Table of Contents

概要

Cisco Unified Customer Voice Portal（CVP）サバイバビリティ スクリプトが入力ゲートウェイの着信ダイヤルピアで設定されるときこの資料に接続失敗を解決する方法を記述されています。

Kabeer Noorudeen によって貢献される、Cisco TAC エンジニア。

前提条件

要件

次の項目に関する知識があることが推奨されます。

- CVP 広範囲のコールフロー
- IOS Gateway および PRI プロトコル
- Cisco Unified Intelligent Contact Management（ICM）、Cisco Unified Contact Center Enterprise（UCCE）配備

使用するコンポーネント

このドキュメントの情報は、次のソフトウェアのバージョンに基づくものです。

- CVP サーバ 9.0 以上に
- UCCE 9.0 以上に
- IOS Gateway 15.X

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。 ネットワークが稼働中の場合は、コマンドが及ぼす潜在的な影響を十分に理解しておく必要があります。

問題

サービスがにサバイバビリティ スクリプトを引き起こす 着信POT Dialpeer 追加されるときコールはこの原因と失敗します:

ccCallDisconnect: 原因 Value=81

注: コールが失敗するとき報告されるエラーコードを表示するために debug コマンドを、**デバッグ VoIP CCAPI インプット**、実行して下さい。 さらにコールが失敗するとき報告される SIP エラーメッセージが表示されたいと思う場合、debug コマンド **デバッグ ccsip メッセージを**実行して下さい。

矛盾したエラーメッセージは**デバッグ ccsip メッセージが**有効に なるとき報告されます。
SIP メッセージはコールが失敗するそれ以外、紛らわしい場合もあるネットワーク VRU ラベルが戻った後多くの示す値を与えません。

サービスが着信POT ダイヤル ピアからのから取除かれる場合、すべてはうまく働きます。

手順

ステップ 1： 1 つは入力ゲートウェイ デバッグを有効に します: **デバッグ VoIP CCAPI インプット、デバッグ ccsip メッセージ、デバッグ音声 アプリケーションすべて、**および **debug isdn q931。**

ステップ 2.問題を再現して下さい。 ゲートウェイにコールを送信して下さい。

ステップ 3.ログを集めて下さい。 コマンドを、**show logging** 使用して下さい**。**

CCAPI ログ（**デバッグ VoIP CCAPI インプット**）を検知 すれば、正常にサバイバビリティ スクリプトを離れて渡されるコール gets がそれからコール失敗することをちょうど見、:

スポイラー

*Sep 13 19:29:57.507 CT: //21/9E2AF1DC800C/CCAPI/cc_process_call_setup_ind:
>>>>CCAPI はアプリケーション「_ManagedAppProcess_cvp-survivability」に渡しましたタグ 101 の CID 21 を
*Sep 13 19:29:58.507 CT: //21/9E2AF1DC800C/CCAPI/ccCallSetupAck:
コール Id=21

*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/cc_api_call_disconnected:
原因 Value=81、Interface=0x2500E10、コール Id=21
*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/cc_api_call_disconnected:
呼び出しエントリ（Responsed=FALSE、原因 Value=81、リトライ Count=0）
*Sep 13 19:29:58.551 CT: //22/9E2AF1DC800C/CCAPI/ccCallDisconnect:
原因 Value=81、Tag=0x0、呼び出しエントリ（前の接続解除 Cause=0、接続解除 Cause=0）
*Sep 13 19:29:58.551 CT: //22/9E2AF1DC800C/CCAPI/ccCallDisconnect:
原因 Value=81、呼び出しエントリ（Responsed=FALSE、原因 Value=81）
*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/ccCallDisconnect:
原因 Value=81、Tag=0x0、呼び出しエントリ（前の接続解除 Cause=0、接続解除 Cause=81）
*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/ccCallDisconnect:
原因 Value=81、呼び出しエントリ（Responsed=TRUE、原因 Value=81）

*Sep 13 19:29:57.507 CT: //21/9E2AF1DC800C/CCAPI/cc_process_call_setup_ind: >>>>CCAPI はアプリケーション「_ManagedAppProcess_cvp-survivability " *Sep 13 19:29:58.507 CT にタグ 101 の CID 21 を渡しました: //21/9E2AF1DC800C/CCAPI/ccCallSetupAck: コール Id=21*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/cc_api_call_disconnected: 原因 Value=81、Interface=0x2500E10、コール Id=21*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/cc_api_call_disconnected: 呼び出しエントリ（Responsed=FALSE、原因 Value=81、リトライ Count=0)*Sep 13 19:29:58.551 CT: //22/9E2AF1DC800C/CCAPI/ccCallDisconnect: 原因 Value=81、Tag=0x0、呼び出しエントリ（前の接続解除 Cause=0、接続解除 Cause=0)*Sep 13 19:29:58.551 CT: //22/9E2AF1DC800C/CCAPI/ccCallDisconnect: 原因 Value=81、呼び出しエントリ（Responsed=FALSE、原因 Value=81)*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/ccCallDisconnect: 原因 Value=81、Tag=0x0、呼び出しエントリ（前の接続解除 Cause=0、接続解除 Cause=81)*Sep 13 19:29:58.551 CT: //21/9E2AF1DC800C/CCAPI/ccCallDisconnect: 原因 Value=81、呼び出しエントリ（Responsed=TRUE、原因 Value=81）

tcl（例えば debugvoice **アプリケーションすべて**）のためのデバッグを検知 すれば、tcl が現在の呼び出し状態を得ることを試みるとき切られる発信者に会います。

スポイラー

*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_InfotagObjCmd: infotag は evt_state_current なります
*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_InfotagGetObjCmd: infotag は evt_state_current なります
*Sep 13 20:29:54.211 CT: //81//AFW_:/vtr_ev_state_current: argc 2
*Sep 13 20:29:54.211 CT: //81//AFW_:/vtr_ev_state_current: イベント[CALL_INIT]
***Sep 13 20:29:54.211 CT: //81//TCL: /tcl_FSMObjCmd: fsm setstate CALLER_DISCONNECTED**
***Sep 13 20:29:54.211 CT: //81//TCL: /tcl_FSMSetStateObjCmd: setstate setstate CALLER_DISCONNECTED**
*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_InfotagObjCmd: infotag は con_ofleg 81 を得ます

*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_InfotagGetObjCmd: infotag は con_ofleg 81 を得ます
*Sep 13 20:29:54.211 CT: //81//AFW_:/vtr_co_ofleg: argc 3 argindex 2
*Sep 13 20:29:54.211 CT: //81//Tcl: /tcl_parseCallID_vartagObj: VARTAG 変換 レグ Count=1
*Sep 13 20:29:54.211 CT: //81//AFW_:/vtr_co_ofleg: EV_CONNECTIONS []
*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_LegObjCmd: レグ 接続解除 81
*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_LegDisconnectObjCmd: 接続解除 81
*Sep 13 20:29:54.211 CT: //81//Tcl: /tcl_parseCallID_vartagObj: VARTAG 変換 レグ Count=1
*Sep 13 20:29:54.211 CT: //81//TCL: /tcl_InfotagObjCmd: infotag は evt_state_current *Sep 13 20:29:54.211 CT を得ます: //81//TCL: /tcl_InfotagGetObjCmd: infotag は evt_state_current *Sep 13 20:29:54.211 CT を得ます: //81//AFW_:/vtr_ev_state_current: argc 2*Sep 13 20:29:54.211 CT: //81//AFW_:/vtr_ev_state_current: イベント[CALL_INIT] *Sep 13 20:29:54.211 CT: //81//TCL: /tcl_FSMObjCmd: fsm setstate CALLER_DISCONNECTED *Sep 13 20:29:54.211 CT: //81//TCL: /tcl_FSMSetStateObjCmd: setstate setstate CALLER_DISCONNECTED *Sep 13 20:29:54.211 CT: //81//TCL: /tcl_InfotagObjCmd: infotag は con_ofleg 81 *Sep 13 20:29:54.211 CT を得ます: //81//TCL: /tcl_InfotagGetObjCmd: infotag は con_ofleg 81 *Sep 13 20:29:54.211 CT を得ます: //81//AFW_:/vtr_co_ofleg: argc 3 argindex 2*Sep 13 20:29:54.211 CT: //81//Tcl: /tcl_parseCallID_vartagObj: VARTAG 変換 レグ Count=1*Sep 13 20:29:54.211 CT: //81//AFW_:/vtr_co_ofleg: EV_CONNECTIONS [] *Sep 13 20:29:54.211 CT: //81//TCL: /tcl_LegObjCmd: レグ 接続解除 81 *Sep 13 20:29:54.211 CT: //81//TCL: /tcl_LegDisconnectObjCmd: 81 *Sep 13 20:29:54.211 CT を切って下さい: //81//Tcl: /tcl_parseCallID_vartagObj: VARTAG 変換 レグ Count=1

メッセージが PSTN メッセージと同期していないことが isdn メッセージ（**debug isdn q931**）に潜ればこの場合、わかります。 セットアップの直後におよびコール設定受付の前に入っているファシリティ メッセージがあります。 サバイバビリティ スクリプトはこの状況に対処できません。

スポイラー

*Sep 13 19:53:31.763 CT: ISDN Se0/0/0:23 Q931: RX < -セットアップ pd = 8 callref = 0x1114
***Sep 13 19:53:31.767 CT: ISDN Se0/0/0:23 Q931: RX < -ファシリティ pd = 8 callref = 0x1114**
***Sep 13 19:53:31.767 CT: ISDN Se0/0/0:23 **エラー**: L3_BadPeerMsg: イベント 0x62 Cr 0x9114 callid 0x0**
*Sep 13 19:53:31.767 CT: ISDN Se0/0/0:23 Q931: TX - > RELEASE_COMP pd = 8 callref = 0x9114
*Sep 13 19:53:32.767 CT: ISDN Se0/0/0:23 Q931: TX - > CALL_PROC pd = 8 callref = 0x9114
*Sep 13 19:53:32.767 CT: ISDN Se0/0/0:23 Q931: TX - > 接続応答 pd = 8 callref = 0x9114
*Sep 13 19:53:32.803 CT: ISDN Se0/0/0:23 Q931: RX < - RELEASE_COMP pd = 8 callref = 0x1114
*Sep 13 19:53:32.807 CT: ISDN Se0/0/0:23 Q931: RX < - RELEASE_COMP pd = 8 callref = 0x1114
***Sep 13 19:53:32.807 CT: ISDN Se0/0/0:23 **エラー**: L3_BadPeerMsg: イベント 0x5A Cr 0x9114 callid 0x0**

*Sep 13 19:53:31.763 CT: ISDN Se0/0/0:23 Q931: RX < -セットアップ pd = 8 callref = 0x1114 *Sep 13 19:53:31.767 CT: ISDN Se0/0/0:23 Q931: RX < -ファシリティ pd = 8 callref = 0x1114 *Sep 13 19:53:31.767 CT: ISDN Se0/0/0:23 **エラー**: L3_BadPeerMsg: イベント 0x62 Cr 0x9114 callid 0x0*Sep 13 19:53:31.767 CT: ISDN Se0/0/0:23 Q931: TX - > RELEASE_COMP pd = 8 callref = 0x9114 *Sep 13 19:53:32.767 CT: ISDN Se0/0/0:23 Q931: TX - > CALL_PROC pd

= 8 callref = 0x9114 *Sep 13 19:53:32.767 CT: ISDN Se0/0/0:23 Q931: TX - > 接続応答 pd = 8 callref = 0x9114*Sep 13 19:53:32.803 CT: ISDN Se0/0/0:23 Q931: RX < - RELEASE_COMP pd = 8 callref = 0x1114 *Sep 13 19:53:32.807 CT: ISDN Se0/0/0:23 Q931: RX < - RELEASE_COMP pd = 8 callref = 0x1114 *Sep 13 19:53:32.807 CT: ISDN Se0/0/0:23 **エラー**: L3_BadPeerMsg: イベント 0x5A Cr 0x9114 callid 0x0

解決策

L3_BadPeerMsg はこの問題を解決するここにキーです。 このエラーはゲートウェイと PSTN 間のスイッチタイプにミスマッチがあるとき報告されます。

推奨事項は PSTN によっておよびゲートウェイが一致する ISDNスイッチタイプを設定 するためにコマンドを実行することです。 このシナリオで PSTN のスイッチタイプは **primary-ni** です。 米国顧客向けに、広く使われたスイッチタイプは **primary-ni** です。

入力ゲートウェイで設定されたコマンド **isdn switch-type primary-ni は**問題を解決しました。