排查VoIP呼叫故障並對其進行調試

目錄

簡介

本文檔介紹對VoIP網路進行故障排除和調試的基本技術和命令。

必要條件

需求

思科建議您瞭解以下主題：

- VoIP配置

- 語音Qos
- VoIP網路的設計和部署

採用元件

本文件所述內容不限於特定軟體和硬體版本。但是，顯示的輸出基於Cisco IOS®軟體版本12.3(8)。

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路運作中，請確保您瞭解任何指令可能造成的影響。

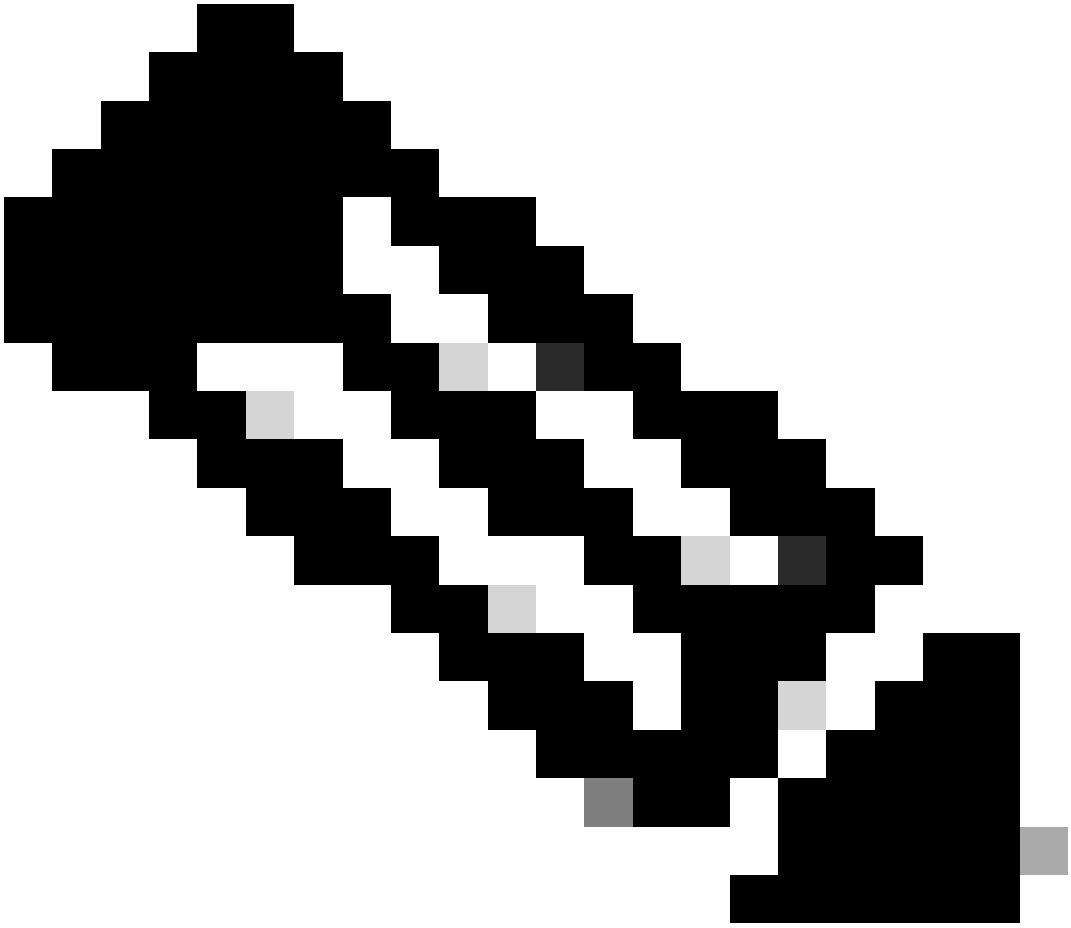
慣例

如需文件慣例的詳細資訊，請參閱思科技術提示慣例。

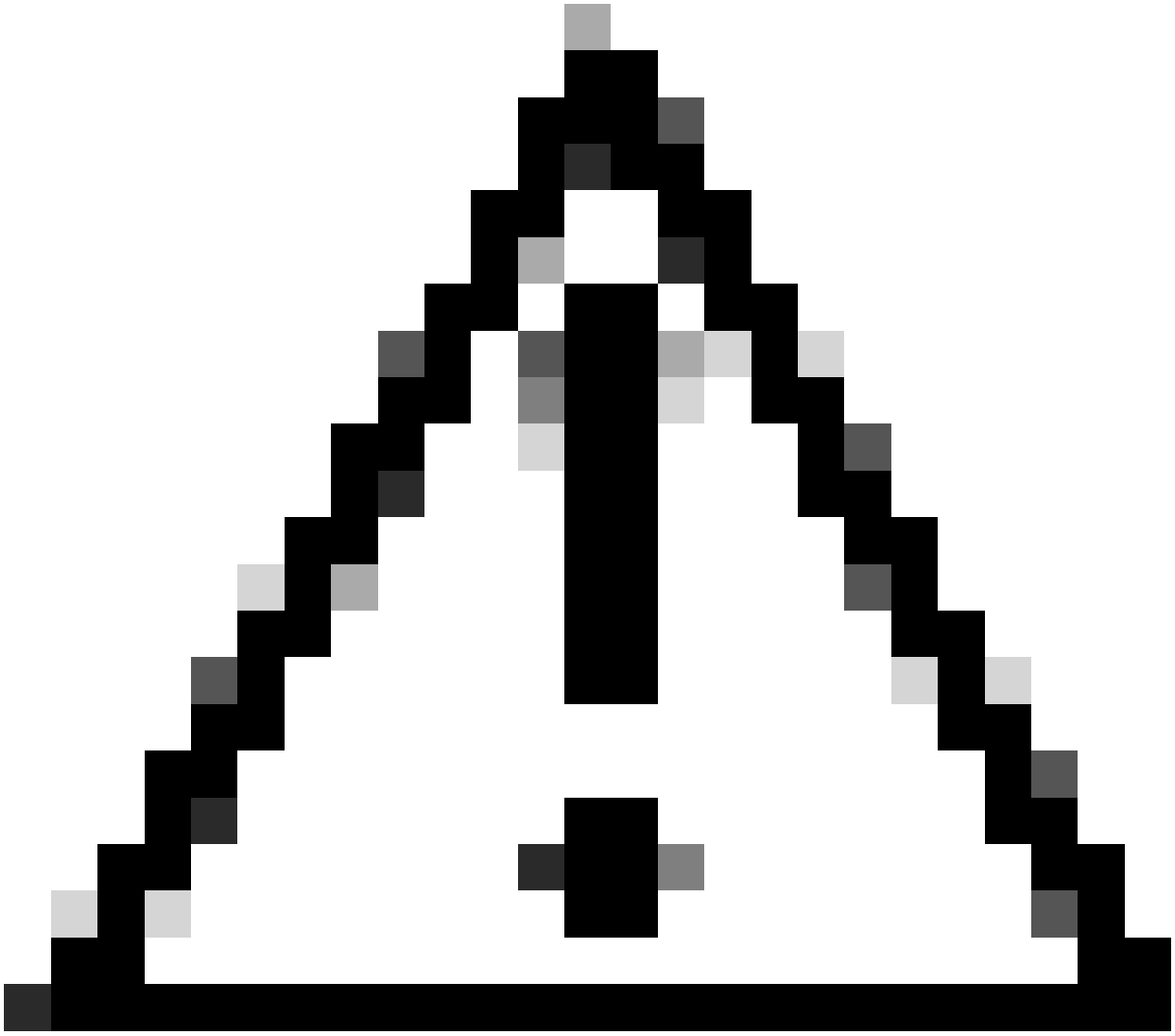
背景資訊

本文件說明進行 VoIP 網路疑難排解和偵錯時，所使用的基本技巧和指令。下列步驟概略說明了思科路由器中的語音通話流程與電話架構，以及逐步進行 VoIP 疑難排解的作法：

1. 驗證數字和模擬信令。
2. 驗證從模擬和數字語音埠接收和傳送的數字。
3. 檢驗端到端VoIP信令。
4. 瞭解VoIP服務品質(QoS)問題。
5. 瞭解VoIP的原因代碼和調試值的詳細資訊。



注意：本文檔不介紹Cisco VoIP網關和網守中使用的Cisco IOS體系結構的各個方面。相反，其目的是顯示哪些命令可用，以及命令輸出中的哪些欄位可能最有價值。



注意：調試Cisco IOS需要佔用大量處理器。使用本文檔中列出的調試時，應特別謹慎。有關詳細資訊，請參閱有關Debug命令的重要資訊。

需要在日誌中啟用時間戳的情況下運行調試。在啟用模式下，使用命令service timestamps debug datetime msec和service timestampslog datetime msec啟用時間戳。時間戳有助於確定狀態更改之間的時間間隔。

網路中的呼叫流

在開始任何VoIP故障排除或調試之前，需要考慮的一個重要因素是VoIP呼叫由三個呼叫段組成。這些呼叫段包括源普通舊式電話系統(POTS)、VoIP和目的POTS。如下圖所示。故障排除和調試需要首先分別關注每個支路，然後關注整個VoIP呼叫。

路由器呼叫流

以下定義解釋了路由器呼叫流程圖中顯示的主要元件的功能：

呼叫控制API（應用程式程式設計介面）— 三個客戶端使用呼叫控制API。三個客戶端是CLI、簡單網路管理協定(SNMP)代理和會話應用。呼叫控制API（也稱為CCAPI）的主要功能是：

- 辨識呼叫段（例如，哪個撥號對等體？它來自何處？）。
- 決定哪個會話應用程式接收呼叫（例如，由誰處理呼叫？）。
- 呼叫資料包處理程式。
- 將呼叫段連線起來。
- 開始記錄呼叫統計資訊。

Session Application and Dial Plan Mapper— 會話應用程式使用撥號計畫對映器將號碼對映到撥號對等體（本地POTS或遠端VoIP）。撥號方案對映器使用撥號對等體表查詢活動的撥號對等體。

電話和VoIP服務提供商介面(SPI)— 電話SPI與POTS（模擬：fxs、fxo、e&m數字：isdn、qsig、e&m等）撥號對等體通訊。VoIP SPI是VoIP對等體的特定介面。電話/DSP驅動程式向電話SPI提供服務，而VoIP SPI依賴於會話協定。

電話介面架構

此圖顯示Cisco路由器電話構建塊的架構，以及它們如何相互互動。

此清單說明主要圖表元件的功能和定義：

- 呼叫控制應用程式設計介面(CCAPI) -用於建立、終止和橋接呼叫段的軟體實體。
- 語音電話服務提供商(VTSP)— 一個Cisco IOS進程，為來自呼叫控制API的請求提供服務，並將適當的請求傳達給數位訊號處理器(DSP)或VPM。
- 語音處理器模組(VPM)—VPM負責在電話埠信令狀態機(SSM)、DSP資源管理器和VTSP之間橋接並協調信令處理。
- DSP資源管理器—DSPRM提供介面，由VTSP用來向DSP傳送消息或從DSP接收消息。
- 資料包處理程式— 資料包處理程式在DSP和對等呼叫段之間轉發資料包。
- 呼叫對等體— 呼叫對等體是相反的呼叫段。這可以是另一個電話語音連線(POTS)、VoFR、VoATM或VoIP連線。

驗證數字和模擬信令（POTS呼叫段）

驗證數字和模擬信令的方法是：

- 確定已接收正確的掛機和摘機模擬或數字信令。
- 確定路由器和交換機（CO或PBX）上是否配置了正確的E&M、FXO和FXS信令。
- 驗證DSP是否處於數字收集模式。

以下各節中概述的命令可用於驗證信令。

show controllers T1 / E1 （數位）

show controllers t1 [slot/port]—首先使用此命令。它顯示路由器和交換機（CO或PBX）之間的數字T1連線是啟動還是關閉，以及它是否正常工作。此命令的輸出如下所示：

```
<#root>

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
```

如果使用E1，請使用show controllers e1命令。如需詳細資訊，請參閱：

- T1第1層故障排除
- T1故障排除流程圖
- 排除串列線路問題

show voice port

show voice portslot-number/port— 使用此命令顯示Cisco語音介面卡(VIC)的埠狀態和語音埠上配置的引數。與所有Cisco IOS命令一樣，show running-config不顯示預設值，而此命令顯示預設值。

以下是E&M語音埠的輸出示例：

```
<#root>

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（語音處理器模組）

以下命令用於調試VPM電話介面：

- debug vpm signal— 此命令用於收集信令事件的調試資訊，有助於解決PBX信令問題。
- debug vpm spi— 此命令跟蹤語音埠模組服務提供商介面(SPI)如何與呼叫控制API進行通訊。此debug命令顯示有關如何處理每個網路指示和應用程式請求的資訊。
- debug vpm dsp— 此命令顯示從VPM上的DSP傳送到路由器的消息，在您懷疑VPM不工作時很有幫助。這是一個簡單的檢查VPM是否響應摘機指示和評估來自介面的信令消息的計時的方法。
- debug vpm all— 此EXEC命令啟用了所有debug vpm命令：debug vpm spi、debug vpm signal和debug vpm dsp。
- debug vpm port— 使用此命令將調試輸出限制到特定埠。例如，此輸出僅顯示埠1/0/0的debug vpm dspmessages：

```
debug vpm dsp

debug vpm port 1/0/0
```

debug vpm signalCommand的輸出示例

```
<#root>

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
```

如果掛機和摘機未正確發出訊號，請檢查以下專案：

- 驗證佈線是正確的。
- 驗證路由器和交換機（CO或PBX）是否都已正確接地。
- 檢驗連線的兩端是否具有等效的信令配置。不匹配的配置可能導致不完整或單向信令。

有關E&M故障排除的詳細資訊，請參閱瞭解模擬E & M介面型別和佈線並對其進行故障排除。

debug vpm spi命令的輸出示例

```
<#root>

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
```

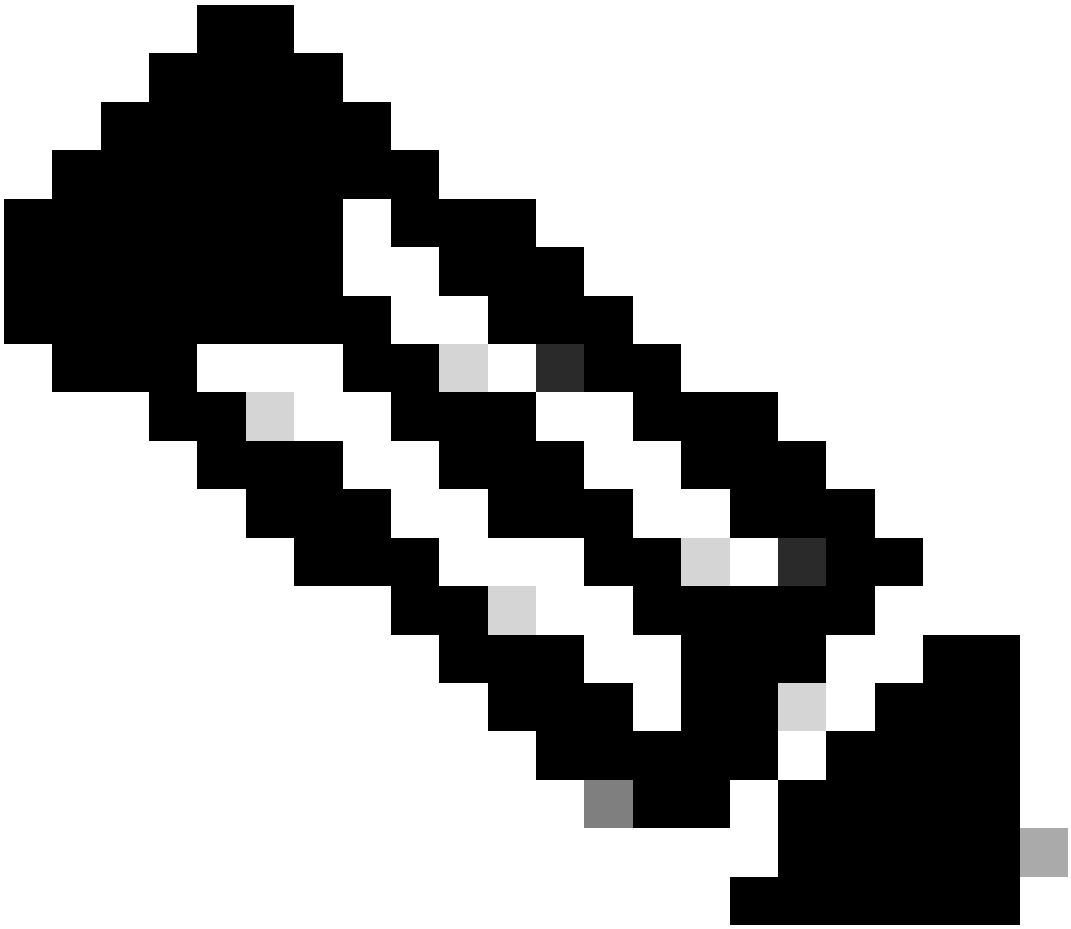
驗證接收和傳送的數字（POTS呼叫段）

驗證掛機和摘機信令並正確工作後，驗證語音埠（數字或模擬）上接收或傳送了正確的數字。如果傳送或接收的數字不完整或不正確，則撥號對等體不匹配或交換機（CO或PBX）無法撥打正確的電站。可用於驗證接收/傳送數字的一些命令包括：

- show dialplan number— 此命令用於顯示在撥打特定的電話號碼後到達哪個撥號對等體。
- debug vtsp session— 此命令顯示有關如何處理每個網路指示和應用程式請求的資訊、信令指示以及DSP控制消息。
- debug vtsp dsp— 在Cisco IOS軟體版本12.3以前，此命令在語音埠收到數字後將數字顯示出來。但是，在Cisco IOS軟體版本12.3及以後，debug命令的輸出不再顯示數字。可結合使用debug hpi detail和debug hpinotification來檢視傳入的數字。
- debug vtsp all— 此命令啟用了以下debug voice telephony service provider (VTSP)命令：debug vtsp session、debug vtsp error和debug vtsp dsp。

show dialplan number

show dialplan number <digit_string>— 此命令顯示數字字串所匹配的撥號對等體。如果可以匹配多個撥號對等體，則它們都按照匹配的順序顯示。



注意：對於長度可變的撥號對等體，您需要在電話號碼末尾使用#號，以便與以T結尾的目

標模式匹配。

此命令的輸出如下所示：

```
<#root>

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命令顯示有關路由器如何根據來自信令堆疊的信令指示以及來自應用程式的請求與DSP進行互動的資訊。此debug命令顯示有關如何處理每個網路指示和應用程式請求的資訊、信令指示以及DSP控制消息。

```
<#root>

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=-38

act_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
```

如果確定沒有正確傳送或接收數字，則可能有必要使用數字捕獲器（測試工具）或T1測試器來驗證以正確的頻率和時間間隔傳送數字。如果針對交換器（CO或PBX）以「不正確」傳送這些值，則可能需要調整路由器或交換器（CO或PBX）上的某些值，以使它們相符且可互操作。這些通常是數字持續時間和數字間持續時間的值。檢查數字是否正確傳送的另一個專案是交換機（CO或PBX）中可增加或刪除數字的任何號碼轉換表。

驗證端對端VoIP訊號傳送（VOIP呼叫段）

在驗證語音埠信令工作正常且接收到正確數字後，轉到VoIP呼叫控制故障排除和調試。這些因素解釋了為什麼呼叫控制調試會成為一項複雜的工作：

- Cisco VoIP網關使用H.323信令完成呼叫。H.323由呼叫協商和呼叫建立三層組成：H.225、H.245和H.323。這些協定結合使用TCP和UDP來設定和建立呼叫。
- 端到端VoIP調試顯示許多Cisco IOS狀態機。任何狀態機發生問題都可能導致呼叫失敗。
- 端到端VoIP調試可能非常詳細，並會建立大量調試輸出。

debug voip ccapi inout

用來調試端到端VoIP呼叫的主要命令是debug voip ccapi inout。呼叫調試的輸出如下所示。

```
<#root>

!--- 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)
```

如果呼叫失敗且原因似乎出現在呼叫設定的VoIP部分，則可能需要檢視呼叫設定的H.225或H.245 TCP部分，而不僅僅是H.323設定的UDP部分。可用於調試H.225或H.245呼叫建立的命令有：

- debug ip tcp transactions和debug ip tcp packet— 這些命令檢查H.225和H.245協商的TCP部

分。它們返回IP地址、TCP埠和TCP連線的狀態。

- debug cch323 h225— 此命令檢查呼叫協商的H.225部分，並根據已處理的事件來跟蹤H.225狀態機的狀態轉換。將其視為三部分H.323呼叫設定的第1層。
- debug cch323 h245— 此命令檢查呼叫協商的H.245部分，並根據已處理的事件來跟蹤H.245狀態機的狀態轉換。將其視為三部分H.323呼叫設定的第2層。

瞭解VoIP服務品質(QoS)問題

當VoIP呼叫正確建立後，下一步是驗證語音品質是否良好。雖然本文檔未涉及QoS故障排除，但需要考慮以下準則以實現良好的語音品質：

- 瞭解VoIP呼叫在每個編解碼器上消耗的頻寬。這包括第2層和IP/UDP/RTP報頭。有關詳細資訊，請參閱修改語音呼叫的頻寬佔用量計算。
- 瞭解呼叫經過的IP網路的特徵。例如，幀中繼網路在CIR時的頻寬與CIR以上（或突發量）的頻寬大不相同，在幀中繼網雲中，資料包可能會被丟棄或排隊。確保儘可能控制和消除延遲和抖動。單向傳輸延遲不得超過150毫秒（根據G.114建議）。
- 使用隊列技術來標識和優先處理VoIP流量。
- 當您透過低速連結傳輸VoIP時，請使用第2層封包分段技術，例如點對點連結上使用含連結分段和交錯(LFI)的MLPPP，或訊框中繼連結上使用FRF.12。對較大資料包進行分段可減少VoIP流量傳輸過程中的抖動和延遲，因為VoIP資料包可以交替傳送到鏈路上。
- 請嘗試使用其他編解碼器，並嘗試在啟用和停用VAD的情況下進行呼叫，以便將問題範圍縮小到DSP（而不是IP網路）。

透過VoIP，在排除QoS故障時需要注意的主要事項是丟棄的資料包和可能導致延遲和抖動的網路瓶頸。

尋找：

- 介面捨棄
- 緩衝區丟棄
- 介面擁塞
- 鏈路擁塞

需要檢查VoIP呼叫路徑中的每個介面。此外，還可以消除丟棄和擁塞。此外，需要儘可能減少往返延遲。VoIP終端之間的ping可以指示鏈路的往返延遲。儘可能的往返延遲不能超過300毫秒。如果延遲確實必須超過此值，則需要努力確保此延遲是恆定的，以便不會引入抖動或可變延遲。

還必須進行驗證，以確保Cisco IOS排隊機制將VoIP資料包放置在正確的隊列中。Cisco IOS命令，例如show queue interface或show priority有助於對排隊進行驗證。

VoIP原因代碼和調試值的詳細資訊

當您讀取調試以及調試中的關聯值時，請使用這些表。

Q.931呼叫斷開原因（debug voip ccapi inout的cause_codes）

呼叫斷開原因值（十六進位制）	意義與數字（以十進位為單位）
CC_CAUSE_UANUM = 0x1	未分配號碼。(1)
CC_CAUSE_NO_ROUTE = 0x3	沒有通往目的地的路由。(3)
CC_CAUSE_NORM = 0x10	正常呼叫清除。(16)
CC_CAUSE_BUSY = 0x11	使用者忙。(17)
CC_CAUSE_NORS = 0x12	無使用者回應。(18)
CC_CAUSE_NOAN = 0x13	無使用者應答。(19)
CC_CAUSE_REJECT = 0x15	呼叫被拒絕。(21)
CC_CAUSE_INVALID_NUMBER = 0x1C	無效號碼。(28)
CC_CAUSE_UNSP = 0x1F	正常，未指定。(31)
CC_CAUSE_NO_CIRCUIT = 0x22	沒有電路。(34)
CC_CAUSE_NO_REQ_CIRCUIT = 0x2C	沒有要求的電路。(44)
CC_CAUSE_NO_RESOURCE = 0x2F	無資源。(47)[1]
CC_CAUSE_NOSV = 0x3F	服務或選項不可用，或未指定。(63)

[1]此問題可能由於H323設定中的編解碼器不匹配而發生，因此第一個故障排除步驟是對VoIP撥號對等體進行硬編碼以使用正確的編解碼器。

編解碼器協商值（來自debug voip ccapi inout）

有關編解碼器的詳細資訊，請參閱瞭解編解碼器：複雜性、硬體支援、MOS和協商。

協商值	意義
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	清除通道

音調型別

音調型別	意義
CC_TONE_RINGBACK 0x1	鈴聲
CC_TONE_FAX 0x2	傳真音調
CC_TONE_BUSY 0x4	忙音
CC_TONE_DIALTONE 0x8	撥號音
CC_TONE_OOS 0x10	服務中斷音調
CC_TONE_ADDR_ACK 0x20	地址確認音
CC_TONE_DISCONNECT 0x40	中斷連線音
CC_TONE_OFF_HOOK_NOTICE 0x80	提示電話處於摘機狀態的音調
CC_TONE_OFF_HOOK_ALERT 0x100	CC_TONE_OFF_HOOK_NOTICE的更緊急版本
CC_TONE_CUSTOM 0x200	自訂色調-當您指定自訂色調時使用
CC_TONE_NULL 0x0	空色調

FAX-Rate和VAD功能值

值	意義
CC_CAP_FAX_NONE 0x1	傳真停用或無法使用
CC_CAP_FAX_VOICE 0x2	語音通話
CC_CAP_FAX_144 0x4	1.44萬波特
CC_CAP_FAX_96 0x8	9,600波特
CC_CAP_FAX_72 0x10	7,200波特
CC_CAP_FAX_48 0x20	4,800波特
CC_CAP_FAX_24 0x40	2,400波特
CC_CAP_VAD_OFF 0x1	VAD已停用
CC_CAP_VAD_ON 0x2	啟用VAD

相關資訊

- T1第1層故障排除
- T1故障排除
- 排除串列線路問題
- Cisco IP電話故障排除
- 思科技術支援與下載

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。