VoIP コールの基本に対するトラブルシューティングとデバッグ

目次

概要

このドキュメントでは、VoIP ネットワークのトラブルシューティングとデバッグに関する基本テクニックとコマンドを紹介します。 Cisco ルータの音声コール フローとテレフォニー アーキテクチャの概要を紹介し、その後で、次の VoIP トラブルシューティング方法の手順を説明します。

1. デジタルとアナログのシグナリングを確認する。
2. アナログとデジタルの音声ポート経由で送受信されるディジットを確認する。
3. エンドツーエンド VoIP シグナリングを確認する。
4. VoIP Quality of Service（QoS）の問題を理解する。

5. VoIP の原因コードとデバッグ値の詳細を理解する。

注: このドキュメントは、Cisco VoIP ゲートウェイとゲートキーパーで使われる Cisco IOS® アーキテクチャのすべての側面を説明しているわけではありません。 目的は、どのコマンドが使用できるか、コマンド出力のどのフィールドが最も有益であるかを示すことです。

注意： Cisco IOS のデバッグ作業は、プロセッサに負荷をかける可能性があります。 このドキュメントで紹介するデバッグ方法を使用するときには注意が必要です。 詳細は、『debug コマンドに関する重要な情報』を参照してください。

デバッグは、ログのタイムスタンプをイネーブルにした状態で実行する必要があります。 コマンドの追加によってタイムスタンプを有効にします。 service timestamps debug datetime msec、service timestamps log datetime msec コマンドを追加します。 タイムスタンプは、状態変化の時間間隔を調べるのに役立ちます。

前提条件

要件

このドキュメントは、VoIP ネットワークの設計と導入に関係するネットワーク担当者を対象にしています。 このドキュメントの読者には、次の項目に関する知識が必要です。

- VoIP の設定
- 音声 QoS

使用するコンポーネント

このドキュメントは、特定のソフトウェアやハードウェアのバージョンに限定されるものではありません。 ただし、ここで示す出力は、Cisco IOS® ソフトウェア リリース 12.3(8) に基づくものです。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されたものです。 このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。 対象のネットワークが実稼働中である場合には、どのような作業についても、その潜在的な影響について確実に理解しておく必要があります。

表記法

ドキュメント表記の詳細は、『シスコ テクニカル ティップスの表記法』を参照してください。

ネットワーク内のコール フロー

VoIP のトラブルシューティングやデバッグを行う前に考慮しておくべき重要な要素は、VoIP コールが 3 つのコール レッグで構成されているということです。 この 3 つのコール レッグとは、送信元 Plain Old Telephone Systems（POTS; 一般電話サービス）、VoIP、宛先 POTS のことです。 これを次の図に示します。 トラブルシューティングとデバッグでは、最初に各レッグを別個に注目し、その後、VoIP コール全体を注目する必要があります。

ルータのコール フロー

次の定義では、ルータのコール フロー図に示される主なコンポーネントの機能を説明します。

コール制御 Application Programming Interface (API; アプリケーション プログラミング インターフェイス) ：3 つのクライアントがコール制御 API を利用しています。 その 3 つのクライアントとは、CLI、Simple Network Management Protocol (SNMP; 簡易ネットワーク管理プロトコル) エージェント、およびセッション アプリケーションです。 Calll Control API (CCAPI; コール制御 API) の主な機能は、次のタスクを行うことです。

- コール レッグを特定する (それがどのダイヤル ピアなのか、 どこから来たのかなど) 。
- どのセッション アプリケーションがコールを受け入れるのかを決定する (誰がそれを取り上げるのかなど) 。
- パケット ハンドラを呼び出す。
- コール レッグをともに協議する。
- コール統計情報の記録を開始する。

セッション アプリケーションとダイヤル プラン マッパー ：セッション アプリケーションは、ダイヤル プラン マッパーを使ってダイヤル ピア (ローカル POTS またはリモート VoIP) に番号をマップします。 ダイヤル プラン マッパーは、ダイヤル ピア テーブルを使ってアクティブなダイヤル ピアを検索します。

テレフォニーと VoIP Service Provider Interface (SPI; サービス プロバイダー インターフェイス) ：テレフォニー SPI は、POTS (アナログ： fxs、fxo、e&m、デジタル： isdn、qsig、e&m など) ダイヤル ピアと通信します。 VoIP SPI は、VoIP ピアに対する特定のインターフェイスです。 テレフォニー/DSP ドライバはテレフォニー SPI にサービスを配信し、一方で、VoIP SPI はセッション プロトコルに依存します。

テレフォニー インターフェイスのアーキテクチャ

この図は、Cisco ルータのテレフォニー基盤のアーキテクチャと、それらが互いにやり取りする方法を示しています。

次のリストは、上の図の主要コンポーネントの機能と定義を示しています。

- **コール制御アプリケーション プログラミング インターフェイス (CCAPI)** ：コール レッグを確立、終了、およびブリッジ処理するソフトウェア エンティティです。
- **Voice Telephony Service Provider (VTSP)** ：コール制御 API からの要求を処理し、Digital Signal Processor (DSP; デジタル シグナル プロセッサ) または VPM への適切な要求を形成する IOS プロセスです。
- **Voice Processor Module (VPM; 音声処理モジュール)** ：VPM は、テレフォニー ポート signaling state machine (SSM) 、DSP リソース マネージャ、VTSP 間のシグナリング プロセスをブリッジングして調整することを担当します。
- **DSP Resource Manager (DSPRM; DSP リソース マネージャ)** ：DSPRM は、VTSP が DSP との間でメッセージを送受信するインターフェイスを提供します。
- **パケット ハンドラ** ：パケット ハンドラは DSP とピア コール レッグとの間でパケットを転送します。
- **コール ピア** ：コール ピアは相手側のコール レッグです。 これは、別のテレフォニー音声接続 (POTS) 、VoFR、VoATM、または VoIP 接続です。

デジタルとアナログのシグナリングの確認 (POTS コール レッ

グ）

デジタルおよびアナログ シグナリングを検証する目的は次のとおりです。

- 適切なオンフックとオフフックのアナログまたはデジタルのシグナリングが受信されているかどうかを判断すること。
- 適切な E&M、FXO、および FXS シグナリングがルータとスイッチ（CO または PBX）の両方で設定されているかどうかを判断すること。
- DSP がディジット収集モードになっていることを確認すること。

これ以降のセクションで概説するコマンドは、シグナリングの確認のために使用できます。

show controllers T1 / E1（デジタル）

show controllers t1 [slot/port]：このコマンドを最初に使用します。 このコマンドは、ルータとスイッチ（CO または PBX）の間のデジタル T1 接続がアップまたはダウンしているか、正しく機能しているかを示します。 このコマンドの出力は次のようになります。

```
router# show controllers T1 1/0 T1 1/0 is up. Applique
type is Channelized T1 Cablelength is short 133 No
alarms detected. Framing is ESF, Line Code is B8ZS,
Clock Source is Line Primary. Data in current interval
(6 seconds elapsed): 0 Line Code Violations, 0 Path Code
Violations 0 Slip Secs, 0 Fr Loss Secs, 0 Line Err Secs,
0 Degraded Mins 0 Errored Secs, 0 Bursty Err Secs, 0
Severely Err Secs, 0 Unavail Secs
```

E1 を使用している場合は、show controllers e1 コマンドを使用します。 詳細については、次のサイトを参照してください。

- T1 レイヤ 1 トラブルシューティング
- T1 トラブルシューティング フローチャート
- シリアル回線の問題に関するトラブルシューティング

show voice port

show voice port slot-number/port：このコマンドを使用すると、Cisco voice interface card（VIC; 音声インターフェイス カード）の音声ポートで設定されるポート状態とパラメータが表示されます。 すべての IOS コマンドと同様、デフォルトは show running-config では何も表示されませんが、このコマンドでは表示されます。

次に示すのは、E&M 音声ポートのサンプル出力です。

```
router# show voice port 1/0:1 recEive and transMit Slot
is 1, Sub-unit is 0, Port is 1 Type of VoicePort is E&M
Operation State is DORMANT Administrative State is UP No
Interface Down Failure Description is not set Noise
Regeneration is enabled Non Linear Processing is enabled
Music On Hold Threshold is Set to -38 dBm In Gain is Set
to 0 dB Out Attenuation is Set to 0 dB Echo Cancellation
is enabled Echo Cancel Coverage is set to 16 ms
Connection Mode is normal Connection Number is not set
Initial Time Out is set to 10 s Interdigit Time Out is
set to 10 s Call-Disconnect Time Out is set to 60 s
```

```
Region Tone is set for US Voice card specific Info
Follows: Out Attenuation is Set to 0 dB Echo
Cancellation is enabled Echo Cancel Coverage is set to
16 ms Connection Mode is normal (could be trunk or plar)
Connection Number is not set Initial Time Out is set to
10 s Interdigit Time Out is set to 10 s Call-Disconnect
Time Out is set to 60 s Region Tone is set for US Voice
card specific Info Follows: Signal Type is wink-start
Operation Type is 2-wire E&M Type is 1 Dial Type is dtmf
In Seizure is inactive Out Seizure is inactive Digit
Duration Timing is set to 100 ms InterDigit Duration
Timing is set to 100 ms Pulse Rate Timing is set to 10
pulses/second InterDigit Pulse Duration Timing is set to
500 ms Clear Wait Duration Timing is set to 400 ms Wink
Wait Duration Timing is set to 200 ms Wink Duration
Timing is set to 200 ms Delay Start Timing is set to 300
ms Delay Duration Timing is set to 2000 ms Dial Pulse
Min. Delay is set to 140 ms
```

debug vpm（音声処理モジュール）

次のコマンドは、VPM テレフォニー インターフェイスをデバッグするために使用します。

- debug vpm signal：このコマンドは、シグナリング イベント用のデバッグ情報を収集するために使用し、PBX へのシグナリングに関する問題を解決する際に役に立つ可能性があります。
- debug vpm spi：このコマンドは、音声ポート モジュールのサービス プロバイダー インターフェイス（SPI）が、コール制御 API とインターフェイスを取る方法をトレースします。 このデバッグ コマンドは、各ネットワーク インジケーションおよびアプリケーション要求がどのように処理されたかを表示します。
- debug vpm dsp：このコマンドは、VPM 上の DSP からルータへのメッセージを表示します。これは、VPM が機能していないと疑われる場合に役に立つコマンドです。 これは、VPM がオフフック表示に応答しているかどうかをチェックし、インターフェイスからのシグナリング メッセージのタイミングを評価するための簡単な方法です。
- 全 debug vpm はこの EXEC コマンド debug vpm コマンドすべてを有効に します: **debug vpm spi**、**debug vpm signal**、および **debug vpm dsp**）をイネーブルにするために使用します。
- debug vpm port：このコマンドを使用すると、デバッグの出力を特定のポートに制限できます。 たとえば、次の出力は、**debug vpm dsp** メッセージをポート 1/0/0 だけに表示します。
  `debug vpm dsp`

  `debug vpm port 1/0/0` 詳細は、『VoIP デバッグ コマンド』を参照してください。

debug vpm signal コマンドの**サンプル出力**

```
maui-voip-austin#debug vpm signal !--- FXS port 1/0/0
goes from the "on-hook" to "off-hook" !--- state.
htsp_process_event: [1/0/0, 1.2 , 36]
fxsls_onhook_offhook htsp_setup_ind *Mar 10
16:08:55.958: htsp_process_event: [1/0/0, 1.3 , 8] !---
Sends ringing alert to the called phone. *Mar 10
16:09:02.410: htsp_process_event: [1/0/0, 1.3 , 10]
htsp_alert_notify *Mar 10 16:09:03.378:
htsp_process_event: [1/0/0, 1.3 , 11] !--- End of phone
call, port goes "on-hook". *Mar 10 16:09:11.966:
htsp_process_event: [1/0/0, 1.3 , 6] *Mar 10
16:09:17.218: htsp_process_event: [1/0/0, 1.3 , 28]
```

```
fxsls_offhook_onhook *Mar 10 16:09:17.370:
htsp_process_event: [1/0/0, 1.3 , 41]
fxsls_offhook_timer *Mar 10 16:09:17.382:
htsp_process_event: [1/0/0, 1.2 , 7]
fxsls_onhook_release
```

オンフックとオフフックが適切にシグナリングしない場合、次の項目を確認します。

- 配線が正しいことを確認します。
- ルータとスイッチ（CO または PBX）の両方が適切に接地されていることを確認します。
- 接続の両端のシグナリング設定が一致していることを確認する。 設定が一致していない場合は、シグナリングが不完全または一方向になる可能性があります。

E&M トラブルシューティングの詳細は、『アナログ E&M インターフェイスのタイプおよび配線の説明とトラブルシューティング』を参照してください。

debug vpm spi コマンドの**サンプル出力**

```
maui-voip-austin#debug vpm spi Voice Port Module Session
debugging is enabled !--- The DSP is put into digit
collection mode. *Mar 10 16:48:55.710:
dsp_digit_collect_on: [1/0/0] packet_len=20
channel_id=128 packet_id=35 min_inter_delay=290
max_inter_delay=3200 mim_make_time=18 max_make _time=75
min_brake_time=18 max_brake_time=75
```

受信および送信されたディジットの検証（POTS コール レッグ）

オンフックとオフフックのシグナリングが確認され、正しく動作しているのであれば、正しいディジットが音声ポート（デジタルまたはアナログ）で送受信されていることを確認します。 未完成のまたは不正なディジットが送受信された場合は、ダイヤル ピアが一致しないか、スイッチ（CO または PBX）が正しいステーションを呼び出すことができません。 受信/送信されているディジットの検証に使用できるコマンドには、次のものがあります。

- show dialplan number：このコマンドは、特定の電話番号がダイヤルされたときにどのダイヤル ピアに到達するのかを表示するのに使用されます。
- debug vtsp session：このコマンドは、各ネットワーク指示とアプリケーション要求が処理される方法、シグナリング指示、DSP 制御メッセージの情報を表示します。
- debug vtsp dsp：12.3 よりも前の Cisco IOS ソフトウェア リリースでは、このコマンドは、音声ポートが受信したディジットを表示します。 ただし、Cisco IOS ソフトウェア リリース 12.3 以降では、debug コマンドの出力にディジットは表示されなくなりました。 着信するディジットを確認するには、debug hpi detail と debug hpi notification の組み合わせを使用できます。
- 全デバッグ vtsp はこのコマンドこれらのデバッグ音声テレフォニーサービスプロバイダー（VTSP）コマンドを有効に します: debug vtsp session、debug vtsp error、debug vtsp dsp がイネーブルになります。

詳細は、『VoIP デバッグ コマンド』を参照してください。

show dialplan number

show dialplan number <digit_string>：このコマンドでは、ディジット文字列に一致するダイヤル

ピアが表示されます。 複数のダイヤル ピアが一致する場合、一致した順序ですべて表示されます。

注: T で終わる宛先パターンで一致させるには、可変長のダイヤル ピアの電話番号の最後に # 記号を使用する必要があります。

このコマンドの出力は次のようになります。

```
maui-voip-austin#show dialplan number 5000 Dial string
terminator: # 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 session

debug vtsp session コマンドは、シグナリング スタックからのシグナリング指示とアプリケーションからの要求に基づいて、ルータが DSP とやり取りする方法について情報を表示します。 この **debug** コマンドは、それぞれのネットワーク指示とアプリケーション要求が処理される方法、シグナリング指示、および DSP 制御メッセージについての情報を表示します。

```
maui-voip-austin#debug vtsp session Voice telephony call
control session debugging is on !--- Output is
suppressed. !--- 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
```

ディジットが正しく送受信されていないと判断した場合、digit-grabber（テスト ツール）または T1 テスターを使用して、ディジットが正しい頻度とタイミング間隔で送信されているかどうかを確認することが必要な可能性があります。 スイッチ（CO または PBX）に対してディジットが「不正に」送信されている場合、ルータまたはスイッチ（CO または PBX）上のいくつかの値を一致するように調整して、相互運用ができるようにする必要が出る可能性もあります。 修正する必要があるのは、通常は digit duration および inter-digit duration の値です。 ディジットが正しく送信されているかどうかを判断するもう 1 つの項目は、ディジットの追加や削除ができるスイッチ（CO または PBX）内の番号変換テーブルです。

エンドツーエンドの VoIP シグナリングの検証（VoIP コール レッグ）

音声ポートのシグナリングが適切に動作しており、正しいディジットが受信されていることを確認した後、VoIP コール制御トラブルシューティングとデバッグに進みます。 次のような理由で、コール制御デバッグが複雑な作業になる可能性があります。

- Cisco VoIP ゲートウェイは H.323 シグナリングを使用してコールを完了します。 H.323 は、コール ネゴシエーションとコール確立の 3 つのレイヤである、 H.225、H.245、H.323 から構成されています。 これらのプロトコルは、コールをセットアップおよび確立するために TCP と UDP を組み合せて使用します。
- エンドツーエンド VoIP デバッグは、いくつかの IOS state-machine を示します。 state-machine に何らかの問題があると、コールが失敗する原因になります。
- エンドツーエンドの VoIP のデバッグは非常に冗長で、大量のデバッグ出力が生成される場合があります。

debug voip ccapi inout

エンドツーエンド VoIP コールをデバッグする主なコマンドは、debug voip ccapi inout です。 あるコール デバッグからの出力を次の出力に示します。

```
!--- Action: A VoIP call is originated through the !---
Telephony SPI (pots leg) to extension 5000. !--- Some
output is omitted. maui-voip-austin#debug voip ccapi
inout voip ccAPI function enter/exit debugging is on !--
- Call leg identification, source peer: Call !---
originated from dial-peer 1 pots !--- (extension 4000).
*Mar 15 22:07:11.959: cc_api_call_setup_ind
(vdbPtr=0x81B09EFC, callInfo={called=, calling=4000,
fdest=0 peer_tag=1}, callID=0x81B628F0) !--- CCAPI
invokes the session application. *Mar 15 22:07:11.963:
cc_process_call_setup_ind (event=0x81B67E44) handed call
to app "SESSION" *Mar 15 22:07:11.963: sess_appl:
ev(23=CC_EV_CALL_SETUP_IND), cid(88), disp(0) !---
Allocate call leg identifiers "callid = 0x59" *Mar 15
22:07:11.963: ccCallSetContext (callID=0x58,
context=0x81BAF154) *Mar 15 22:07:11.963: ccCallSetupAck
(callID=0x58) !--- Instruct VTSP to generate dialtone .
```

***Mar 15 22:07:11.963: ccGenerateTone (callID=0x58** tone=8) *!--- VTSP passes digits to CCAPI.* *Mar 15 22:07:20.275:cc_api_call_digit_begin (vdbPtr=0x81B09EFC,callID=0x58,digit=5, flags=0x1, timestamp=0xC2E63BB7, expiration=0x0) *Mar 15 22:07:20.279: sess_appl: ev(10=CC_EV_CALL_DIGIT_BEGIN), cid(88), disp(0) *Mar 15 22:07:20.279: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0) *Mar 15 22:07:20.279: ssaIgnore cid(88), st(0),oldst(0), ev(10) ***Mar 15 22:07:20.327: cc_api_call_digit (vdbPtr=0x81B09EFC, callID=0x58, digit=5** , duration=100) *Mar 15 22:07:20.327: sess_appl: ev(9=CC_EV_CALL_DIGIT), cid(88), disp(0) *Mar 15 22:07:20.327: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDes t(0) *Mar 15 22:07:21.975:cc_api_call_digit_begin (vdbPtr=0x81B09EFC,callID=0x58,digit=0, flags=0x1, timestamp=0xC2E63BB7, expiration=0x0) *Mar 15 22:07:21.979: sess_appl: ev(10=CC_EV_CALL_DIGIT_BEGIN), cid(88), disp(0) *Mar 15 22:07:21.979: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDes t(0) *Mar 15 22:07:21.979: ssaIgnore cid(88), st(0),oldst(0), ev(10) ***Mar 15 22:07:22.075: cc_api_call_digit (vdbPtr=0x81B09EFC, callID=0x58, digit=0** , duration=150) *Mar 15 22:07:22.079: sess_appl: ev(9=CC_EV_CALL_DIGIT), cid(88), disp(0) *Mar 15 22:07:22.079: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0) *Mar 15 22:07:23.235: cc_api_call_digit_begin (vdbPtr=0x81B09EFC, callID=0x58, dgit=0, flags=0x1, timestamp=0xC2E63BB7, expiration=0x0) *Mar 15 22:07:23.239: sess_appl: ev(10=CC_EV_CALL_DIGIT_BEGIN), cid(88), disp(0) *Mar 15 22:07:23.239: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0) *Mar 15 22:07:23.239: ssaIgnore cid(88), st(0),oldst(0), ev(10) ***Mar 15 22:07:23.335: cc_api_call_digit (vdbPtr=0x81B09EFC, callID=0x58, digit=0** , duration=150) *Mar 15 22:07:23.339: sess_appl: ev(9=CC_EV_CALL_DIGIT), cid(88), disp(0) *Mar 15 22:07:23.339: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDes t(0) *Mar 15 22:07:25.147: cc_api_call_digit_begin (vdbPtr=0x81B09EFC, callID=0x58, d igit=0, flags=0x1, timestamp=0xC2E63BB7, expiration=0x0) *Mar 15 22:07:25.147: sess_appl: ev(10=CC_EV_CALL_DIGIT_BEGIN), cid(88), disp(0) *Mar 15 22:07:25.147: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0) *Mar 15 22:07:25.147: ssaIgnore cid(88), st(0),oldst(0), ev(10) ***Mar 15 22:07:25.255: cc_api_call_digit (vdbPtr=0x81B09EFC, callID=0x58, digit=0** , duration=160) *Mar 15 22:07:25.259: sess_appl: ev(9=CC_EV_CALL_DIGIT), cid(88), disp(0) *Mar 15 22:07:25.259: ssaTraceSct: cid(88)st(0)oldst(0)cfid(-1)csize(0)in(1)fDest(0) *!--- Matched dial-peer 2 voip. Destination number !--- 5000*
***Mar 15 22:07:25.259: ssaSetupPeer cid(88) peer list:tag(2) called number(5000)** *Mar 15 22:07:25.259: ssaSetupPeer cid(88), **destPat(5000)**, matched(4), prefix(), peer(81C04A10) *!--- Continue to call an interface and start the !--- next call leg.* ***Mar 15 22:07:25.259: ccCallProceeding (callID=0x58**, prog_ind=0x0) ***Mar 15 22:07:25.259: ccCallSetupRequest (Inbound call = 0x58, outbound peer =2, dest=, params=0x81BAF168 mode=0, *callID=0x81B6DE58) *Mar 15 22:07:25.259: callingNumber=4000, calledNumber=5000**, redirectNumber= *!--- VoIP call setup.* ***Mar 15 22:07:25.263: ccIFCallSetupRequest:** (vdbPtr=0x81A75558, dest=, **callParams={called=5000, calling=4000, fdest=0,**

```
voice_peer_tag=2}, mode=0x0) *Mar 15 22:07:25.263:
ccCallSetContext (callID=0x59, context=0x81BAF3E4) *Mar
15 22:07:25.375: ccCallAlert (callID=0x58, prog_ind=0x8,
sig_ind=0x1) !--- POTS and VoIP call legs are tied
together. *Mar 15 22:07:25.375: ccConferenceCreate
(confID=0x81B6DEA0, callID1=0x58, callI D2=0x59,
tag=0x0) *Mar 15 22:07:25.375: cc_api_bridge_done
(confID=0x1E, srcIF=0x81B09EFC, srcCall ID=0x58,
dstCallID=0x59, disposition=0, tag=0x0) !--- Exchange
capability bitmasks with remote !--- the VoIP gateway !-
-- (Codec, VAD, VoIP or FAX, FAX-rate, and so forth).
*Mar 15 22:07:26.127: cc_api_caps_ind
(dstVdbPtr=0x81B09EFC, dstCallId=0x58, src
CallId=0x59,caps={codec=0x4, fax_rate=0x2, vad=0x2,
modem=0x1 codec_bytes=20, signal_type=0}) !--- Both
gateways agree on capabilities. *Mar 15 22:07:26.127:
cc_api_caps_ack (dstVdbPtr=0x81B09EFC, dstCallId=0x58,
src CallId=0x59, caps={codec=0x4, fax_rate=0x2, vad=0x2,
modem=0x1 codec_bytes=20, signal_type=0}) *Mar 15
22:07:26.139: cc_api_caps_ack (dstVdbPtr=0x81A75558,
dstCallId=0x59, src CallId=0x58, caps={codec=0x4,
fax_rate=0x2, vad=0x2, modem=0x1 codec_bytes=20,
signal_type=0}) *Mar 15 22:07:35.259: cc_api_call_digit
(vdbPtr=0x81B09EFC, callID=0x58, digit=T , duration=0)
*Mar 15 22:07:35.259: sess_appl: ev(9=CC_EV_CALL_DIGIT),
cid(88), disp(0) *Mar 15 22:07:35.259: ssaTraceSct:
cid(88)st(4)oldst(3)cfid(30)csize(0)in(1) fDest(0)-
cid2(89)st2(4)oldst2(1) *Mar 15 22:07:35.399:
cc_api_call_connected (vdbPtr=0x81A75558, callID=0x59)
*Mar 15 22:07:35.399: sess_appl:
ev(8=CC_EV_CALL_CONNECTED), cid(89), disp(0) *Mar 15
22:07:35.399: ssaTraceSct:
cid(89)st(4)oldst(1)cfid(30)csize(0)in(0) fDest(0)-
cid2(88)st2(4)oldst2(4) !--- VoIP call is connected.
*Mar 15 22:07:35.399: ccCallConnect (callID=0x58) !---
VoIP call is disconnected. Cause = 0x10 *Mar 15
23:29:39.530: ccCallDisconnect (callID=0x5B, cause=0x10
tag=0x0)
```

コールが失敗し、その原因がコール セットアップの VoIP 部分にあるように思われる場合、H.323 セットアップの UDP 部分だけではなく、コール セットアップの H.225 または H.245 TCP 部分を確認する必要がある可能性があります。 H.225 または H.245 コール セットアップのデバッグには、次のコマンドを使用できます。

- debug ip tcp transactions と debug ip tcp packet：これらのコマンドは、H.225 と H.245 のネゴシエーションの TCP 部分を検査します。 IP アドレス、TCP ポート、TCP 接続の状態が返されます。
- debug cch323 h225：このコマンドは、コール ネゴシエーションの H.225 部分を検査して、処理されたイベントに基づいて H.225 state machine の状態遷移をトレースします。 これを H.323 コール セットアップの 3 つの部分のレイヤ 1 部分と見なします。
- debug cch323 h245：このコマンドは、コール ネゴシエーションの H.245 部分を検査して、処理されたイベントに基づいて H.245 state machine の状態遷移をトレースします。 3 人の部 H.323 コールセットアップのレイヤ2 部品としてこれについて考えて下さい。

VoIP Quality of Service（QoS）に関する問題について

VoIP コールが適切に確立されたら、次の手順は音声品質が良好であるかどうかを確認することで

す。 QoS のトラブルシューティングはこのドキュメントの対象外ですが、次のガイドラインは、良好な音声品質を実現するために考慮する必要があります。

- VoIP コールが各コーデックでどの程度帯域幅を消費するのかを理解します。 これには、レイヤ 2 と IP/UDP/RTP ヘッダーが含まれています。 詳細については、『VoIP - コール単位の帯域幅の使用量』を参照してください。
- コールが通過する IP ネットワークの特性を理解します。 たとえば、CIR のフレーム リレー ネットワークの帯域幅は、CIR を超過しているもの（またはバースト）の帯域幅とはかなり異なっています。超過する場合は、パケットがフレーム リレー クラウド内で廃棄またはキューイングされる可能性があります。 可能な限り、遅延とジッタを制御し、排除するようにします。 一方向の伝送遅延は 150 ミリ秒以下にする必要があります（G.114 勧告による）。
- キューイング手法を使用することにより、VoIP トラフィックを識別して優先順位を設定できます。
- 低速リンク経由で VoIP を転送するときには、ポイントツーポイント リンクでの Link Fragmentation and Interleaving（LFI）を備えた MLPPP、フレーム リレー リンクでの FRF.12 など、レイヤ 2 パケット フラグメンテーション テクニックの使用を検討してください。 大きなデータ パケットをフラグメント化することにより、VoIP パケットをリンク上にインターリーブできるため、VoIP トラフィックを伝送する際のジッタや遅延が減少します。
- 別のコーデックを使用したり、VAD をイネーブルおよびディセーブルにしてコールを行ったりすると、問題を IP ネットワークに対するのではなく、DSP に絞り込める可能性があります。

VoIP における QoS 問題のトラブルシューティングでは、廃棄されたパケットと、遅延およびジッタの原因となるネットワーク ボトルネックについて主に調査します。

次を調査します。

- インターフェイスの廃棄
- バッファの廃棄
- インターフェイスの輻輳
- リンク輻輳

VoIP コールのパスにあるそれぞれのインターフェイスは、検査する必要があります。 また、廃棄と輻輳を排除します。 また、ラウンドトリップ遅延はできるだけ短縮する必要があります。 VoIP エンド ポイント間で ping を実行すると、リンクのラウンドトリップ遅延がわかります。 ラウンド トリップ遅延は、可能な限り 300 ミリ秒以下にします。 遅延がやむを得ずこの値を上回る場合は、この遅延を必ず一定にして、ジッタや可変遅延を招かないようにする必要があります。

また、IOS キューイング メカニズムが VoIP パケットを適切なキューの中に配置することを確認する必要もあります。 キューイングの確認には、show queue interface や show priority などの IOS コマンドが役に立ちます。

VoIP に関する原因コードおよびデバッグ値の詳細

デバッグと、デバッグ内の関連する値を読み取るときには、次の表を使用します。

Q.931 コール接続解除の原因（debug voip ccapi inout の cause_codes）

Q.931 原因コードと値の詳細は、『ISDN スイッチ タイプ、コード、および値』を参照してください。

コール接続解除原因の値（16 進数）	意味および番号（10 進数）
CC_CAUSE_UANUM = 0x1	未割り当て番号（1）
CC_CAUSE_NO_ROUTE = 0x3	宛先への経路がない（3）
CC_CAUSE_NORM = 0x10	正常なコール クリア（16）
CC_CAUSE_BUSY = 0x11	ユーザがビジー（17）
CC_CAUSE_NORS = 0x12	ユーザからの応答がない（18）
CC_CAUSE_NOAN = 0x13	ユーザからの返答がない（19）
CC_CAUSE_REJECT = 0x15	コール拒否（21）
CC_CAUSE_INVALID_NUMBER = 0x1C	無効な番号（28）
CC_CAUSE_UNSP = 0x1F	正常、指定されていない（31）
CC_CAUSE_NO_CIRCUIT = 0x22	回線がない（34）
CC_CAUSE_NO_REQ_CIRCUIT = 0x2C	要求された回線がない（44）
CC_CAUSE_NO_RESOURCE = 0x2F	リソースがない（47）[1]
CC_CAUSE_NOSV = 0x3F	サービスまたはオプションが利用可能ではないか、指定されていません。（63）

1 この問題は、H323 セットアップ内のコーデック不一致のために発生する可能性があるので、トラブルシューティングの最初の手順は、VoIP ダイヤル ピアをハードコードして、正しいコーデックを使用することです。

コーデック ネゴシエーションの値（debug voip ccapi inout からの値）

コーデックの詳細は、『コーデックについて： 複雑度、ハードウェア サポート、MOS、およびネゴシエーション』を参照してください。

ネゴシエーション値	意味
codec=0x00000001	G711 ULAW 64K PCM
codec=0x00000002	G711 ALAW 64K PCM
codec=0x00000004	G729

codec=0x00000004	G729IETF
codec=0x00000008	G729a
codec=0x00000010	G726r16
codec=0x00000020	G726r24
codec=0x00000040	G726r32
codec=0x00000080	G728
codec=0x00000100	G723r63
codec=0x00000200	G723r53
codec=0x00000400	GSMFR
codec=0x00000800	G729b
codec=0x00001000	G729ab
codec=0x00002000	G723ar63
codec=0x00004000	G723ar53
codec=0x00008000	CLEAR_CHANNEL

トーン タイプ

トーン タイプ	意味
CC_TONE_RINGBACK 0x1	呼び出しトーン
CC_TONE_FAX 0x2	FAX トーン
CC_TONE_BUSY 0x4	ビジー トーン
CC_TONE_DIALTONE 0x8	ダイヤル トーン
CC_TONE_OOS 0x10	アウト オブ サービス トーン
CC_TONE_ADDR_ACK 0x20	アドレス確認応答トーン
CC_TONE_DISCONNECT 0x40	切断トーン
CC_TONE_OFF_HOOK_NOTICE 0x80	電話がオフフックのままだったことを示すトーン
CC_TONE_OFF_HOOK_ALERT 0x100	CC_TONE_OFF_HOOK_NOTICE のさらに緊急なバージョン
CC_TONE_CUSTOM 0x200	カスタム トーン：カスタム トーンを指定するときに使用される
CC_TONE_NULL 0x0	ヌル トーン

FAX 速度および VAD 機能の値

値	意味
CC_CAP_FAX_NONE 0x1	FAX が無効または利用不可能
CC_CAP_FAX_VOICE 0x2	音声コール
CC_CAP_FAX_144 0x4	14,400 ボー

CC_CAP_FAX_96 0x8	9,600 ボー
CC_CAP_FAX_72 0x10	7,200 ボー
CC_CAP_FAX_48 0x20	4,800 ボー
CC_CAP_FAX_24 0x40	2,400 ボー
CC_CAP_VAD_OFF 0x1	VAD が無効
CC_CAP_VAD_ON 0x2	VAD が有効

関連情報

- ダイヤル プラン、ダイヤル ピア、およびディジット操作の設定
- VoIP デバッグ コマンド
- T1 レイヤ 1 トラブルシューティング
- T1 トラブルシューティング フローチャート
- シリアル回線の問題に関するトラブルシューティング
- 音声に関する技術サポート
- 音声とユニファイド コミュニケーションに関する製品サポート
- Cisco IP Telephony のトラブルシューティング
- テクニカルサポート - Cisco Systems