 progress message to the originating gateway. The originating gateway is unable to decode the H.225 progress message.
- Third-party originating gateways and gatekeepers cannot properly parse H.225 progress messages.
- ISDN switch sends back inband ringback but Alert message does not contain a PI.

Solutions

Try the following solutions:

1. Configure the Cisco IOS global configuration command **voice call send-alert** command in the terminating gateway. This command enables the terminating gateway to send an Alert message instead of a Progress message after it receives a call setup. If this command is unavailable or does not work, go to step 2.

For more information on this command refer to the Cisco IOS Voice Command Reference..

2. Upgrade the Cisco IOS software onthe originating gateway to a current Cisco IOS software release. If this does not work, go to step 3. 3. Configure the terminating gateway to send a PI = 8 in the alert message. Configure the command **progress_ind alert enable 8** under the **voice dial-peer # pots** configuration. This command overrides the PI value received in ISDN alert message and causes the router to cut through the audio path back towards the calling party prior to connect.

For more information on this command refer to the Cisco IOS Voice Command Reference.

Note: The **progress_ind** alert and the **progress_ind setup** commands are hidden in some versions of Cisco IOS and might not be visible within CLI help. However, if the **progress_ind** progress command is available in the help parser, the above commands are also available and can be entered into the dial peer in their entirety. These commands subsequently appear in the running configuration output.

No Ringback Tone on VoIP Inbound Calls to Cisco CallManager (or Third-Party VoIP Devices) Through Cisco IOS Gateway

Symptom

A POTS user (on the PSTN or PBX) places a call to an IP phone through a Cisco gateway and does not hear ringback tone before the call is answered.

Problem Description

This commonly occurs when the inbound call does not come in to the Cisco gateway/ router with a PI=3. ISDN switches send a PI=3 in the setup message to inform the gateway that the originating call is non-ISDN and that in-band messages are expected.

Solutions

Use one of the following solutions:

- Configure the **progress_ind setup enable 3** command when configuring the VoIP dial peer in the Cisco gateway. This command forces the gateway to treat the inbound ISDN setup message as if it came in with a PI = 3 generates an in-band ringback tone towards the calling party when the H.225 alert message does not contain a PI of 1, 2, or 8.

  For more information on this command refer to the Cisco IOS Voice Command Reference.

Note: The **progress_ind alert** and the **progress_ind setup** commands are hidden in some versions of Cisco IOS and might not be visible within CLI help. However, if the **progress_ind progress** command is available in the help parser, the above commands are also available and can be entered into the dial peer in their entirety. These commands subsequently appear in the running configuration.

- An alternative to the **progress_ind setup** command is the **tone ringback alert-no-pi** command in **dial-peer voice** configuration mode. This command causes the gateway to generate ringback towards the calling party if an alert is received on the IP call leg with no PI present. It differs from the **progress_ind setup** command in that the outbound H.225 setup message does not contain a PI of 3 with the **tone ringback** command. Some devices might not accept setup messages when a PI is included.

No Ringback Tone on VoIP Outbound Calls from Cisco CallManager (or Third-Party Device) Through Cisco IOS Gateway

Symptom

A user makes an outbound call from a Cisco IP phone to the PSTN through a Cisco IOS gateway and does not hear ringback tone.

Problem Description

In this situation, the originating device expects in-band ringback tones, but either of the following might be happening:

- The PSTN or switch is not providing the ringback tone.
- The Cisco IOS gateway is not cutting through the audio to the originating device.

If the PSTN is providing in-band ringback, but the Q.931 alert message is not providing a PI indicating that there is in-band information, the gateway does not cut through the audio until the call is connected.

Solutions

Follow one of the solutions below:

- Ringback tones must come from the PSTN for trunk circuits in this situation. There are two dial-peer subcommands which may help. On the Cisco IOS gateway under the outgoing POTS dial peer, configure the command **progress_ind alert enable 8**. This command presents the Q.931 alert message to the software on the gateway as if the alert message had a PI of 8 and cut through the audio path.

Note: The **progress_ind** alert and the **progress_ind setup** commands are hidden in some versions of Cisco IOS and might not be visible within CLI help. However, if the **progress_ind** progress command is available in the help parser, the above commands are also be available and can be entered into the dial peer in their entirety. These commands subsequently appear in the running configuration.

- In Cisco IOS software releases 12.2(2)T and later, configure the command **progress_ind setup enable 3** when configuring the POTS dial peer. This command causes the gateway to send a PI with a value of 3 in the ISDN setup message, indicating to the PSTN or PBX that the originating device is a non-ISDN device and in-band information should be presented. This command should be used with the command **progress_ind alert enable 8**.
- If the PSTN device, such as an ISDN phone directly connected to a BRI port on the gateway, is not able to generate ringback in-band, the gateway can be configured to generate ringback on the IP call leg. Configure the **tone ringback alert-no-pi** command when configuring the POTS dial peer. When the ISDN alert is received with no PI present, the gateway generates the ringback and includes PI=0x8 in the H.225 alert message.

No Ringback to PSTN When IP Phones Initiate a Call Transfer (Cisco CallManager or Cisco Unity Voice Mail)

Symptom

When a call to an IP phone is answered and then transferred, the caller does not hear ringback. When the transferred call is answered, both parties are able to hear each other.

Problem Description

From the perspective of the Cisco IOS gateway, the call is completed once the call is answered by an IP phone through Cisco CallManager or Unity Voice Mail system. Any further progress tones (in case of a call transfer) should be generated by the terminating device. However, Cisco CallManager and Unity are not capable of generating the in-band progress tones.

Solution for Cisco CallManager 3.0

To solve this problem you can either check the following conditions, or configure the Cisco IOS gateway as an MGCP gateway, instead of an H.323 gateway.

- You must have Cisco CallManager Release 3.0 (8) or higher.
- From the CallManager Administration page go to the Service menu and select **Service Parameters**. Perform the following steps for each active CallManager server:
  - In the Configured Services box, select **Cisco CallManager**.
  - In the Param drop-down list box, select **ToSendH225UserInfoMsg**.
  - Set the Value drop-down list box, to T for true.
  - Upgrade the gateway to Cisco IOS Release 12.2 (2) or higher.

This problem is documented in DDTS software bug CSCds11354.

Note: These fixes are valid for ringback tones, but not for other progress tones, such as busy signal. For more information, refer to Troubleshooting No Ringback Tone on ISDN-VoIP (H.323) Calls, document 22983.

Solution for Cisco CallManager 3.3 or Newer

To solve this problem for Cisco CallManager 3.3 or newer, perform the following steps:

1. From the Cisco CallManager Administration page, go to the **Service** menu and select **Service Parameters**.
2. Select the Cisco CallManager server from the drop down box.
3. Select the system to modify from the drop down box. Make sure to select **Cisco CallManager**.
4. From the Cisco CallManager Administration page go to the **Service menu** and select **Service Parameters**.
5. From the drop-down box, select the server you wish to change.
6. In the second drop-down box, select the service:**Cisco CallManager**.
7. In the *Cluster Wide Parameters (Device - H323)'* section, under the **Send H25User Info Message selection**, verify that the box for **No Ringback** is not checked and the box for **User Infro for Ring Back Tone** is checked.

Troubleshooting H.323 Gateway One-Way or No Audio

One-way audio and no audio are problems that are fairly common during a new VoIP network installation. Most of these problems are caused by misconfigurations. This section addresses some of the common issues that are known to result in IP telephony one-way audio conversations involving Cisco gateways.

This section provides scenarios and solutions to the following problems:

- On a phone call from an IP station through a Cisco IOS voice gateway, only one party receives audio (one-way communication).

- On a toll-bypass call between two Cisco gateways, only one party receives audio (one-way communication).

The causes for one-way audio or no audio in IP telephony can be varied, however the root of the problem is usually related to IP routing issues. For one-way audio, pay attention to which direction is experiencing the audio loss. Check the following topics if you are experiencing this issue:

Ensuring IP Routing Is Enabled on Cisco IOS Gateways

Some Cisco IOS gateways, such as the VG200, have IP routing disabled by default. This leads to one-way voice problems. To enable IP routing, simply type the following global configuration command in your Cisco IOS gateway:

```
Router(config)#ip routing
```

Checking Basic IP Routing

Basic IP reachability should always be checked first. Because Real-Time Protocol (RTP) streams are connectionless (transported over UDP), traffic might travel successfully in one direction, but get lost in the opposite direction.

Once a call is established, an RTP stream carrying audio needs to flow in both directions between the endpoints. It could be that subnet A can be reached from subnet B, but subnet B cannot be reached from subnet A, so the audio stream from A to B always gets lost.

This is a basic routing issue, so use IP routing troubleshooting methods to get to the stage at which you can successfully ping phone A from gateway B. Bear in mind that ping is a bidirectional verification.

Check the following IP routing issues:

- Default gateways configured at the end stations
- IP routes on the default gateways, mentioned above, leading to the destination networks

Verify the default gateway configuration on various Cisco IP phones:

- Cisco IP Phone 7910-Press Settings, select option **6**, push volume down until the Default Router field shows up.
- Cisco IP Phone 7960/40-Press Settings, select option **3**, scroll down until the Default Router field shows up.
- Cisco IP Phone 2sp+/30vip-Press **#, then press **#** until gtwy= shows up.

Note:

If you are using the Cisco IP SoftPhone application and more than one NIC (network interface card) is installed in the box, make sure the box sources the correct NIC. This issue is often present in IP SoftPhone software Version 1.1.x.

Note: If you are using Cisco DT24+ gateways, check the DHCP scope and make sure there is a default gateway (003 router) option in the scope. The 003 router parameter is what populates the Default Gateway field in the devices and PCs. Scope option 3 should have the IP address of the router interface that is doing routing for the gateway.

Binding the H.323 Signaling to a Specific IP Address

When the Cisco IOS gateway has multiple active IP interfaces, some of the H.323 signaling might be sourced from one IP address and other parts of it might reference different source addresses. This can generate various kinds of problems, one of them being one-way audio.

To get around this problem, the H.323 signaling can be bound to a specific source address, which can belong to a physical or virtual interface (loopback). The command syntax to use under interface configuration mode is **h323-gateway voip bind srcaddr** *ip_address*. Configure this command under the interface by using the IP address that Cisco CallManager points to.

Caution: A bug exists in Cisco IOS Release 12.2(6) where this solution can actually cause a one-way audio problem. For more information, refer to bug ID CSCdw69681 in Cisco Software Bug Toolkit (registered customers only).

Checking That Answer Supervision Is Being Sent and Received Correctly From the Telco or Switch

In an implementation that has a Cisco IOS gateway connected to a telco or switch, verify that answer supervision is being sent correctly when the called device answers the call from behind the switch. Failure to receive answer supervision causes the Cisco IOS gateway to fail cut through (open) the audio path in a forward direction, causing a one-way voice. A workaround is to configure **voice rtp send-recv on**.

For more information, see the next section, Using the voice rtp send-recv Command to Establish Early Two-Way Audio.

Using the voice rtp send-recv Command to Establish Early Two-Way Audio

The voice path is established in the backward direction as soon as the RTP stream is started. The forward audio path is not cut through until the Cisco IOS gateway receives a connect message from the remote end.

In some cases it is necessary to establish a two-way audio path as soon as the RTP channel is opened, before the connect message is received. To achieve this, use the **voice rtp send-recv** global configuration command.

Checking cRTP Settings on a Link-by-Link Basis

This issue applies to scenarios, such as toll-bypass, where more than one Cisco IOS gateway is involved in the voice path and compressed RTP (cRTP) is used. cRTP, or RTP header compression, is a method for making the VoIP packet headers smaller to regain bandwidth. cRTP takes the 40-byte IP/UDP/RTP header on a VoIP packet and compresses it to 2-4 bytes per packet, yielding approximately 12 KB of bandwidth for a G.729 encoded call with cRTP. For more information on cRTP, refer to Voice Over IP - Per Call Bandwidth Consumption, document ID 7934.

cRTP is done on a hop-by-hop basis with decompression and recompression on every hop. Each packet header needs to be examined for routing, therefore cRTP needs to be enabled on both sides of an IP link.

It is also important to verify that cRTP is working as expected on both ends of the link. Cisco IOS levels vary in terms of switching paths and concurrent cRTP support.

In summary, the history of cRTP support on Cisco IOS is:

- Until Cisco IOS software Release 12.0.5T, cRTP is process-switched.
- In Cisco IOS software Release 12.0(7)T through Cisco IOS Release 12.1(1)T, fast- and Cisco-express forwarding (CEF)-switching support for cRTP are introduced.
- In Cisco IOS software Release 12.1(2)T, algorithmic performance improvements are introduced.

If you are running cRTP using Cisco IOS Release 12.1, verify that bug CSCds08210 does not affect your Cisco IOS version (VoIP and fax not working with RTP header compression on).

Verifying Minimum Software Level for NAT on Cisco IOS Gateways

If you are using Network Address Translation (NAT), the minimum software level requirements must be met. Earlier versions of NAT do not support skinny protocol translation and leads to one-way voice issues.

The minimum software levels required for using NAT and skinny simultaneously are Cisco IOS® Software 12.1(5)T for IOS gateways to support skinny and H.323v2 with NAT.

Note: If your Cisco CallManager is using a TCP port for skinny signaling different from the default port (2000), you need to adjust the NAT router with the **ip nat service skinny tcp port** *number* global configuration command.

The minimum software level required for using NAT and skinny simultaneously on a PIX firewall is Release 6.0.

Note: These levels of software do not necessarily support all of the RAS messages necessary for full gatekeeper support. Gatekeeper support is outside the scope of this document.

Disabling voice-fastpath on Cisco AS5350 and Cisco AS5400 Universal Gateways

The Cisco IOS command **voice-fastpath enable** is a hidden global configuration command for the Cisco AS5350 and Cisco AS5400. This command is enabled by default. To disable it, use the **no voice-fastpath enable** global configuration command.

When enabled, this command caches the IP address and UDP port number information for the logical channel opened for a specific call and prevents the RTP stream from getting to the application layer, but rather forwards the packets at a lower layer, which helps reduce CPU utilization marginally in high call volume scenarios.

When supplementary services, such as hold or transfer are used, the **voice-fastpath** command causes the router to stream the audio to the cached IP address and UDP port, disregarding the new logical channel information generated after a call on hold is resumed or a transfer is completed. To get around this problem, traffic should 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/UDP port pair. That is why you should disable **voice-fastpath** to support supplementary services.

Configurinng the VPN IP Address with SoftPhone

Cisco IP SoftPhone enables you to make a PC work like a Cisco IP Phone 7900 Series phone. Remote users who connect back to their company network through VPN need to configure some additional settings in order to avoid a one-way voice problem. This is a result of the media stream needing to have the endpoint of

the connection specified.

The solution is to configure the VPN IP address, instead of the IP address of the network adapter, under the Network Audio Settings.

Verifying One-Way Audio

A useful command to verify packet flow is **debug cch323 rtp**. This command displays packets transmitted (X) and received (R) by the router. An uppercase character indicates successful transmission/reception, and a lowercase character indicates a dropped packet.

```
Router# debug cch323 rtp
RTP packet tracing is enabled
Router#
!--- This is an unanswered outgoing call.
!--- Notice that voice path only cuts through in forward
!--- direction and that packets are dropped. Indeed,
!--- received packets are traffic from the IP phone to the PSTN
!--- phone. These will be dropped until the call is answered.
Mar 3 23:46:23.690: ****** cut through in FORWARD direction *****
XXXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XrXrXXrrrrrrrrrrrrrrr
Router#
Router#
!--- This is an example of an answered call:
Router#
Router#
*Mar 3 23:53:26.570: ****** cut through in FORWARD direction *****
XXXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXrXr
XXrrrrrXrXrXrXrXrXrXrXrXrXrXrXrrXXrrXrXrXrXrXXXXXXXXXXXXXXXXrXXXXXXXXrXrXrXXrrXr
XrXrXrXrXrXrXrXXrrrrrrrrrrrrrrrrrrrrrrrrrrrrr
!-- At this point the remote end picks up the phone.

*Mar 3 23:53:30.378: ****** cut through in BOTH direction *****
XRXRXRXRXRXRXRXRXXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXR
XXRRXRXRXXRRXRXRXRXRXXRXRXRXRXRXRRXRXXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXXRRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XXRRRRRRRRRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXRXR
XRXRXRXRXRXRXRXRXRXRXRXXRRRXR

!-- End of conversation.
```

Using the Test Call Feature to Verify Voice Path

The Test Call feature provides the ability for a remote station or gateway to establish a call to any destination address from a Test Call station located at a network operations center and to audibly verify the voice path:

- The system administrator can manually place a specific CIC on the target TG in test mode.
- The TCL script in OGW can specify a destination CIC or DS0 number.
- This circuit number can then be carried to the terminating gateway.

- The TGW can override any local circuit selection configured and use the specified circuit number for the outgoing call.

Information About the Test Call Feature

This section provides summary information about the test call feature. This section also describes some of the restrictions and limitations that should be taken into account when you are enabling and activating the test call feature.

Route Server

- The administrator can configure parameters, termination trunk group, and destination gateway IP address in the route server.
- The route server detects this test call by the "Test Call" source trunk group label, bypasses the normal routing logic, and routes the call directly to a specified trunk group based on the configured destination (gateway IP address plus trunk group label). All trunk group members (GW IP address plus trunk group label) in a trunk group will be displayed in the route server routing hierarchy and the test call person can select any trunk requisite.
- The group member can be specified as the destination so that the test call can be routed to a specific destination trunk group label on a specific gateway.
- A test call syslog is generated in the route server. This syslog covers the route information (incoming carrier ID/trunk group ID/trunk group number, outgoing carrier ID/trunk group ID/trunk group number), the IP address of the destination gateway, and the called number, and can be browsed from its GUI web.
- Because the ARQ message sent from the test station OGW will not have any GTD payload, the route server builds a GTD if the Termination trunk group is ISDN/ISUP/R2. The route server also sends the same format of the currently used route descriptor to the test station OGW.

TCL Script

- TCL scripts in the OGW send the route descriptor to the mediation server for accounting purposes.
- TCL scripts in the OGW override the GTD CPC value received from the route server with the configured value.

  The configured value may be changed using the **application**, **service**, and **paramspace** command structure shown in the Sample Tasks.

- TCL scripts in the OGW insert the termination CIC port number in the outgoing H225 Setup message.
- The TCL script in the OGW handles two-stage calls. If the call is originated from the test call trunk group, the TCL script in the OGW parses the dialed digits to identify the delimiter and split the dialed digits to the E164 number and CIC number.

Limitations

- Test-mode status maintained within the gateway is reset if any interface is added or removed from the trunk group.
- If the DS0 is placed in test mode, it is not selected for nontest calls. The system administrator must reset the DS0 from the test mode.
- The test mode configuration is not saved for reloads. Once the gateway reloads, the test mode configuration is lost. The administrator has to place the CIC in test mode again to place a test call.

Test Mode Limitations

- In some SS7 networks, the signaling on the terminating switch can override the local channel selection. In such cases, there is no guarantee that the test call will be placed on that specified channel.
- The number of the DS0 or CIC in test-mode status does not affect the call capacity update to the H323 client.
- When a DS0 or channel is placed in test-mode, the trunk group circuit selection algorithm avoids using the channel for outgoing nontest calls. But if call routing on the gateway is not based on the trunk group or carrier-ID, then this channel may be used. Similarly, the DS0 or channel is not guaranteed to be reserved if the gateway supports hybrid routing (some calls are based on the trunk group or carrier-ID routing and some calls not)
- Placing a DS0 or channel in test mode would only guarantee that no new outgoing calls will be placed on the DS0 in the trunk. However, no control is exercised on existing calls or incoming calls in that channel. To clear the existing call through the channel (if desired), enter this command:

  clear call voice causecode cause id callID

- All incoming calls through the channel in test mode will not be rejected. It is up to the terminating switch to correspondingly place the channel in test mode and avoid sending calls through this channel.

Feature Limitations

- The Test Call feature allows the Tcl script in the OGW to send the CIC number for Test Call to the TGW. If the corresponding DS0 is not placed in test mode, the trunk selection API would make a best-effort attempt to select the corresponding CIC or DS0 number if it is free and would continue with the test call setup.
- The CIC number is carried in H225 Setup message H323 V4 field. Thus this feature would work only in H323v4 networks,
- The test call feature is not supported in SIP and H323 v2 networks.

Test Call Command-Line Interface

To enable the test call feature, use a new CLI to place specific CIC or DS0s of a trunk group in test mode. The normal circuit selection algorithm would not select this DS0 or CIC number for nontest calls.

```
gateway#  test trunkgroup  tg-label [set-tgCic | reset-tgCic] cic-number [cpc test cpc value]
```

In the syntax, the set-tgCic keyword is used to set the test mode, and the reset-tgCic is used to reset from test mode.

To display the mapping of trunk groups to sequential CIC numbers, use the modified **show trunk group** command. A new keyword cic has been added:

```
gateway# show trunk group tg-label [cic]
```

In the syntax, the new cic keyword displays the mapping of trunk group DS0 numbers to sequential CIC numbers. A sample output follows:

```
gateway#  show trunk group 1 cic
trunk group: 1
        Description:
        trunk group label: 1
```

```
 Se3/0:23
 Timeslot  1   2   3   4   5   6   7   8   9   10   11   12
 cic       1   2   3   4   5   6   7   8   9   10   11   12
 Timeslot  13  14  15  16  17  18  19  20  21  22   23   24
 cic       13  14  15  16  17  18  19  20  21  22   23   24

 Se5/1:23
 Timeslot  1   2   3   4   5   6   7   8   9   10   11   12
 cic       25  26  27  28  29  30  31  32  33  34   35   36
 Timeslot  13  14  15  16  17  18  19  20  21  22   23   24
 cic       37  38  39  40  41  42  43  44  45  46   47   48

gateway# show trunk group
trunk group: 1
        Description:
        trunk group label: 1
        Translation profile (Incoming):
        Translation profile (Outgoing):
        Hunt Scheme is least-used
        Max Calls (Incoming):   NOT-SET (Any)   50 (Voice)      NOT-SET (Data)
        Max Calls (Outgoing):   NOT-SET (Any)   50 (Voice)      NOT-SET (Data)
        Retries: 0
        Trunk Se3/0:23  Preference DEFAULT
                Member Timeslots : 1-24
                Total channels available : 12
                Data = 0, Voice = 0, Modem = 0, Pending = 0, Free = 12
        Trunk Se5/1:23  Preference DEFAULT
                Member Timeslots : 1-24
                Total channels available : 23
                Data = 0, Voice = 0, Modem = 0, Pending = 0, Free = 23
                  Test mode = 1                  ---- new
                Testmode Timeslot(s) : 19        ---- new
        Total calls for trunk group: Data = 0, Voice = 0, Modem = 0
                                     Pend = 0, Free = 35
        advertise_flag 0x00000040, capacity timer 25 sec tripl_config_mask 0x00000000
        AC_curr 35, FD_curr 0, SD_curr 0
        succ_curr 7 tot_curr 7
        succ_report 7 tot_report 7
        changed 1 replacement position 0
```

Sample Tasks

The following is a sample set of tasks, showing the existing and new or modified CLI that enables the test call feature. This sample is provided for reference purposes only.

Originating Analog Gateway Configuration

1. Trunk group configuration:

```
!
trunk group  41001
 max-retry 1
 hunt-scheme sequential both up
!
```

2. Add trunk group to analog voice port (FXS):

```
voice-port 1/0/0
 trunk-group 41001
!
```

3. Application with kyuss script configuration:

```
application
 service mkyuss flash:app_kyuss.3_0_1.tcl
  paramspace english language en
  paramspace english index 1
  paramspace english location tftp://<tftp-server-ip>/prompts/en
  paramspace english prefix en
  param test_call_CPC 10
  param test_call_src_tg 41001
!
```

4. Dial-peer configuration:

```
dial-peer voice 1025 pots
 trunkgroup 41001
 service mkyuss
 destination-pattern 2015550100
 trunk-group-label source 41001
dial-peer voice 1000 voip
 destination-pattern 2015550...
 session target ras
 incoming called-number 2015551...
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 no vad
```

======= Terminating side ===============

1. Gateway configuration:

```
trunk group  41001
 max-retry 1
 hunt-scheme sequential both up
!
!
!
controller E1 3/0
 ds0-group 0 timeslots 1-15,17-31 type e&m-immediate-start
 cas-custom 0
 trunk-group 41001
!
dial-peer voice 1025 pots
 trunkgroup 41001
 service mkyuss
 destination-pattern 2015550100
!
dial-peer voice 1000 voip
 destination-pattern 2015550...
 session target ras
 incoming called-number 2015551...
 dtmf-relay h245-alphanumeric
 codec g711ulaw
 no vad
```

2. Put terminating CIC in test mode:

```
test trunkgroup 41001 reset_tgCic 21 cpc 10
```

3. Dial destination phone number:

```
Dialed string from orig gw analog phone : <dest_phone_num>*<cic>#
```

======= Terminating side ==========