VoIP 통화 문제 해결 및 디버그

목차

소개

사전 요구 사항

- 요구 사항
- 사용되는 구성 요소
- 표기 규칙

배경 정보

네트워크의 통화 흐름

라우터 통화 흐름

텔레포니 인터페이스 아키텍처

디지털 및 아날로그 신호 확인(POTS Call-Leg)

- show controllers T1 / E1(디지털)
- 음성 포트 표시
- vpm(음성 프로세서 모듈) 디버그

Verify Digits Received and Sent(수신 및 발신 숫자 확인)(POTS Call-Leg)

- 다이얼 플랜 번호 표시
- vtsp 세션 디버그

엔드 투 엔드 VoIP 신호 처리(VOIP Call-Leg) 확인

- voip ccapi inout 디버그

VoIP QoS(Quality of Service) 문제 이해

VoIP에 대한 원인 코드 및 디버그 값의 세부 정보

- Q.931 통화 연결 끊김 원인(디버그 voip ccapi 입력 시 cause codes)
- 코덱 협상 값(디버그 voip ccapi 입력)
- 톤 유형
- FAX-Rate 및 VAD 기능 값

관련 정보

소개

이 문서에서는 VoIP 네트워크의 문제를 해결하고 디버깅하는 기본 기술 및 명령에 대해 설명합니다
.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- VoIP 컨피그레이션

- 음성 QoS
- VoIP 네트워크 설계 및 구축

사용되는 구성 요소

이 문서는 특정 소프트웨어 및 하드웨어 버전으로 한정되지 않습니다. 그러나 표시된 출력은 Cisco IOS® Software 릴리스 12.3(8)을 기반으로 합니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

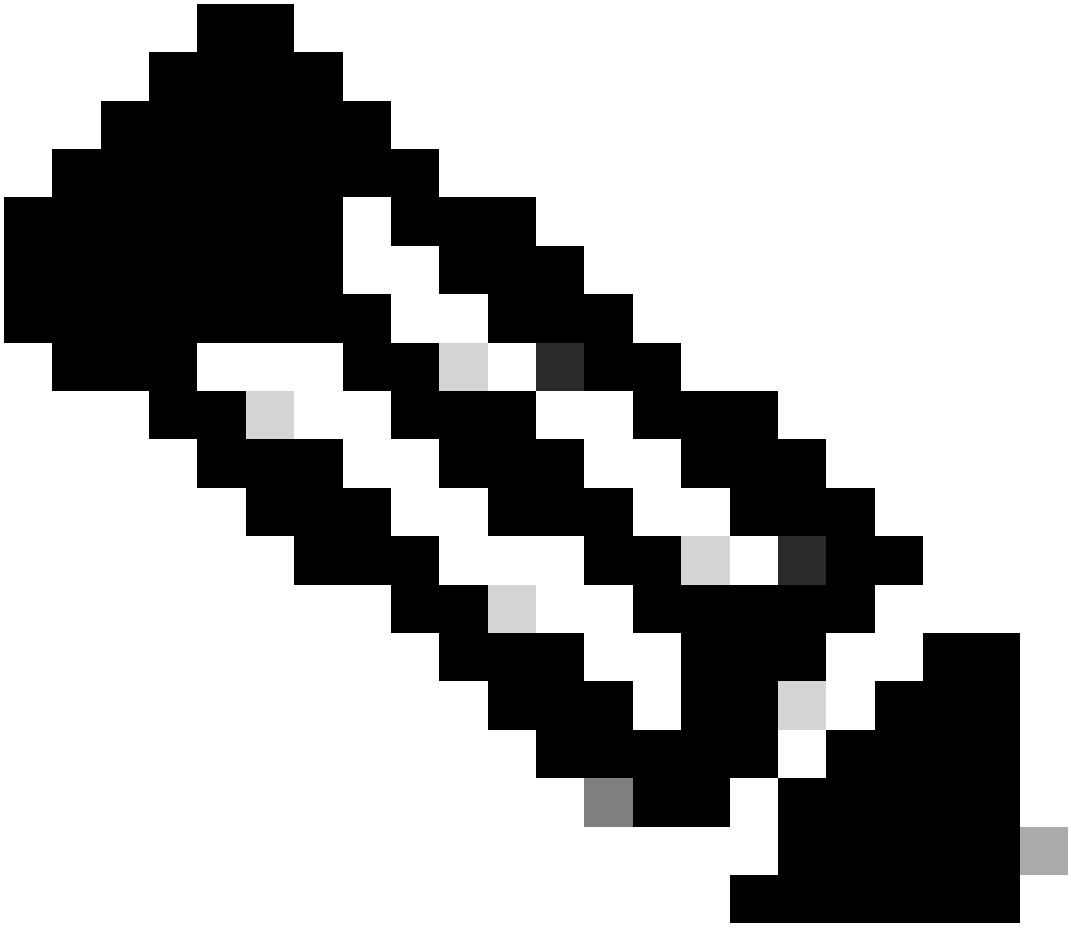
표기 규칙

문서 규칙에 대한 자세한 내용은 Cisco 기술 팁 표기 규칙을 참조하십시오.

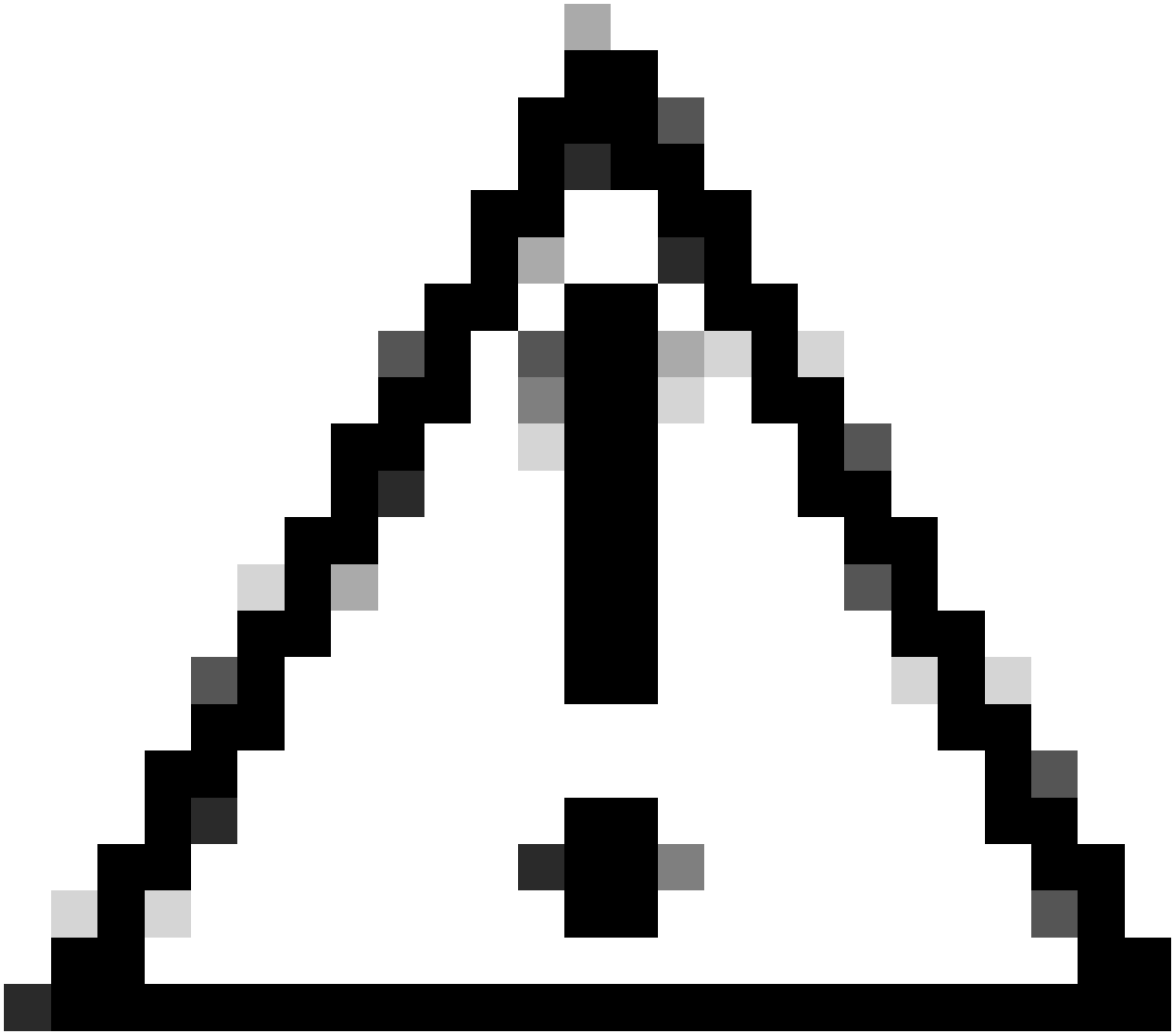
배경 정보

이 문서에서는 VoIP 네트워크의 문제 해결 및 디버깅을 위한 기본 기법과 명령에 대해 설명합니다. Cisco 라우터의 음성 통화 흐름 및 텔레포니 아키텍처에 대한 개요와 더불어 다음 단계에서 제시되는 단계별 VoIP 문제 해결 방법이 제공됩니다.

1. 디지털 및 아날로그 신호 처리를 확인합니다.
2. 아날로그 및 디지털 음성 포트에서 수신 및 전송된 숫자를 확인합니다.
3. 엔드 투 엔드 VoIP 신호 처리를 확인합니다.
4. VoIP QoS(Quality of Service) 문제를 파악합니다.
5. VoIP에 대한 원인 코드 및 디버그 값의 세부 정보를 이해합니다.



참고: 이 문서에서는 Cisco VoIP 게이트웨이 및 게이트키퍼에 사용되는 Cisco IOS 아키텍처의 모든 측면을 설명하지는 않습니다. 대신 어떤 명령을 사용할 수 있으며 명령 출력에서 어떤 필드가 가장 유용한지 보여 주기 위한 것입니다.



주의: Cisco IOS를 디버깅하는 데 프로세서 사용량이 많습니다. 이 문서에 나열된 디버그를 사용할 때는 주의해야 합니다. 자세한 내용은 Debug 명령에 대한 중요 정보를 참조하십시오.

디버깅은 로그에서 활성화된 타임스탬프로 실행해야 합니다. enable 모드에서 service timestamps debug datetime msec, service timestampslog datetime msec 명령을 사용하여 시간 스탬프를 활성화합니다. 타임스탬프는 상태 변경 간의 시간 간격을 결정하는 데 도움이 됩니다.

네트워크의 통화 흐름

VoIP 문제 해결 또는 디버깅을 시작하기 전에 고려해야 할 중요한 요소는 VoIP 통화가 3개의 통화 레그로 구성된다는 것입니다. 이러한 통화 레그는 소스 POTS(Plain Old Telephone Systems), VoIP 및 대상 POTS입니다. 이 다이어그램은 다음과 같습니다. 트러블슈팅 및 디버깅은 먼저 각 레그에 독립적으로 초점을 맞춘 다음 VoIP 통화 전체에 중점을 두어야 합니다.

라우터 통화 흐름

이러한 정의는 라우터 통화 흐름 다이어그램에 표시되는 주요 구성 요소의 기능을 설명합니다.

Call Control API(Application Programming Interface) - 세 클라이언트가 통화 제어 API를 사용합니다. 세 클라이언트는 CLI, SNMP(Simple Network Management Protocol) 에이전트 및 세션 애플리케이션입니다. 통화 제어 API(CCAPI라고도 함)의 주요 기능은 다음과 같습니다.

- 통화 레그를 확인합니다(예: 어느 다이얼 피어입니까? 어디서 왔습니까?).
- 통화를 받을 세션 애플리케이션을 결정합니다(예: 누가 처리합니까?).
- 패킷 처리기를 호출합니다.
- 통화 레그를 함께 회의합니다.
- 통화 통계를 기록하기 시작합니다.

Session Application and Dial Plan Mapper(세션 애플리케이션 및 다이얼 플랜 매퍼) - 세션 애플리케이션은 다이얼 플랜 매퍼를 사용하여 다이얼 피어(로컬 POTS 또는 원격 VoIP)에 번호를 매핑합니다. Dial Plan Mapper는 Dial Peer Table을 사용하여 활성 다이얼 피어를 찾습니다.

텔레포니 및 VoIP SPI(Service Provider Interface) - 텔레포니 SPI는 POTS(아날로그: fxs, fxo, e&m 디지털: isdn, qsig, e&m 등) 다이얼 피어와 통신합니다. VoIP SPI는 VoIP 피어에 대한 특정 인터페이스입니다. 텔레포니/DSP 드라이버는 VoIP SPI가 세션 프로토콜을 사용하는 동안 텔레포니 SPI에 서비스를 제공합니다.

텔레포니 인터페이스 아키텍처

이 다이어그램은 Cisco 라우터 텔레포니 구성 요소의 아키텍처와 상호 작용 방식을 보여줍니다.

이 목록에서는 기본 다이어그램 구성 요소의 기능 및 정의에 대해 설명합니다.

- CCAPI(Call Control Application Programming Interface) - 통화 레그를 설정, 종료 및 연결하는 소프트웨어 엔티티입니다.
- VTSP(Voice Telephony Service Provider) - 통화 제어 API의 요청을 서비스하고 DSP(Digital Signal Processor) 또는 VPM에 대한 적절한 요청을 작성하는 Cisco IOS 프로세스입니다.
- VPM(Voice Processor Module) - VPM은 SSM(Telephony Ports Signaling State Machine), DSP Resource Manager 및 VTSP 간의 시그널링 프로세스 브리징 및 조정 역할을 담당합니다.
- DSP Resource Manager—DSPRM은 VTSP가 DSP와 메시지를 주고받을 수 있는 인터페이스를 제공합니다.
- Packet Handler(패킷 처리기) - 패킷 처리기는 DSP와 피어 통화 레그 사이에 패킷을 전달합니다.
- Call Peer(통화 피어) - 통화 피어가 반대 통화 레그입니다. 다른 POTS(Telephony Voice Connection), VoFR, VoATM 또는 VoIP 연결이 될 수 있습니다.

디지털 및 아날로그 신호 확인(POTS Call-Leg)

디지털 및 아날로그 신호 확인:

- 적절한 온 훅 및 오프 훅 아날로그 또는 디지털 신호가 수신되었는지 확인합니다.
- 라우터와 스위치(CO 또는 PBX) 모두에 적절한 E&M, FXO 및 FXS 신호 처리가 구성되어 있는지 확인합니다.
- DSP가 숫자 수집 모드에 있는지 확인합니다.

이 섹션에 설명된 명령을 사용하여 신호 처리를 확인할 수 있습니다.

show controllers T1 / E1(디지털)

show controllers t1 [slot/port] - 먼저 이 명령을 사용합니다. 라우터와 스위치(CO 또는 PBX) 간의 디지털 T1 연결이 작동 또는 중단되었고 제대로 작동하는지 보여줍니다. 이 명령의 출력은 다음과 같습니다.

```
<#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
```

E1을 사용하는 경우 show controllers e1 명령을 사용합니다. 자세한 내용은 다음을 참조하십시오.

- T1 레이어 1 트러블슈팅
- T1 문제 해결 순서도
- 직렬 회선 문제 해결

음성 포트 표시

show voice portslot-number/port - 이 명령을 사용하여 Cisco VIC(voice interface card)의 음성 포트에 구성된 포트 상태 및 매개변수를 표시할 수 있습니다. 모든 Cisco IOS 명령과 마찬가지로, 기본 값은 show running-config에 표시되지 않지만 이 명령과 함께 표시됩니다.

다음은 E&M 음성 포트의 샘플 출력입니다.

```
<#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
```

vpm(음성 프로세서 모듈) 디버그

다음 명령은 VPM 텔레포니 인터페이스를 디버깅하는 데 사용됩니다.

- debug vpm signal - 이 명령은 이벤트를 시그널링하기 위한 디버그 정보를 수집하는 데 사용되며 PBX에 대한 시그널링 문제를 해결하는 데 유용할 수 있습니다.
- debug vpm spi - 이 명령은 음성 포트 모듈 SPI(service provider interface)가 호출 제어 API와 어떻게 인터페이스하는지 추적합니다. 이 debug 명령은 각 네트워크 표시 및 애플리케이션 요청이 처리되는 방식에 대한 정보를 표시합니다.
- debug vpm dsp - 이 명령은 VPM의 DSP에서 라우터로 보내는 메시지를 표시하며, VPM이 작동하지 않는 것으로 의심되는 경우 유용할 수 있습니다. VPM이 오프 훅 표시에 응답하는지 확인하고 인터페이스에서 오는 신호 메시지의 타이밍을 평가하는 간단한 방법입니다.
- debug vpm all—이 EXEC 명령은 debug vpm 명령, debug vpm spi, debug vpm signal 및 debug vpm dsp를 모두 활성화합니다.
- debug vpm port - 디버그 출력을 특정 포트로 제한하려면 이 명령을 사용합니다. 예를 들어 이 출력은 포트 1/0/0에 대해서만 debug vpm dspmessages를 표시합니다.

```
debug vpm dsp

debug vpm port 1/0/0
```

디버그 vpm signalCommand의 샘플 출력

```
<#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
```

온 훅 및 오프 훅 신호가 제대로 표시되지 않으면 다음 항목을 확인하십시오.

- 케이블이 올바른지 확인합니다.
- 라우터와 스위치(CO 또는 PBX)가 모두 제대로 접지되었는지 확인합니다.
- 연결의 양쪽 끝에 동등한 신호 컨피그레이션이 있는지 확인합니다. 컨피그레이션이 일치하지 않으면 불완전하거나 단방향 시그널링이 발생할 수 있습니다.

E&M 트러블슈팅에 대한 자세한 내용은 아날로그 E&M 인터페이스 유형 및 배선 배열 이해 및 트러블슈팅을 참조하십시오.

디버그 vpm spiCommand에 대한 샘플 출력

```
<#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
```

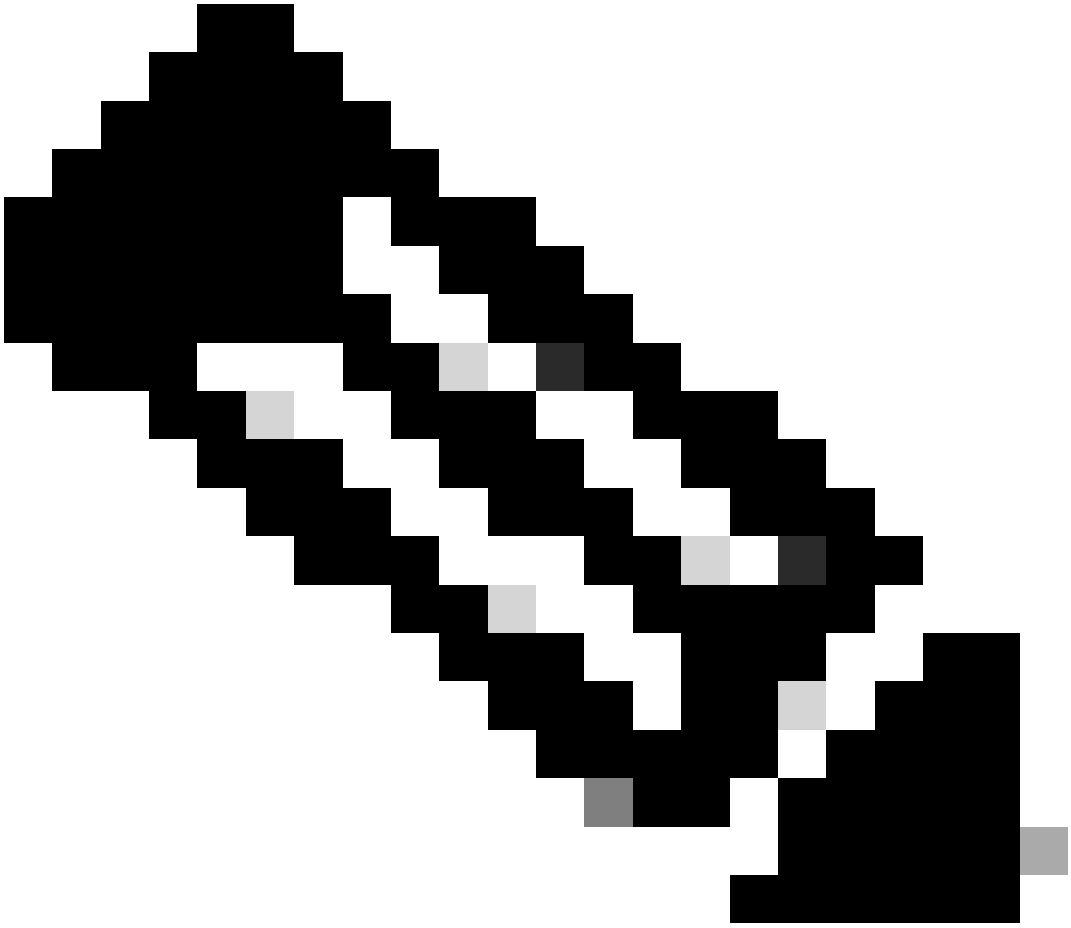
Verify Digits Received and Sent(수신 및 발신 숫자 확인)(POTS Call-Leg)

온 훅 및 오프 훅 신호가 확인되어 제대로 작동하면 올바른 숫자가 음성 포트(디지털 또는 아날로그)에서 수신되거나 전송되는지 확인합니다. 다이얼 피어가 일치하지 않거나 불완전하거나 잘못된 숫자를 보내거나 받을 경우 스위치(CO 또는 PBX)가 올바른 스테이션에 연결할 수 없습니다. 수신/발신된 숫자를 확인하는 데 사용할 수 있는 일부 명령은 다음과 같습니다.

- show dialplan number - 이 명령은 특정 전화 번호가 다이얼될 때 어떤 다이얼 피어에 연결되는지 표시하는 데 사용됩니다.
- debug vtsp session - 이 명령은 각 네트워크 표시 및 애플리케이션 요청이 처리되는 방법, 신호 표시 및 DSP 제어 메시지에 대한 정보를 표시합니다.
- debug vtsp dsp —Cisco IOS Software Release 12.3 이전 버전에서는 이 명령이 음성 포트에서 수신한 숫자를 표시합니다. 그러나 Cisco IOS Software Release 12.3 이상에서는 debug 명령의 출력에 숫자가 더 이상 표시되지 않습니다. debug hpi detail 및 debug hpinotification의 조합을 사용하여 수신 숫자를 확인할 수 있습니다.
- debug vtsp all - 이 명령은 debug vtsp(voice telephony service provider) 명령(debug vtsp session, debug vtsp error, debug vtsp error 및 debug vtsp dsp)을 활성화합니다.

다이얼 플랜 번호 표시

show dialplan number <digit_string>—이 명령은 일련의 숫자로 일치하는 다이얼 피어를 표시합니다. 여러 다이얼 피어를 일치시킬 수 있는 경우 모두 일치하는 순서대로 표시됩니다.



참고: T로 끝나는 대상 패턴을 매칭하려면 전화 번호 끝에 있는 # 기호를 가변 길이의 다이얼 피어에 사용해야 합니다.

이 명령의 출력은 다음과 같습니다.

```
<#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
```

vtsp 세션 디버그

debug vtsp session 명령은 라우터가 신호 처리 스택의 신호 처리 표시 및 애플리케이션의 요청에 따라 DSP와 상호 작용하는 방법에 대한 정보를 표시합니다. 이 debug 명령은 각 네트워크 표시 및 애플리케이션 요청이 처리되는 방식, 신호 표시 및 DSP 제어 메시지에 대한 정보를 표시합니다.

```
<#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
```

숫자가 제대로 전송되거나 수신되지 않는 것으로 확인되면, 숫자-그래버(테스트 도구) 또는 T1 테스터를 사용하여 숫자가 올바른 빈도와 타이밍 간격으로 전송되는지 확인해야 할 수 있습니다. 스위치(CO 또는 PBX)에 대해 "잘못" 전송되는 경우, 라우터 또는 스위치(CO 또는 PBX)의 일부 값을 조정해야 일치시키고 상호 작용할 수 있습니다. 일반적으로 숫자 지속 시간 및 숫자 간 지속 시간 값입니다. 숫자가 올바르게 전송된 것처럼 보이는지 여부를 검사하기 위한 또 다른 항목은 숫자를 추가하거나 제거할 수 있는 스위치(CO 또는 PBX)의 숫자 변환 테이블입니다.

엔드 투 엔드 VoIP 신호 처리(VOIP Call-Leg) 확인

음성 포트 신호 처리가 제대로 작동하고 올바른 숫자가 수신되는지 확인한 후 VoIP 통화 제어 문제 해결 및 디버깅으로 이동합니다. 이러한 요인은 통화 제어 디버깅이 복잡한 작업이 될 수 있는 이유를 설명합니다.

- Cisco VoIP 게이트웨이는 H.323 신호를 사용하여 통화를 완료합니다. H.323은 통화 협상 및 통화 설정의 세 가지 레이어(H.225, H.245 및 H.323)로 구성됩니다. 이러한 프로토콜은 TCP와 UDP의 조합을 사용하여 통화를 설정하고 설정합니다.
- 엔드 투 엔드 VoIP 디버깅은 여러 Cisco IOS 상태 머신을 표시합니다. 모든 상태 시스템에 문제가 있으면 통화가 실패할 수 있습니다.
- 엔드-투-엔드 VoIP 디버깅은 매우 자세할 수 있으며 디버그 출력을 많이 생성할 수 있습니다.

voip ccapi inout 디버그

엔드 투 엔드 VoIP 통화를 디버깅하는 기본 명령은 debug voip ccapi inout입니다. 이 출력에는 호출 디버그의 출력이 표시됩니다.

```
<#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)
```

통화가 실패하고 원인이 통화 설정의 VoIP 부분에 있는 것으로 나타나면 H.323 설정의 UDP 부분만 보는 것이 아니라 통화 설정의 H.225 또는 H.245 TCP 부분을 확인해야 할 수 있습니다. H.225 또는 H.245 통화 설정을 디버깅하는 데 사용할 수 있는 명령은 다음과 같습니다.

- debug ip tcp transactions and debug ip tcp packet - 이 명령은 H.225 및 H.245 협상의 TCP 부분을 검사합니다. IP 주소, TCP 포트 및 TCP 연결의 상태를 반환합니다.
- debug cch323 h225 - 이 명령은 통화 협상의 H.225 부분을 검사하고 처리된 이벤트를 기반으로 H.225 상태 머신의 상태 전환을 추적합니다. 3부 H.323 통화 설정의 레이어 1 부분으로 간주합니다.
- debug cch323 h245 - 이 명령은 통화 협상의 H.245 부분을 검사하고 처리된 이벤트를 기반으로 H.245 상태 머신의 상태 전환을 추적합니다. H.323 통화 설정 3부 중 레이어 2 부분으로 간주합니다.

VoIP QoS(Quality of Service) 문제 이해

VoIP 통화가 제대로 설정되었을 때 다음 단계는 음질이 양호한지 확인하는 것입니다. 이 문서에서는 QoS 트러블슈팅에 대해 다루지 않지만 양호한 음성 품질을 얻으려면 다음 지침을 고려해야 합니다.

- VoIP 통화가 각 코덱에서 사용하는 대역폭 양을 파악합니다. 여기에는 레이어 2 및 IP/UDP/RTP 헤더가 포함됩니다. 자세한 내용은 음성 통화에 대한 대역폭 소비 계산 수정을 참조하십시오.
- 통화가 이동하는 IP 네트워크의 특성을 파악합니다. 예를 들어, CIR에서 프레임 릴레이 네트워크의 대역폭은 CIR(또는 버스트) 위의 대역폭과 크게 다릅니다. 이 경우 프레임 릴레이 클라우드에서 패킷을 삭제하거나 큐에 넣을 수 있습니다. 지연 및 지터를 최대한 제어하고 제거합니다. 단방향 전송 지연은 150ms를 초과할 수 없습니다(G.114 권장 사항당).
- VoIP 트래픽을 식별하고 우선 순위를 지정할 수 있는 대기열 처리 기술을 사용합니다.
- 저속 링크를 통해 VoIP를 전송하는 경우 포인트-투-포인트 링크에서 LFI(Link Fragmentation and Interleaving)를 사용하는 MLPPP 또는 프레임 릴레이 링크에서 FRF.12와 같은 레이어 2 패킷 조각화 기술을 사용합니다. 더 큰 데이터 패킷의 프래그먼트화는 VoIP 패킷이 링크에 인터리빙될 수 있기 때문에 VoIP 트래픽을 전송하는 데 있어서 덜 지터(jitter)와 지연을 허용한다.
- 다른 코덱을 사용하고 VAD가 활성화 및 비활성화된 통화를 시도하여 IP 네트워크가 아닌 DSP로 문제를 좁힐 수 있습니다.

VoIP에서는 QoS 문제를 해결할 때 주로 고려해야 할 사항이 삭제된 패킷과 지연 및 지터를 일으킬 수 있는 네트워크 병목 현상이 발생합니다.

검색 대상:

- 인터페이스 삭제
- 버퍼 삭제
- 인터페이스 혼잡
- 링크 혼잡

VoIP 통화 경로의 각 인터페이스를 검사해야 합니다. 또한 낙하와 혼잡을 없앱니다. 또한, 왕복 지연 시간을 최대한 줄여야 합니다. VoIP 엔드포인트 간의 ping은 링크의 왕복 지연을 나타냅니다. 왕복 지연 시간은 300ms를 초과할 수 없습니다. 지연이 이 값을 초과해야 하는 경우 지터 또는 가변 지연을 초래하지 않도록 이 지연이 일정하도록 해야 합니다.

Cisco IOS 큐잉 메커니즘이 VoIP 패킷을 적절한 큐 내에 배치하는지 확인해야 합니다. Cisco IOS 명령(예: show queue interface 또는 show priority)을 사용하면 대기를 확인할 수 있습니다.

VoIP에 대한 원인 코드 및 디버그 값의 세부 정보

디버깅 및 디버깅 내의 관련 값을 읽을 때 이 테이블을 사용합니다.

Q.931 통화 연결 끊김 원인(디버그 voip ccapi 입력 시 cause_codes)

통화 연결 끊김 원인 값(16진수)	의미 및 숫자(10진수)
CC_CAUSE_UANUM = 0x1	할당되지 않은 번호. (1)
CC_CAUSE_NO_ROUTE = 0x3	대상에 대한 경로가 없습니다. (3)
CC_CAUSE_NORM = 0x10	일반 통화 지우기. (16)
CC_CAUSE_BUSY = 0x11	사용자가 사용 중입니다. (17)
CC_CAUSE_NORS = 0x12	사용자 응답이 없습니다. (18)
CC_CAUSE_NOAN = 0x13	사용자 응답이 없습니다. (19)
CC_CAUSE_REJECT = 0x15	통화가 거부되었습니다. (21)
CC_CAUSE_INVALID_NUMBER = 0x1C	잘못된 번호입니다. (28)
CC_CAUSE_UNSP = 0x1F	일반, 지정되지 않음. (31)
CC_CAUSE_NO_CIRCUIT = 0x22	회선 없음. (34)
CC_CAUSE_NO_REQ_CIRCUIT = 0x2C	요청한 회로가 없습니다. (44)
CC_CAUSE_NO_RESOURCE = 0x2F	리소스가 없습니다. (47)[1]
CC_CAUSE_NOSV = 0x3F	서비스 또는 옵션을 사용할 수 없거나 지정되지 않았습니다. (63)

[1] 이 문제는 H323 설정 내의 코덱 불일치로 인해 발생할 수 있으므로 첫 번째 문제 해결 단계에서는 올바른 코덱을 사용하도록 VoIP 다이얼 피어를 하드 코딩합니다.

코덱 협상 값(디버그 voip ccapi 입력)

코덱에 대한 자세한 내용은 코덱 이해: 복잡성, 하드웨어 지원, MOS 및 협상을 참조하십시오.

협상 값	의미
코덱=0x00000001	G711 ULAW 64K PCM
코덱=0x00000002	G711 ALAW 64K PCM
코덱=0x00000004	G729
코덱=0x00000004	G729IETF
코덱=0x00000008	G729a
코덱=0x00000010	G726r16
코덱=0x00000020	G726r24
코덱=0x00000040	G726r32
코덱=0x00000080	G728
코덱=0x00000100	G723r63
코덱=0x00000200	G723r53

코덱=0x00000400	GSMFR
코덱=0x00000800	G729b
코덱=0x00001000	G729ab
코덱=0x00002000	G723ar63
코덱=0x00004000	G723ar53
코덱=0x00008000	채널 지우기(_C)

톤 유형

톤 유형	의미
CC_TONE_RINGBACK 0x1	벨소리
CC_TONE_FAX 0x2	팩스 신호음
CC_TONE_BUSY 0x4	통화중 신호음
CC_TONE_DIALTONE 0x8	발신음
CC_TONE_OOS 0x10	서비스 중단 신호음
CC_TONE_ADDR_ACK 0x20	주소 승인 신호음
CC_TONE_DISCONNECT 0x40	연결 끊기 신호음
CC_TONE_OFF_HOOK_NOTICE 0x80	전화기가 오프 훅임을 나타내는 신호음
CC_TONE_OFF_HOOK_ALERT 0x100	더 긴급한 버전의 CC_TONE_OFF_HOOK_NOTICE
CC_TONE_CUSTOM 0x200	Custom Tone(사용자 지정 톤) - 사용자 지정 톤을 지정할 때 사용합니다.
CC_TONE_NULL 0x0	Null 톤

FAX-Rate 및 VAD 기능 값

값	의미
CC_CAP_FAX_NONE 0x1	팩스를 사용할 수 없거나 사용할 수 없음
CC_CAP_FAX_VOICE 0x2	음성 통화
CC_CAP_FAX_144 0x4	14,400보드
CC_CAP_FAX_96 0x8	9,600보드
CC_CAP_FAX_72 0x10	7,200보드
CC_CAP_FAX_48 0x20	4,800보드
CC_CAP_FAX_24 0x40	2,400보드
CC_CAP_VAD_OFF 0x1	VAD 사용 안 함
CC_CAP_VAD_ON 0x2	VAD 사용

관련 정보

- T1 레이어 1 트러블슈팅
- T1 문제 해결
- 직렬 회선 문제 해결
- Cisco IP 텔레포니 문제 해결
- Cisco 기술 지원 및 다운로드

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.