Configuring Connection Trunk for VoIP Gateways

Document ID: 12432

Contents

Introduction

A trunk (tie−line) is a permanent point−to−point communication line between two voice ports. The **connection trunk** command creates a permanent Voice over IP (VoIP) call between two VoIP gateways. It simulates a trunk connection through the creation of virtual trunk tie−lines between two telephony endpoints. To the connected systems, it appears as if a T1 trunk is directly connected between them.

Prerequisites

Requirements

These platforms support a VoIP connection trunk:

- Cisco 2600, 3600, and 3700 Series digital and analog interfaces
- Cisco 7200/7500 Series digital interfaces
- Cisco MC3810 digital and analog interfaces
- Cisco 1750/1751 and 1760

Note: AS5300/AS5400/AS5800 platforms do not and will not support connection trunks, because they are not suitable for WAN connectivity with heavy traffic volumes.

Components Used

The information in this document is based on these software and hardware versions:

- Cisco IOS® Software Release 12.2(10a) with IP Plus Feature Set
- Cisco 2610 Series Routers

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

Configure

In this section, you are presented with the information to configure the features described in this document.

Note: To find additional information on the commands used in this document, use the Command Lookup Tool (registered customers only)

Connection Trunk Design Considerations and Limitations

- Connection Trunk mode is supported on T1/E1 channel associated signaling (CAS) interfaces. A connection trunk is not supported on T1/E1 interfaces that are using common channel signaling (CCS); for example, QSIG and PRI Q.931. A connection trunk is not supported on Foreign Exchange Office (FXO) ports configured for ground−start.
- Connection Trunk mode is a permanent connection; the VoIP call is always connected independently of the plain old telephone service (POTS) port being on−hook or off−hook. Connection Trunk has statically configured endpoints and does not require a user to dial to connect calls. It also allows supplemental call signaling, such as hookflash or point−to−point hoot−n−holler, to be passed over the IP network between the two telephony devices.
- Connection Trunk mode is supported with these voice port combinations:
  - recEive and transMit (E & M) to E & M (same type)
  - FXO to Foreign Exchange Station (FXS)
  - FXS to FXS (with no signaling)

  Note: These voice port combinations are permitted between analog to analog, digital to digital, and analog to digital interfaces. Also, when you are configuring FXS to FXS, signaling can not be conveyed because it would not be a transparent path. The connected devices (FXOs) would be trying to signal each other. It is possible to get this design to work if you set the voice path to always be open. Configure `signal-type ext-signal` to the VoIP dial peer, and the router will no longer wait for signaling before it opens the voice path.
- A connection trunk T1 CAS to E1 CAS mapping does not work by default. Bit−order manipulation on the gateways must be performed and may not always work, based on the PBX support of various ABCD bit signaling.
- A connection trunk allows private line, automatic ringdown−Off−Premise−Extension (PLAR−OPX) type of functionality between FXO and FXS ports. This allows remote stations (connected to FXS ports) to appear to the PBX as physically connected stations. If this remote station does not answer a call, it can be rolled−over to centralized voicemail (if it is configured on the PBX).
- A connection trunk, such as PLAR, does not require the router to collect digits from the telephony device. The permanent VoIP call is created when the router is booted and IP connectivity is established. Because of this, the existing customer dial plan does not have to be altered.
- A connection trunk can pass some telephony signaling, such as hookflash, but it does not pass proprietary PBX signaling. It is not a Transparent CCS (T−CSS) feature.
- A connection trunk, such as PLAR, is defined per voice−port. This means that the voice−port can not operate both in Connection Trunk mode and Collect Dialed−Digits mode. The only instance where this might not be completely desirable would be in a remote office that needs to also dial between local extensions without the use of a centralized PBX. This would require the path of the call to go over the VoIP network and back, as opposed to it being switched within the router. Normally, this should not be a concern.

Configuration Guidelines

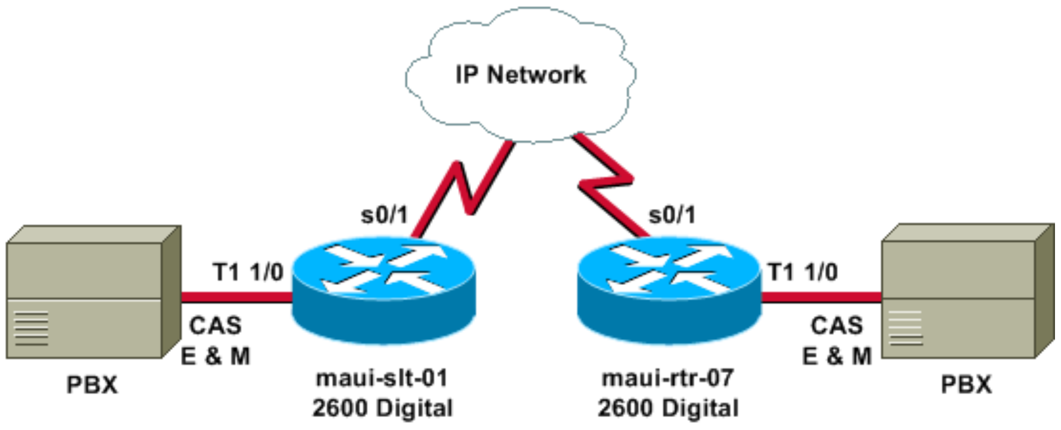
The connection trunk must be configured on both ends of the trunk. When you are configuring a connection trunk with analog interfaces, it must be defined per voice−port. When you are configuring a connection trunk with digital interfaces, there are several options:

- You can define a separate **ds0−group** command for each DS0 (each timeslot), and you can use the **connection trunk** command to define each voice−port that is created. This ensures that DS0 to DS0 mapping is retained on digital trunks.
- You can define a single **ds0−group** command to handle all of the DS0s, and you can define a single **connection trunk** command on the voice−port. This reduces the amount of manual configuration that is required, but there is no guarantee of one−to−one mapping of DS0s on either end of the trunk. In addition, each time that the router reloads, the mapping can be different from the last time. Furthermore, this configuration complicates troubleshooting, because you are not able to isolate the problem to a single (or even a few) timeslots without taking down the entire trunk group. This configuration is also not recommended for T−CCS with proprietary signaling on either end of PBXs, because it would not deliver the signaling channel reliably without one−on−one mapping.
- It is recommended that one side of the connection be configured with the **answer−mode** keyword specified after the **connection trunk** *string* command. This makes one side of the trunk the master side. The gateway (router) with the **answer−mode** keyword is then the slave side. The **answer−mode** command specifies that the gateway will not attempt to initiate a trunk connection, but instead it will wait for an incoming call before it establishes the trunk. This configuration scheme minimizes the time that routers take to bring up trunks and to ensures that trunks go down when connections are lost between two gateways. Otherwise, the gateways might not attempt to re−establish the trunk when the connection is up again.

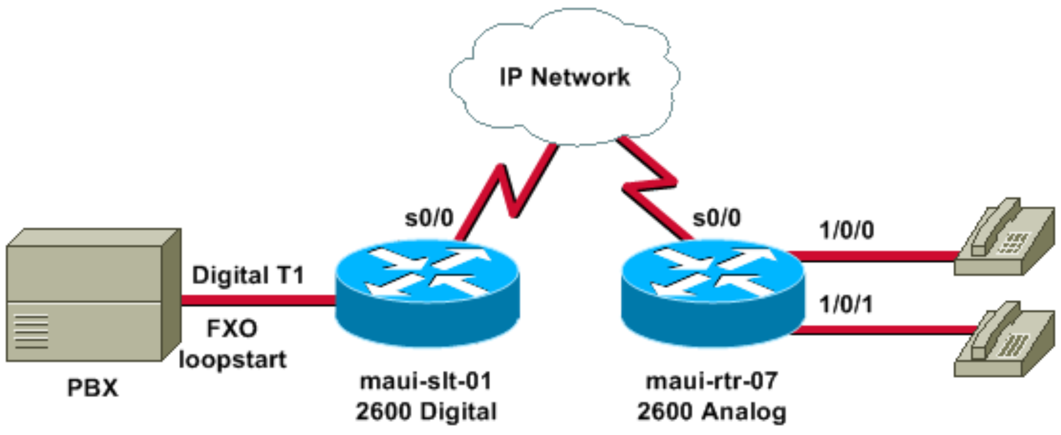
Note: When you issue the **connection trunk** command, you must perform a **shutdown/no shutdown** command sequence on the voice port.

Network Diagram

This document uses these two network setups:



The previous diagram illustrates the digital−to−digital scenario, where both router sides have digital links.



The previous diagram illustrates the digital−to−analog scenario, with digital on one end and analog on the other end.

Configurations

This document uses these configurations:

- Digital−to−digital
    - maui−slt−01
    - maui−rtr−07
- Digital−to−analog
    - maui−slt−01
    - maui−rtr−07

The first configuration (digital−to−digital) shows a typical configuration for a connection trunk between two routers with digital T1 interfaces. In this example, the routers are providing true tie−line replacement between the PBXs.

Digital−to−digital − maui−slt−01

```
version 12.2
 service timestamps debug datetime msec
 service timestamps log datetime msec
 service password-encryption
 !
 hostname maui-slt-01
 !
 voice-card 1
 !
 controller T1 1/0
  framing esf
  linecode b8zs
  ds0-group 1 timeslots 1 type e & m-wink-start
  ds0-group 2 timeslots 2 type e & m-wink-start
  clock source line

!--- The ds0-group command creates the logical voice-ports:
!--- voice-port 1/0:1 and voice-port 1/0:2.

 !
 voice-port 1/0:1
  connection trunk 2000
```

```
!---  master side
!--- This starts the trunk connection using digits 2000 to match
!--- a VoIP dial-peer. The digits are generated internally by the
!--- router and are not received from the voice-port.

 !
 voice-port 1/0:2
  connection trunk 2001
 !
 dial-peer voice 2 voip
  destination-pattern 200.

!--- Matches connection trunk string 2000 and 2001.

  dtmf-relay h245-alphanumeric
  session target ipv4:192.168.100.2
  ip qos dscp cs5 media
 !
 dial-peer voice 1 pots
  destination-pattern 1000
  port 1/0:1

!--- This dial-peer maps to maui-rtr-07 s voice-port 1/0:1.

 !
 dial-peer voice 3 pots
  destination-pattern 1001
  port 1/0:2

!--- This dial-peer maps to maui-rtr-07 s voice-port 1/0:2.

 !
 interface Serial0/1
  ip address 192.168.100.1 255.255.255.0
```

Digital−to−digital − maui−rtr−07

```
version 12.2
 service timestamps debug uptime
 service timestamps log uptime
 service password-encryption
 !
 hostname maui-rtr-07
 !
 voice-card 1
 !
 controller T1 1/0
  framing esf
  linecode b8zs
  ds0-group 1 timeslots 1 type e & m-wink-start
  ds0-group 2 timeslots 2 type e & m-wink-start
  clock source line
 !
 voice-port 1/0:1
  connection trunk 1000 answer-mode

!---  slave side
!--- The answer-mode specifies that the router should not attempt
!--- to initiate a trunk connection, but it should wait for an
!--- incoming call before it establishes the trunk.

 !
 voice-port 1/0:2
  connection trunk 1001 answer-mode
 !
 dial-peer voice 1 voip
  destination-pattern 100.
  dtmf-relay h245-alphanumeric
  session target ipv4:192.168.100.1
  ip qos dscp cs5 media
 !
 dial-peer voice 2 pots
  destination-pattern 2000
  port 1/0:1

!--- This dial-peer terminates the connection
!--- from maui-slt-01 voice-port 1/0:1.

 !
 dial-peer voice 3 pots
  destination-pattern 2001
  port 1/0:2

!--- This dial-peer terminates the connection
!--- from maui-slt-01 voice-port 1/0:2.

 !
 interface Serial0/1
  ip address 192.168.100.2 255.255.255.0
  clockrate 128000
 !
```

The second configuration (digital−to−analog) shows a typical configuration for a connection trunk between two similar routers, one with digital T1 interfaces and another with analog interfaces. The interfaces must be the same type for this to work (for example, E & M wink to E & M wink, E & M immediate to E & M immediate, FXO to FXS and vice versa). In our example, FXO loopstart is signaling on the digital T1 interface and there are analog FXS ports with FXS loopstart signaling on the corresponding side.

Digital−to−analog − maui−slt−01

```
version 12.2
 service timestamps debug datetime msec
 service timestamps log datetime msec
 service password-encryption
 !
 hostname maui-slt-01
 !
 voice vad-time 40000

 !
 voice-card 1

 !
 controller T1 1/0
  framing esf
  linecode b8zs
  ds0-group 1 timeslots 1 type fxo-loopstart
  clock source line

!--- The ds0-group command creates the logical voice-ports:
!--- voice-port 1/0:1 and voice-port 1/0:2.

 !
 voice-port 1/0:1
  connection trunk 2000

!---  master side
!--- This starts the trunk connection using digits 2000 to match
!--- a VoIP dial-peer. The digits are generated internally by the
```

```
!--- router and are not received from the voice-port.

 !
 !
 !
 dial-peer voice 2 voip
  destination-pattern 200.

!--- Matches connection trunk string 2000 and 2001.

  dtmf-relay h245-alphanumeric
  session target ipv4:192.168.100.2
  ip qos dscp cs5 media
 !
 dial-peer voice 1 pots
  destination-pattern 1000
  port 1/0:1

!--- This dial-peer maps to maui-rtr-07's voice-port 1/0/0.

 !
 !
 !
 interface Serial0/1
  ip address 192.168.100.1 255.255.255.0
 !
```

Digital−to−analog − maui−rtr−07

```
version 12.2
 service timestamps debug uptime
 service timestamps log uptime
 service password-encryption
 !
 hostname maui-rtr-07
 !
 !
 voice-port 1/0/0
  connection trunk 1000 answer-mode

!---  slave side
!--- The answer-mode specifies that the router should not attempt
!--- to initiate a trunk connection, but it should wait for an
!--- incoming call before it establishes the trunk.

 !
 !
 dial-peer voice 1 voip
  destination-pattern 100.
  dtmf-relay h245-alphanumeric
  session target ipv4:192.168.100.1
  ip qos dscp cs5 media
 !
 dial-peer voice 2 pots
  destination-pattern 2000
  port 1/0/0

!--- This dial-peer terminates the connection
!--- from maui-slt-01 voice-port 1/0:1.

 !
 !
 !
 interface Serial0/1
  ip address 192.168.100.2 255.255.255.0
  clockrate 128000
 !
```

Verify

This section provides information that you can use to confirm that your configuration is working properly.

Certain **show** commands are supported by the Output Interpreter Tool (registered customers only) , which allows you to view an analysis of **show** command output.

- **show voice call summary** Used to verify that all trunks are up and in the `S_CONNECT` state.

When the trunks comes up, the console will display the message `%HTSP−5−UPDOWN: Trunk port(channel) [1/0:1(1)] is up`.

This is sample output from the **show voice call summary** command:

```
PORT         CODEC    VAD VTSP STATE            VPM STATE
============ ======== === ==================== ======================
3/0:0.1       g729r8    n  S_CONNECT             S_TRUNKED
3/0:1.2       g729r8    n  S_CONNECT             S_TRUNKED
3/0:2.3       g729r8    n  S_CONNECT             S_TRUNKED
```

A trunk that is not up will show up as `S_TRUNK_PEND`:

```
PORT         CODEC    VAD VTSP STATE            VPM STATE
============ ======== === ==================== ======================
3/0:0.1       -             -      -             S_TRUNK_PEND
3/0:1.2       g729r8    n  S_CONNECT             S_TRUNKED
3/0:2.3       g729r8    n  S_CONNECT             S_TRUNKED
```

Troubleshoot

This section provides information that you can use to troubleshoot your configuration.

Troubleshooting Commands

Certain **show** commands are supported by the Output Interpreter Tool (registered customers only) , which allows you to view an analysis of **show** command output.

Note: Before issuing **debug** commands, please see Important Information on Debug Commands.

- **show call history voice | Include DisconnectText** Shows the disconnect reason for the last few failed calls.
- **show voice call summary** Shows the call active on both call legs.
- **show voice dsp** Shows that the Digital Signal Processors (DSPs) are in use and are processing packets.

For more information on troubleshooting VoIP calls, refer to Troubleshooting and Debugging VoIP Call Basics and VoIP Debug Commands.

The associated voice−ports on both routers must be **shutdown/no shutdown** after you configure the connection trunk. This also clears the voice ports if you see `user busy` as a disconnect cause.

This is sample command output from the **show voice dsp** command:

```
BOOT                         PAK
TYPE DSP CH CODEC    VERS STATE STATE  RST AI PORT    TS ABORT   TX/RX-PAK-CNT
==== === == ======= ==== ===== ====== === == ======= == ===== ===============
C549 000 01 g729r8   3.4 busy  idle     0  0 3/0:12  13     0 3522765/3578769
         00 g729r8   .41 busy  idle     0  0 3/0:0    1     0 3505023/3560759
C549 001 01 g729r8   3.4 busy  idle     0  0 3/0:13  14     0 3522761/3578601
         00 g729r8   .41 busy  idle     0  0 3/0:1    2     0 3522794/3578579
```

The next sample output is the most common debug output for the **debug voip ccapi inout** command. This debug was taken under the common mistake of a missing POTS peer on the called side. In the example, the analog side router does not have a POTS peer to terminate the trunk; the digital calling side will have these debugs in this situation:

```
maui-slt-01#

*Mar  1 00:11:19.903: cc_api_call_setup_ind (vdbPtr=0x620B2DE8,
callInfo={called=2000,called_oct3=0x81,calling=,calling_oct3=0x0,
calling_oct3a=0x0,calling_xlated=false,subscriber_type_str=RegularLine
,fdest=1,peer_tag=2, prog_ind=3},callID=0x621C45F0)
*Mar  1 00:11:19.903: cc_api_call_setup_ind type 3 , prot 0
*Mar  1 00:11:19.903: cc_process_call_setup_ind (event=0x62332908)
*Mar  1 00:11:19.903: >>>>CCAPI handed cid 3 with tag 2 to app "DEFAULT"
*Mar  1 00:11:19.907: sess_appl: ev(24=CC_EV_CALL_SETUP_IND), cid(3), disp(0)
*Mar  1 00:11:19.907: sess_appl: ev(SSA_EV_CALL_SETUP_IND), cid(3), disp(0)
*Mar  1 00:11:19.907: ssaCallSetupInd
*Mar  1 00:11:19.907: ccCallSetContext (callID=0x3, context=0x621C4E90)
*Mar  1 00:11:19.907: ssaCallSetupInd cid(3), st(SSA_CS_MAPPING),oldst(0),
ev(24)ev->e.evCallSetupInd.nCallInfo.finalDestFlag = 1
*Mar  1 00:11:19.907: ssaCallSetupInd finalDest cllng(1000), clled(2000)
*Mar  1 00:11:19.907: ssaCallSetupInd cid(3), st(SSA_CS_CALL_SETTING),
oldst(0), ev(24)dpMatchPeersMoreArg result= 0
*Mar  1 00:11:19.907: ssaSetupPeer cid(3) peer list:
tag(1) called number (2000)
*Mar  1 00:11:19.907: ssaSetupPeer cid(3), destPat(2000), matched(1),
prefix(), peer(61EE565C), peer->encapType (2)
*Mar  1 00:11:19.907: ccCallProceeding (callID=0x3, prog_ind=0x0)
*Mar  1 00:11:19.907: ccCallSetupRequest (Inbound call = 0x3, outbound
peer =1, dest=, params=0x6233BD30 mode=0, *callID=0x6233C098, prog_ind = 3)
*Mar  1 00:11:19.907: ccCallSetupRequest numbering_type 0x81
*Mar  1 00:11:19.907: ccCallSetupRequest encapType 2 clid_restrict_disable 1
null_orig_clg 1 clid_transparent 0 callingNumber 1000
*Mar 1 00:11:19.907: dest pattern 2..., called 2000, digit_strip 0
*Mar 1 00:11:19.907: callingNumber=1000, calledNumber=2000, redirectNumber=
display_info= calling_oct3a=0
*Mar 1 00:11:19.907: accountNumber=, finalDestFlag=1,
guid=1d0d.9a0f.14f0.11cc.8008.b3df.433e.6402
*Mar 1 00:11:19.911: peer_tag=1
*Mar 1 00:11:19.911: ccIFCallSetupRequestPrivate: (vdbPtr=0x621D74DC, dest=,
callParams={called=2000,called_oct3=0x81, calling=1000,calling_oct3=0x0,
calling_xlated=false, subscriber_type_str=RegularLine, fdest=1,
voice_peer_tag=1}, mode=0x0) vdbPtr type = 1
*Mar 1 00:11:19.911: ccIFCallSetupRequestPrivate: (vdbPtr=0x621D74DC, dest=,
callParams={called=2000, called_oct3 0x81, calling=1000,calling_oct3 0x0,
calling_xlated=false, fdest=1, voice_peer_tag=1}, mode=0x0, xltrc=-5)
*Mar 1 00:11:19.911: ccSaveDialpeerTag (callID=0x3, dialpeer_tag=0x1)
*Mar 1 00:11:19.911: ccCallSetContext (callID=0x4, context=0x624C3094)
*Mar 1 00:11:19.911: ccCallReportDigits (callID=0x3, enable=0x0)
*Mar 1 00:11:19.911: cc_api_call_report_digits_done (vdbPtr=0x620B2DE8,
callID=0x3, disp=0)
*Mar 1 00:11:19.911: sess_appl: ev(52=CC_EV_CALL_REPORT_DIGITS_DONE),
cid(3), disp(0)
*Mar 1 00:11:19.911: cid(3)st(SSA_CS_CALL_SETTING)ev
(SSA_EV_CALL_REPORT_DIGITS_DONE)oldst(SSA_CS_MAPPING)
```

```
cfid(−1)csize(0)in(1)fDest(1)
*Mar 1 00:11:19.911: −cid2(4)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_MAPPING)
*Mar 1 00:11:19.911: ssaReportDigitsDone cid(3) peer list: (empty)
*Mar 1 00:11:19.911: ssaReportDigitsDone callid=3 Reporting disabled.
*Mar 1 00:11:19.947: cc_api_call_disconnected(vdbPtr=0x621D74DC,
callID=0x4, cause=0x1)
*Mar 1 00:11:19.947: sess_appl: ev(11=CC_EV_CALL_DISCONNECTED), cid(4), disp(0)
*Mar 1 00:11:19.947: cid(4)st(SSA_CS_CALL_SETTING)ev(SSA_EV_CALL_DISCONNECTED)
oldst(SSA_CS_MAPPING)cfid(−1)csize(0)in(0)fDest(0)
*Mar 1 00:11:19.947: −cid2(3)st2(SSA_CS_CALL_SETTING)oldst2(SSA_CS_CALL_SETTING)
*Mar 1 00:11:19.951: ssaDiscSetting
*Mar 1 00:11:19.951: ssa: Disconnected cid(4) state(1) cause(0x1)
*Mar 1 00:11:19.951: ccCallDisconnect (callID=0x4, cause=0x1 tag=0x0)
*Mar 1 00:11:19.951: ccCallDisconnect (callID=0x3, cause=0x1 tag=0x0)
*Mar 1 00:11:19.951: cc_api_call_disconnect_done(vdbPtr=0x620B2DE8, callID=0x3,
disp=0, tag=0x0)
*Mar 1 00:11:19.955: sess_appl: ev(12=CC_EV_CALL_DISCONNECT_DONE), cid(3),
disp(0)
*Mar 1 00:11:19.955: cid(3)st(SSA_CS_DISCONNECTING)ev
(SSA_EV_CALL_DISCONNECT_DONE)oldst(SSA_CS_CALL_SETTING)
cfid(−1)csize(0)in(1)fDest(1)
*Mar 1 00:11:19.955: −cid2(4)st2(SSA_CS_DISCONNECTING)oldst2(SSA_CS_CALL_SETTING)
*Mar 1 00:11:19.955: ssaDisconnectDone
*Mar 1 00:11:19.963: cc_api_icpif: expect factor = 0
*Mar 1 00:11:19.963: cc_api_call_disconnect_done(vdbPtr=0x621D74DC,
callID=0x4, disp=0, tag=0x0)
*Mar 1 00:11:19.967: sess_appl: ev(12=CC_EV_CALL_DISCONNECT_DONE),
cid(4), disp(0)
*Mar 1 00:11:19.967: cid(4)st(SSA_CS_DISCONNECTING)ev
(SSA_EV_CALL_DISCONNECT_DONE)oldst(SSA_CS_CALL_SETTING)
cfid(−1)csize(1)in(0)fDest(0)
*Mar 1 00:11:19.967: ssaDisconnectDone
```

Related Information

- **Configuring Connection PLAR for VoIP Gateways**
- **Troubleshooting and Debugging VoIP Call Basics**
- **VoIP Debug Commands**
- **Voice Technology Support**
- **Voice and IP Communications Product Support**
- **Troubleshooting Cisco IP Telephony**
- **Technical Support − Cisco Systems**

Updated: Jan 23, 2006 Document ID: 12432