

VoIP呼叫故障排除和调试

目录

[简介](#)

[先决条件](#)

[要求](#)

[使用的组件](#)

[规则](#)

[背景信息](#)

[网络中的呼叫流](#)

[路由器呼叫流](#)

[电话接口体系结构](#)

[检验数字和模拟信号 \(POTS 呼叫段 \)](#)

[show controllers t1/E1 \(数字式\)](#)

[show voice port](#)

[debug vpm \(语音处理器模块\)](#)

[检收到和发送的数字\(POTS 呼叫段\)](#)

[show dialplan number](#)

[debug vtsp session](#)

[检验端到端 VoIP 信令 \(VOIP 呼叫段 \)](#)

[debug voip ccapi inout](#)

[了解 VoIP 服务质量\(QoS\) 问题](#)

[VoIP 的原因代码和调试值的详细信息](#)

[Q.931 呼叫断开原因 \(来自 debug voip ccapi inout 的 cause_codes \)](#)

[编解码器协商值 \(来自 debug voip ccapi inout \)](#)

[音调类型](#)

[传真速率和 VAD 能力值](#)

[相关信息](#)

简介

本文档介绍用于对VoIP网络进行故障排除和调试的基本技术和命令。

先决条件

要求

Cisco 建议您了解以下主题：

- VoIP 配置

- 语音 QoS
- VoIP网络的设计和部署

使用的组件

本文档不限于特定的软件和硬件版本。但是，显示的输出基于 Cisco IOS® 软件版本 12.3(8)。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

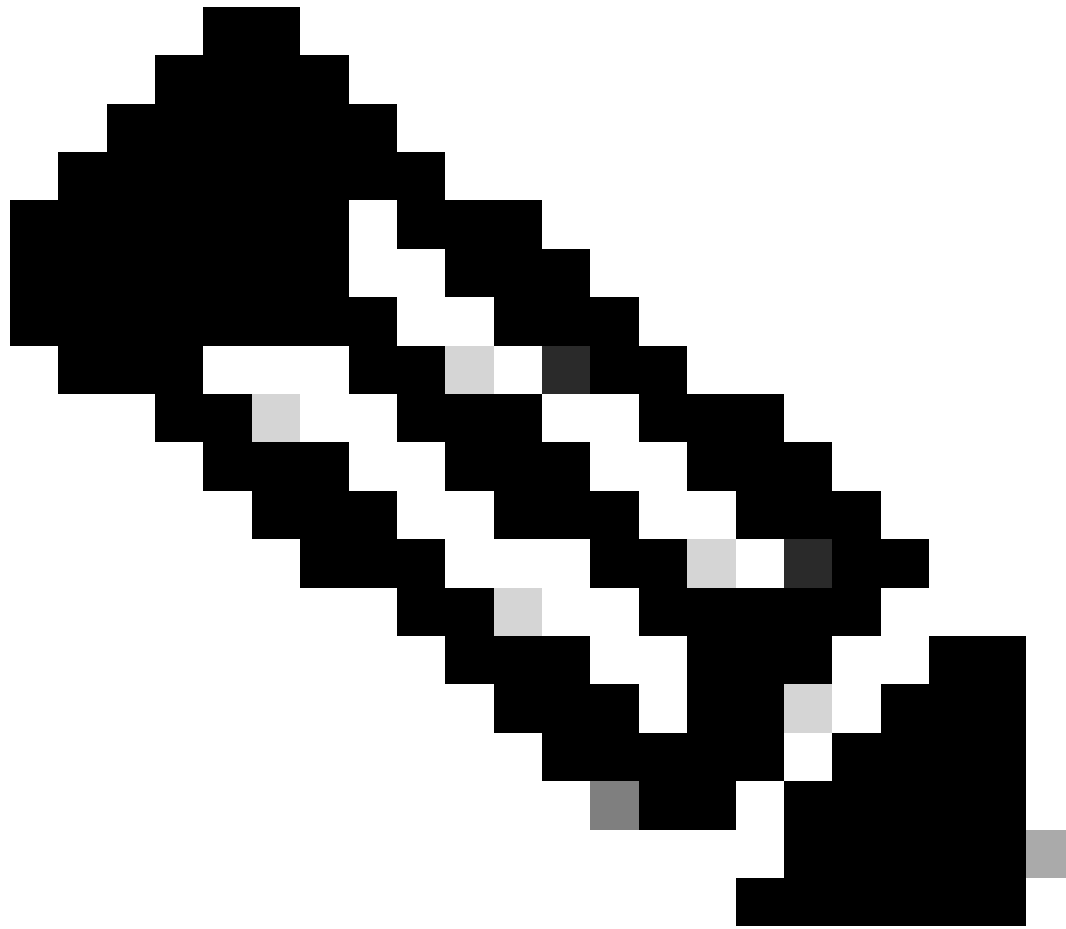
规则

有关文件规则的更多信息请参见“Cisco技术提示规则”。

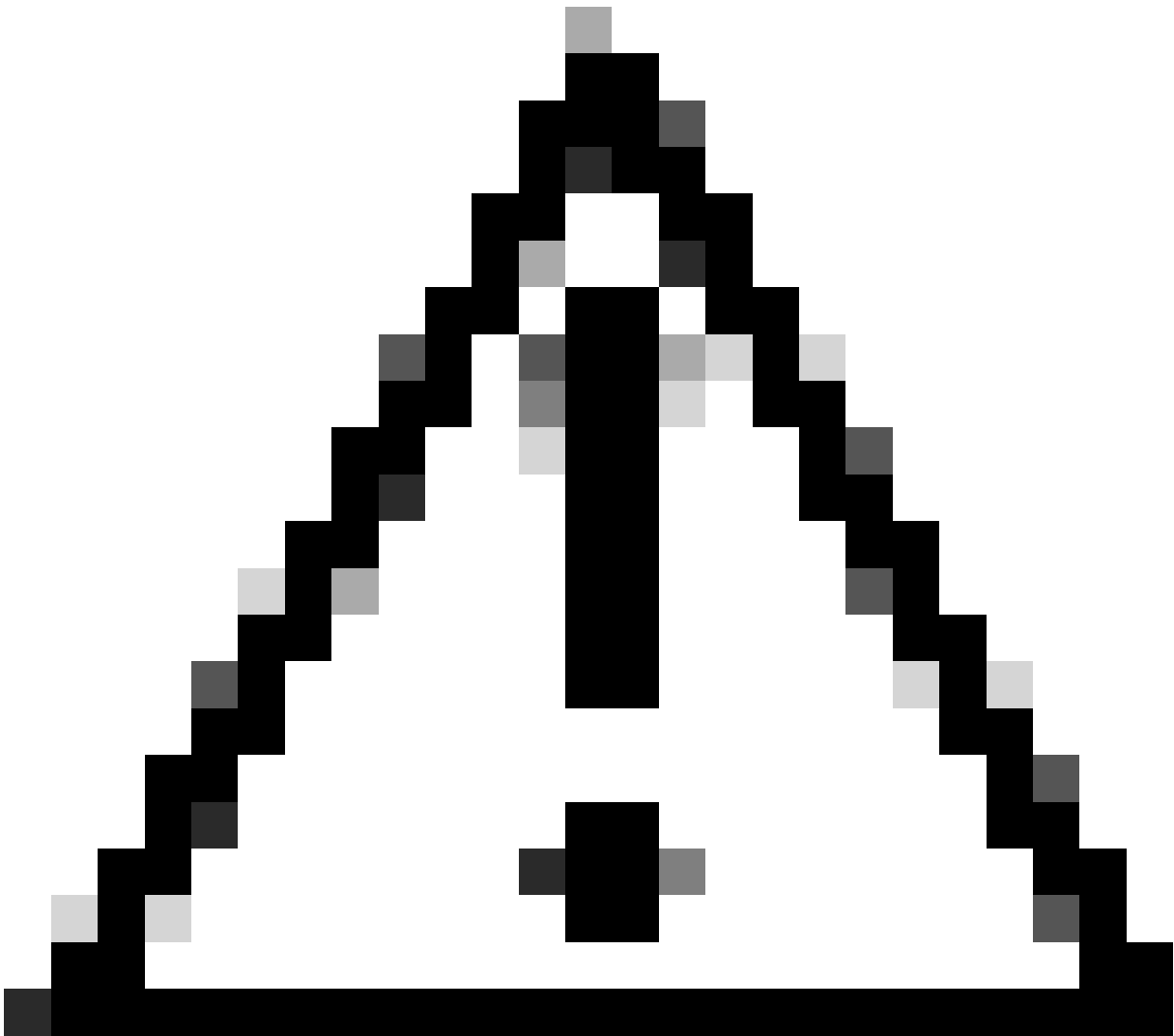
背景信息

本文档展示用于对 VoIP 网络进行故障排除和调试的基本技术和命令。首先概要介绍 Cisco 路由器中的语音呼叫流和电话体系结构，然后按如下步骤分步介绍 VoIP 的故障排除过程：

1. [验证数字和模拟信令。](#)
2. [验证从模拟和数字语音端口接收和发送的数字。](#)
3. [验证端到端 VoIP 信令。](#)
4. [了解 VoIP 服务质量 \(QoS\) 问题。](#)
5. [了解 VoIP 的原因代码和调试值的详细信息。](#)



注意：本文档不解释Cisco VoIP网关和网守中使用的Cisco IOS架构的各个方面。而主要是显示哪些命令可以使用，以及命令输出中的哪些字段最有价值。



注意：调试Cisco IOS需要占用大量处理器。使用本文档中列出的调试时，应特别注意。有关详细信息，请参阅有关 Debug 命令的重要信息。

运行调试时，需要在日志中启用时间戳。在启用模式下，使用命令 `service timestamps debug datetime msec` 和 `service timestamps log datetime msec` 启用时间戳。时间戳有助于确定状态更改之间的间隔时间。

网络中的呼叫流

在开始任何 VoIP 故障排除或调试之前要考虑的一项重要因素是 VoIP 呼叫由三个呼叫段组成。这些呼叫段分别是：源普通旧式电话系统 (POTS)、VoIP 和目的 POTS。如下图所示。故障排除和调试需要先分别关注每一个呼叫段，然后关注整个 VoIP 呼叫。

路由器呼叫流

以下定义解释了路由器呼叫流图中显示的主要组件的功能：

呼叫控制 API (应用程序编程接口) —三个客户端均利用了呼叫控制 API。这三个客户端分别是 CLI、简单网络管理协议 (SNMP) 代理和会话应用。呼叫控制API (也称为CCAPI) 的主要功能是：

- 确定呼叫段 (例如，它是哪个拨号对等体？它来自何处？)。
- 决定由哪个会话应用程序接受呼叫 (例如，由谁处理它？)。
- 调用数据包处理程序。
- 将所有呼叫段合并到一起。
- 开始记录呼叫统计信息。

会话应用程序和拨号方案映射—会话应用程序使用拨号方案映射，将号码映射到拨号对等体 (本地 POTS 或远程 VoIP)。拨号方案映射使用拨号对等体表来查找活动的拨号对等体。

电话和VoIP服务提供商接口(SPI) -电话SPI与POTS (模拟：fxs、fxo、e&m数字：isdn、qsig、e&m等) 拨号对等体通信。VoIP SPI 是 VoIP 对等体的特有接口。电话/DSP 驱动程序向电话 SPI 提供服务，而 VoIP SPI 依赖于会话协议。

电话接口体系结构

此图表显示 Cisco 路由器电话构件的体系结构，以及它们如何彼此互动。

以下列表描述了主要的流程图组件的功能和定义：

- 呼叫控制应用编程接口 (CCAPI) — 建立、终止和桥接呼叫段的软件实体。
- 语音电话服务提供商(VTSP)— 一个Cisco IOS进程，为来自呼叫控制API的请求提供服务，并将适当的请求传达给数字信号处理器(DSP)或VPM。
- 语音处理器模块 (VPM) — VPM 用于在电话端口信号状态机 (SSM)、DSP 资源管理器和 VTSP 之间桥接和协调信令交换过程。
- DSP 资源管理器—DSPRM 提供接口，由 VTSP 用来向 DSP 发送消息或从 DSP 接收消息。
- 数据包处理程序—数据包处理程序在 DSP 和对等呼叫段之间转发数据包。
- 呼叫对等体—呼叫对等体是相反的呼叫段。这可以是另一个电话语音连接 (POTS)、VoFR、VoATM 或 VoIP 连接。

检验数字和模拟信号 (POTS 呼叫段)

验证数字和模拟信令的方法是：

- 确定接收到适当的挂机和摘机模拟或数字信令。
- 确定路由器和交换机配置 (CO 或 PBX) 两端已配置正确的 E&M、FXO 和 FXS 信令。
- 验证 DSP 处于数字收集模式。

这些部分中概要介绍的命令可用于验证信令。

show controllers t1/E1 (数字式)

show controllers t1 [slot/port]—首先使用此命令。它显示路由器与交换机 (CO 或 PBX) 之间的数字 T1 连接是否已启动 , 以及它是否能正常工作。此命令的输出如下所示 :

```
<#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 , 请使用show controllers e1命令。有关详细信息 , 请参阅 :

- [T1 第一层故障排除](#)
- [T1 故障排除流程图](#)
- [串行线路问题的故障排除](#)

show voice port

show voice portslot-number/port— 使用此命令显示端口状态和在Cisco语音接口卡(VIC)的语音端口上配置的参数。与所有Cisco IOS命令一样 , show running-config不显示默认值 , 而此命令显示默认值。

下面是 E&M 语音端口的示例输出 :

```
<#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 (语音处理器模块)

以下命令用于调试 VPM 电话接口 :

- debug vpm signal— 此命令用于收集信令事件的调试信息 , 有助于解决PBX信令问题。
- debug vpm spi — 此命令可以跟踪语音端口模块服务提供商接口 (SPI) 与呼叫控制 API 的交互情况。此 debug 命令显示有关如何处理每个网络指示和应用程序请求的信息。
- debug vpm dsp—此命令显示从 VPM 上的 DSP 发送到路由器的消息 , 在您怀疑 VPM 不工作时很有帮助。这是检查 VPM 是否响应摘机指示以及计算从接口传递消息的时间的简单方法。
- debug vpm all— 此EXEC命令启用了所有debug vpm命令 : debug vpm spi、 debug vpm signal和debug vpm dsp。
- debug vpm port—使用此命令将调试输出限制到特定端口。例如 , 此输出仅显示端口1/0/0的 debug vpm dspmessages :

```
debug vpm dsp
```

```
debug vpm port 1/0/0
```

debug vpm signalCommand的输出示例

```
<#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]
fxs1s_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]

fxs1s_offhook_onhook

*Mar 10 16:09:17.370: htsp_process_event:
[1/0/0, 1.3 , 41] fxs1s_offhook_timer
*Mar 10 16:09:17.382: htsp_process_event:
[1/0/0, 1.2 , 7]

fxs1s_onhook_release
```

如果挂机和摘机未正确发出信令，请检查以下各项：

- 验证接线是正确的。
- 验证路由器和交换机（CO 或 PBX）均正确接地。
- 检验连接的两端是否具有等效的信令配置。不匹配的配置可能导致不完整或单向信令。

有关 E&M 故障排除的详细信息，请参阅了解模拟 E & M 接口类型和接线安排以及故障排除。

debug vpm spiCommand的输出示例

```
<#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
```

检验收到和发送的数字(POTS 呼叫段)

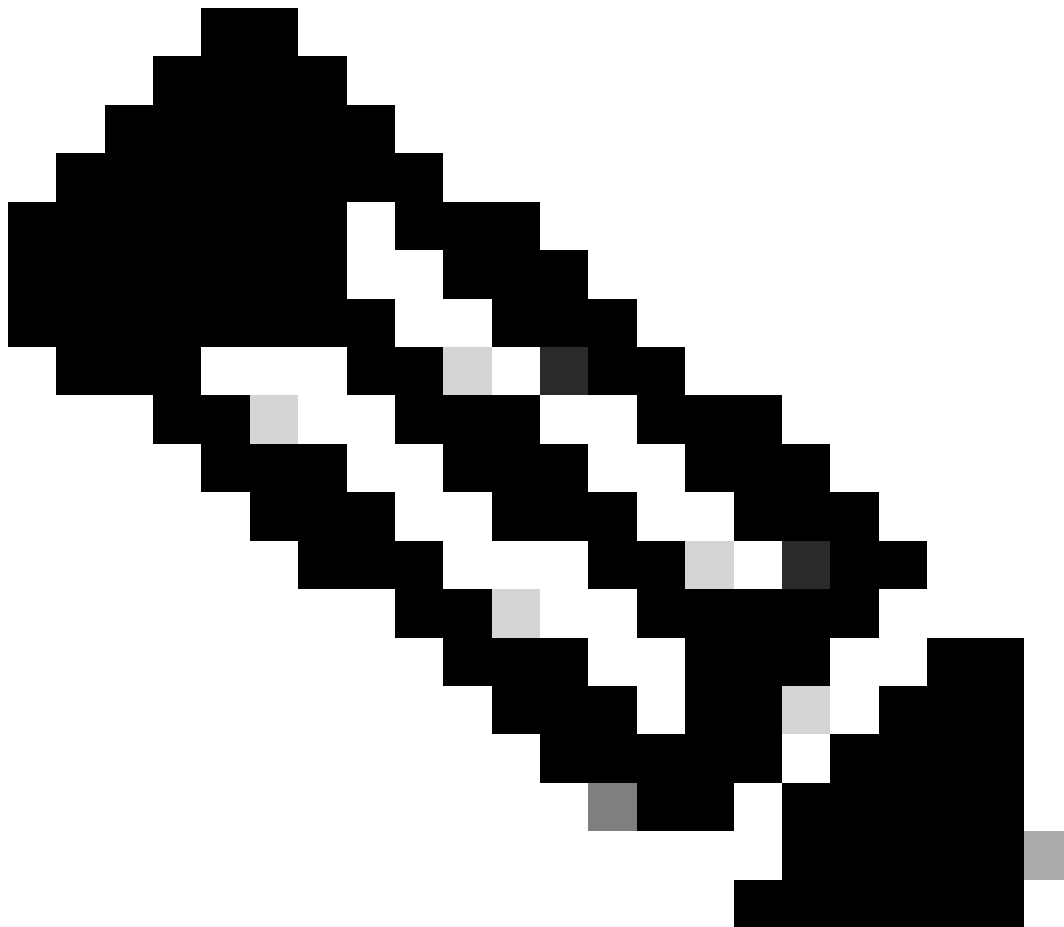
当挂机和摘机信令得到验证并且能正常工作后，请立即验证语音端口（数字或模拟）上收到或发送了正确的数字。如果发送或接收的数字不完整或不正确，则无法匹配拨号对等体，交换机（CO 或

PBX) 也不能使正确的站点响铃。有些命令可用于验证接收/发送的数字：

- show dialplan number—此命令用于显示在拨打特定的电话号码后到达哪个拨号对等体。
- debug vtsp session—此命令显示有关如何处理每个网络指示和应用程序请求的信息、信令指示以及 DSP 控制消息。
- debug vtsp dsp—在 Cisco IOS 软件版本 12.3 以前，此命令在语音端口收到数字后将数字显示出来。但是，在 Cisco IOS 软件版本 12.3 及以后，debug 命令的输出不再显示数字。可结合使用 debug hpi detail 和 debug hpinotification 来查看传入的数字。
- debug vtsp all—此命令启用了以下 debug 语音电话服务提供商(VTSP)命令：debug vtsp session、debug vtsp error、debug vtsp dsp。

show dialplan number

show dialplan number <digit_string>—此命令显示数字字符串所匹配的拨号对等体。如果可以匹配多个拨号对等体，则按匹配的顺序显示它们。



注意：对于长度可变的拨号对等体，您需要在电话号码末尾使用#号，以便匹配以T结尾的目标模式。

此命令的输出如下所示：

```
<#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-rtsp,
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 命令显示有关路由器如何根据来自信令堆栈的信令指示以及来自应用程序的请求与 DSP 进行交互的信息。此 debug 命令显示有关如何处理每个网络指示和应用程序请求的信息、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

```

如果确定没有正确发送或接收数字，则可能有必要使用数字采集器（测试工具）或T1测试器来验证数字是否以正确的频率和计时间隔发送。如果为交换机（CO或PBX）“不正确”发送这些值，则可能需要调整路由器或交换机（CO或PBX）上的某些值，以便它们匹配和互操作。这些值通常是数字持续时间或数字间持续时间值。用来检查数字是否正确发送的另一项是交换机（CO或PBX）中可以添加或删除数字的任意号码转换表。

检验端到端 VoIP 信令 (VOIP 呼叫段)

在您验证语音端口信令能正确工作并且收到了正确的数字之后，即可进行 VoIP 呼叫控制故障排除和调试。以下要素解释了为何呼叫控制调试会变成一项复杂的工作：

- Cisco VoIP网关使用H.323信令完成呼叫。H.323由三层呼叫协商和呼叫建立组成：H.225、H.245和H.323。这些协议结合使用 TCP 和 UDP，设置和建立呼叫。
- 端到端VoIP调试显示许多Cisco IOS状态机。任何状态机的问题都可能会导致呼叫失败。
- 端到端 VoIP 调试可能非常繁琐，并且会产生大量调试输出。

debug voip ccapi inout

用来调试端到端 VoIP 呼叫的主要命令是 debug voip ccapi inout。呼叫调试的输出如以下输出所示。

```
<#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)csz(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)csz(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)csz(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)csz(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)csz(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)csz(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)csz(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)csz(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_app1:
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_app1:
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)
```

如果呼叫失败且原因出现在呼叫设置的VoIP部分，则您可能需要查看呼叫设置的H.225或H.245 TCP部分，而不仅仅是H.323设置的UDP部分。能用来调试 H.225 或 H.245 呼叫建立的命令是：

- debug ip tcp transactions和debug ip tcp packet— 这些命令检查H.225和H.245协商的TCP部分。它们返回 TCP 连接的 IP 地址、TCP 端口和状态。
- debug cch323 h225— 此命令检查呼叫协商的H.225部分，并根据已处理的事件来跟踪H.225状态机的状态转换。可将其当作 H.323 呼叫建立的三个部分中的第 1 层部分。
- debug cch323 h245— 此命令检查呼叫协商的H.245部分，并根据已处理的事件来跟踪H.245状态机的状态转换。可将其当作 H.323 呼叫建立的三个部分中的第 2 层部分。

了解 VoIP 服务质量(QoS) 问题

当 VOIP 呼叫适当建立时，下一步将是检验语音质量是否良好。尽管本文档不涵盖 QoS 故障排除，但是要获得良好的语音质量，需要考虑以下指南：

- 了解 VoIP 呼叫的每个编解码器需要占用多少带宽。这包括第 2 层和 IP/UDP/RTP 标头。有关详细信息，请参阅[修改语音呼叫的带宽消耗计算](#)。
- 了解呼叫途经的 IP 网络的特性。例如，帧中继网络的带宽在 CIR 时与在高 CIR (或突发) 时差别很大。在后一种情况下，数据包可能会被丢弃或在帧中继云中排队。确保尽可能控制并消除延迟和抖动。单向传输延迟不得超过150毫秒 (根据G.114建议) 。
- 使用排队技术确定VoIP流量并划分优先级。
- 当您通过低速链路传输VoIP时，请使用第2层数据包分段技术，例如在点对点链路上使用带链路分段和交织(LFI)的MLPPP，或在帧中继链路上使用FRF.12。由于 VOIP 数据包可以插入链路，较大数据包的分段使传输 VOIP 流量时所发生的抖动和延迟减少。
- 尝试使用不同的编解码器，并在启用和禁用 VAD 的情况下尝试呼叫，尽可能将问题限定到 DSP，而不是 IP 网络。

有了 VoIP，在进行 QoS 故障排除时最主要的问题就是丢弃的数据包和可能导致延迟和抖动的网络瓶颈。

查找：

- 接口丢弃
- 缓冲丢弃
- 接口拥塞
- 链路拥塞

需要检查 VoIP 呼叫路径中的每个接口。并且，消除丢弃和拥塞。而且，需要尽可能减少往返延迟。VoIP 终点之间的 Ping 操作可提供链路往返延迟的指标。尽可能的往返延迟不能超过300毫秒。如果延迟必须超出此值，则需要采取措施，确保此延迟稳定不变，以免导致抖动或可变的延迟。

还必须进行验证，以确保Cisco IOS排队机制将VoIP数据包放置在正确的队列中。Cisco IOS命令，例如show queue interface或show priority有助于对排队进行验证。

VoIP 的原因代码和调试值的详细信息

当您阅读调试以及调试中的相关值时，请使用下面的表。

Q.931 呼叫断开原因 (来自 debug voip ccapi inout 的 cause_codes)

呼叫断开原因值 (十六进制)	含义和编号 (十进制)
CC_CAUSE_UANUM = 0x1	未分配的编号。(1)
CC_CAUSE_NO_ROUTE = 0x3	没有通往目标的路由。(3)
CC_CAUSE_NORM = 0x10	正常呼叫清除。(16)
CC_CAUSE_BUSY = 0x11	用户忙。(17)
CC_CAUSE_NORS = 0x12	用户无响应。(18)
CC_CAUSE_NOAN = 0x13	用户无应答。(19)
CC_CAUSE_REJECT = 0x15	呼叫被拒绝。(21)
CC_CAUSE_INVALID_NUMBER = 0x1C	编号无效。(28)
CC_CAUSE_UNSP = 0x1F	正常，未指定。(31)
CC_CAUSE_NO_CIRCUIT = 0x22	无电路。(34)
CC_CAUSE_NO_REQ_CIRCUIT = 0x2C	无请求的电路。(44)
CC_CAUSE_NO_RESOURCE = 0x2F	无资源。(47) 1
CC_CAUSE_NOSV = 0x3F	服务或选项不可用，或者未指定。(63)

¹此问题可能是由于H323设置中的编解码器不匹配引起的，因此第一个故障排除步骤是对VoIP拨号对等体进行硬编码以使用正确的编解码器。

编解码器协商值 (来自 debug voip ccapi inout)

有关编解码器的详细信息，请参阅[了解编解码器：复杂性、硬件支持、MOS和协商](#)。

协商值	含义
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

音调类型

音调类型	含义
CC_TONE_RINGBACK 0x1	铃声
CC_TONE_FAX 0x2	传真音
CC_TONE_BUSY 0x4	忙音
CC_TONE_DIALTONE 0x8	拨号音
CC_TONE_OOS 0x10	未使用音
CC_TONE_ADDR_ACK 0x20	地址确认音
CC_TONE_DISCONNECT 0x40	断开音
CC_TONE_OFF_HOOK_NOTICE 0x80	表示电话处于摘机状态的声音
CC_TONE_OFF_HOOK_ALERT 0x100	CC_TONE_OFF_HOOK_NOTICE 的更紧急版本
CC_TONE_CUSTOM 0x200	自定义提示音-在指定自定义提示音时使用
CC_TONE_NULL 0x0	空音

传真速率和 VAD 能力值

价值	含义
CC_CAP_FAX_NONE 0x1	传真禁用或不可用
CC_CAP_FAX_VOICE 0x2	语音呼叫
CC_CAP_FAX_144 0x4	14,400 波特
CC_CAP_FAX_96 0x8	9600 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 已禁用
CC_CAP_VAD_ON 0x2	启用 vad

相关信息

- [T1 第一层故障排除](#)
- [T1 故障排除](#)
- [串行线路问题的故障排除](#)
- [Cisco IP 电话故障排除](#)
- [思科技术支持和下载](#)

关于此翻译

思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。