

# Troubleshooting Voice Connection Trunks

Document ID: 23443

## Contents

### Introduction

#### Prerequisites

- Requirements
- Components Used
- Conventions

#### Problem

#### Solution

- Common Issues for Connection Trunks
- Begin to Troubleshoot
- Determine what Calls are Up
- DTMF Troubleshoot

#### Related Information

## Introduction

The voice connection trunks permanently establish voice calls, either Voice over IP (VoIP), Voice over Frame Relay (VoFR), or Voice over ATM (VoATM). The calls are established as soon as the router is turned up and the configuration is complete. As soon as the voice ports are turned up, the voice ports automatically dial the dummy phone number specified under the voice port and place a call to the location. The voice ports complete the call to the other end through the corresponding dial peers. Once this connection is established, as far as the router is concerned, the voice call is in session and is connected.

## Prerequisites

### Requirements

There are no specific prerequisites for this document.

### Components Used

This document is not restricted to specific software and hardware versions.

The information presented in this document was created from 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 before you use it.

### Conventions

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

## Problem

The common problems which pertain to the trunks are transparent to the router and very difficult to troubleshoot. The common issues seen with the voice trunks are manifested when a call is placed over the trunks and nothing is heard. This is one of the known problems with the connection trunks and is caused by

many different issues. Another issue is the Dual Tone Multifrequency (DTMF) tones that are not passed properly, and signaling from Private Branch Exchange (PBX) to PBX are not transported properly. This document troubleshoots these problems.

When the voice trunks are up and active, the signals behave differently in the connection trunks. Any commands that you normally issue under the voice port for signaling characteristics are not relevant and helpful. The voice trunk becomes a signaling conduit and relays the signal across the VoIP link. When you use the voice trunks, the PBX signaling must match end-to-end. As far as the two PBX machines are concerned, the goal is to make the voice trunk connection look identical to a leased T1 line to the PBX, with routers completely transparent while a clear link is established between the two PBXs in the whole process.

When the trunk comes up, the trunk becomes a software cable and the signal type is considered a connector type. The trunk does not care about the signal type that is used. The trunk still comes up even if the signal does not match at both ends. As long as the PBXs at both ends do the same signaling, the trunks function properly.

## Solution

The approach to take when you troubleshoot connection trunk issues is different than that is used for switched calls. To see what really happens after the trunks are verified, you need to look to the PBX signaling. Before you proceed to look at the signaling, verify that the trunks are up and that the Digital Signal Processors (DSPs) process the voice packets.

**Note:** You probably want to disable Voice Activity Detection (VAD) in order to troubleshoot. Once it is verified that the trunks function correctly, you need to look at the Telephony signaling in order to troubleshoot further.

If the trunks are established, and nobody tries to make a call, trunk keepalive messages are sent back and forth between the remote boxes. These keepalives verify the trunk connectivity and carry the signaling information from end-to-end. To verify these keepalives, issue the **debug vpm signal** command. If there are many trunks, the output from **debug vpm** commands, you can limit the output to a single port if you issue of the **debug vpm port x** command option, where "x" is the voice port in question. This is the output from the **debug vpm signal** command issued when you look at all the ports:

```
21:18:12: [3/0:10(11)] send to dsp sig DCBA state 0x0
21:18:12: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
21:18:12: [3/0:12(13)] rcv from dsp SIG DCBA state 0x0
21:18:12: [3/0:20(21)] rcv from dsp SIG DCBA state 0x0
21:18:12: [3/0:12(13)] send to dsp SIG DCBA state 0x0
21:18:12: [3/0:20(21)] send to dsp SIG DCBA state 0x0
21:18:12: [3/0:0(1)] send to dsp SIG DCBA state 0x0
21:18:12: [3/0:3(4)] rcv from dsp SIG DCBA state 0x0
21:18:12: [3/0:9(10)] rcv from dsp SIG DCBA state 0x0
21:18:12: [3/0:3(4)] send to dsp SIG DCBA state 0x0
21:18:13: [3/0:9(10)] send to dsp SIG DCBA state 0x0
21:18:13: [3/0:19(20)] rcv from dsp SIG DCBA state 0x0
```

If you limit this, with the **debug vpm port x** command, the debugs much easier to interpret, as shown in this example:

```
21:21:08: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
21:21:12: [3/0:0(1)] send to dsp SIG DCBA state 0x0
21:21:13: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
21:21:17: [3/0:0(1)] send to dsp SIG DCBA state 0x0
21:21:18: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
21:21:22: [3/0:0(1)] send to dsp SIG DCBA state 0x0
21:21:23: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
```

```
21:21:27: [3/0:0(1)] send to dsp SIG DCBA state 0x0
21:21:28: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
21:21:32: [3/0:0(1)] send to dsp SIG DCBA state 0x0
```

The keepalives are sent and received every five seconds. The terms "sent to dsp" and "received from dsp" are from the Cisco IOS<sup>®</sup> point of view. Substitute PBX for DSP to make it more understandable. These are the messages that are seen while there is no activity on the trunks. The keepalive messages let the routers on each end of the circuit know that the trunks are still up. When five of these messages are missed in a row, the trunk goes down. One of the causes is if the trunks flap constantly in a network. To verify whether the voice trunk keepalives are sent and received, issue the **debug vpm trunk-sc** command. This debug does not generate any output until the trunk keepalives are missed. This is an example of the **debug vpm trunk-sc** command output when keepalives are missed:

```
22:22:38: 3/0:22(23): lost Keepalive
22:22:38: 3/0:22(23): TRUNK_SC state : TRUNK_SC_CONN_WO_CLASS, event TRUNK_RTC_LOST_KEEPAL
22:22:38: 3/0:22(23): trunk_rtc_set_AIS on
22:22:38: 3/0:22(23): trunk_rtc_gen_pattern : SIG pattern 0x0
22:22:38: 3/0:22(23): TRUNK_SC, TRUNK_SC_CONN_WO_CLASS ==> TRUNK_SC_CONN_DEFAULT_IDLE
22:22:39: 3/0:13(14): lost Keepalive
22:22:39: 3/0:13(14): TRUNK_SC state : TRUNK_SC_CONN_WO_CLASS, event TRUNK_RTC_LOST_KEEPAL
22:22:39: 3/0:13(14): trunk_rtc_set_AIS on
22:22:39: 3/0:13(14): trunk_rtc_gen_pattern : SIG pattern 0x0
22:22:39: 3/0:13(14): TRUNK_SC, TRUNK_SC_CONN_WO_CLASS ==> TRUNK_SC_CONN_DEFAULT_IDLE
```

If no output is seen when the **debug vpm trunk-sc** command is issued, then no keepalives are missed. Even if keepalives are missed, the trunk stays up until five sequential messages are missed. This means that a connection must be down for 25 seconds before the trunks go down.

## Common Issues for Connection Trunks

There are several bugs associated with the voice trunk connections. Check these bugs if you see anything unusual. By the time Cisco IOS Software 12.2 was released, most of these issues had been addressed and integrated. You can look through the bugs to make yourself aware that these are causes of problems with older code. One of the most common issues is to get the PBXs to signal correctly over the trunk connection. It seems like a good idea to bring down the trunks and configure the routers so that they work at each end, but the approach is really counter-productive since anything that you changed now becomes moot once the trunks are established. The best way to troubleshoot is with the trunks up and functional.

## Begin to Troubleshoot

It is necessary to look at the basics to establish that these function correctly:

- Are the trunks established? Issue the **show voice call summary** command, and make sure that the trunks are in the S\_CONNECTED state.
- Are the DSPs processing packets? Issue the **show voice dsp** command to verify this. If you do not see packets are processed by the DSPs, it is because VAD is enabled and is suppresses the packets. Turn off VAD, re-establish trunks and look again. Also, check that the packet counters increment when the **show call active voice brief** command is issued. This command also shows whether VAD is enabled for the call log in question.

If the trunks connect to analog ports at any site, it is best to verify the operation of the PBX in non-trunked mode. To troubleshoot analog E&M connectivity problems, refer to Understanding and Troubleshooting Analog E&M Interface Types and Wiring Arrangements . Once everything is verified and functions correctly, bring the trunks up and look at the signaling that is passed between the PBXs.

The ideal way to troubleshoot voice trunk connection problems is to examine the signaling that is passed between the PBXs. It is best to have a Telnet session to each router in question so that the signaling can be observed as it is passed from one end to the other. This document uses E&M wink signaling since it is fairly popular and wink timing has to be taken into consideration.

This is the output from the router connected to the PBX that originates the call:

```
May 22 19:39:03.582: [3/0:0(1)] rcv from dsp sig DCBA state 0x0

!--- It is in idle state.

May 22 19:39:07.774: [3/0:0(1)] send to dsp SIG DCBA state 0x0

!--- ABCD bits=0000.

May 22 19:39:08.586: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:12.778: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:13.586: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:17.777: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:18.593: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:22.781: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:23.593: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:27.781: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:28.597: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:32.785: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:33.597: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:37.789: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:38.601: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:39.777: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:39.797: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:39.817: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF

!--- Receives off-hook from PBX, and passes to remote end.

May 22 19:39:39.837: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.857: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.877: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.897: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.917: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.937: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.957: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.977: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:39.997: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.017: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.037: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.057: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.077: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.089: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:40.097: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.109: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:40.117: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF

!--- Receiving wink from remote side, and passes to PBX.

May 22 19:39:40.129: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:40.137: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.149: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:40.157: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.169: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:40.177: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.189: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:40.197: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.213: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:40.217: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:40.229: [3/0:0(1)] send to dsp SIG DCBA state 0xF
```

May 22 19:39:40.237: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF  
May 22 19:39:40.249: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:40.257: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF  
May 22 19:39:40.269: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:40.289: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:40.309: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:40.329: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:40.349: [3/0:0(1)] send to dsp SIG DCBA state 0x0

*!--- Wink ended from remote side, and passes to PBX.*

May 22 19:39:40.369: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.389: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.409: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.429: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.449: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.469: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.493: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.509: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.529: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.549: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.569: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.589: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.613: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.629: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.649: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.669: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.689: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.709: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.729: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.749: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:40.769: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:45.773: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:50.081: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:50.101: [3/0:0(1)] send to dsp SIG DCBA state 0x0  
May 22 19:39:50.121: [3/0:0(1)] send to dsp SIG DCBA state 0xF

*!--- Wink ends, the remote end is now off-hook, the conversation happens.*

May 22 19:39:50.141: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.161: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.181: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.197: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.221: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.241: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.261: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.261: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF  
May 22 19:39:50.281: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.301: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.321: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.341: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.361: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.381: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.401: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.421: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.441: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.461: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.481: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.501: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.521: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.541: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:50.561: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:39:55.265: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF  
May 22 19:39:55.561: [3/0:0(1)] send to dsp SIG DCBA state 0xF  
May 22 19:40:00.269: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF  
May 22 19:40:00.565: [3/0:0(1)] send to dsp SIG DCBA state 0xF

```
May 22 19:40:05.268: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:05.564: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:10.272: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:10.568: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:15.276: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:15.572: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:19.676: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:19.696: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:19.716: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.736: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.756: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.776: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.796: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.796: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:19.816: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.816: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:19.836: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.836: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
```

*!--- Both side hung up, back to idle state.*

```
May 22 19:40:19.856: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.856: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.876: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.876: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.896: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.896: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.916: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.916: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.936: [3/0:0(1)] send to dsp SIG DCBA state 0x0
```

This output shows the router terminates the call. Network Time Protocol (NTP) is synced.

```
May 22 19:39:03.582: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:07.774: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
```

*!--- Idle state, both side on-hook.*

```
May 22 19:39:08.586: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:12.774: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:13.586: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:15.383: [1/0:0(1)] Signaling RTP packet has no particle
```

*!--- You will see this message if you are running Cisco IOS  
!--- Software Release 12.2(1a) or later. It is not an error  
!--- message, it is a normal functioning state.*

```
May 22 19:39:17.774: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:18.590: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:22.778: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:23.594: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:27.782: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:28.598: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:32.782: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:33.598: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:37.786: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:39:38.602: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:39.778: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:39.798: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:39:39.818: [3/0:0(1)] send to dsp SIG DCBA state 0xF
```

*!--- Remote side off-hook, this is conveyed to the PBX.*

```
May 22 19:39:39.838: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:39.858: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:39.878: [3/0:0(1)] send to dsp SIG DCBA state 0xF
```



*!--- Receive off-hook from PBX.*

```
May 22 19:39:50.137: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.157: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.177: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.197: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.217: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.237: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.257: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.261: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:50.277: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.297: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.317: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.337: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.357: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.377: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.397: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.417: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.437: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.457: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.477: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.497: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.517: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.537: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:39:50.557: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
```

*!--- Both sides off-hook, the conversation happens.*

```
May 22 19:39:55.265: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:39:55.557: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:00.269: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:00.561: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:05.269: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:05.561: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:10.273: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:10.565: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:15.273: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:15.569: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:19.673: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:19.693: [3/0:0(1)] rcv from dsp SIG DCBA state 0xF
May 22 19:40:19.713: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.733: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.753: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.773: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.793: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.797: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:19.813: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.817: [3/0:0(1)] send to dsp SIG DCBA state 0xF
May 22 19:40:19.833: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.837: [3/0:0(1)] send to dsp SIG DCBA state 0x0
```

*!--- Both sides are back on-hook, back to idle.*

```
May 22 19:40:19.853: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.857: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.873: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.877: [3/0:0(1)] send to dsp SIG DCBA state 0x0
May 22 19:40:19.893: [3/0:0(1)] rcv from dsp SIG DCBA state 0x0
May 22 19:40:19.897: [3/0:0(1)] send to dsp SIG DCBA state 0x0
```

**Note:** This output shows the signaling that occurs on both sides of a voice trunk which uses E&M wink signaling. Other types of signaling can be seen that uses these same debugs. If you see calls established correctly (as shown here), two-way audio must be present. This can be verified if you look at either the **show voice dsp** or the **show call active voice brief** command output. If everything looks to be fine there, and you are getting audio problems (no audio or one-way) with analog connections, check these connections again.



## Determine what Calls are Up

Since it does little or no good to look at the **show call active voice** or **show voice call summary** command output for trunked calls, you need a simple method to determine which voice trunks support active calls. One of the easiest ways to do this is to issue the **show voice trunk-conditioning signaling** command in conjunction with the *include* parameter and use *ABCD* as the included string, as shown in here:

```
Phoenix#show voice trunk-conditioning signaling | include ABCD
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=1111, last-RX-ABCD=0000

!--- Timeslot 8.

last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=1111, last-RX-ABCD=1111

!--- Timeslot 10.

last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
last-TX-ABCD=0000, last-RX-ABCD=0000
```

**Note:** This output shows a call active on timeslot 10 and another call being started on timeslot eight. You want to make an alias for this rather long command if you use it a lot.

## DTMF Troubleshoot

Apart from the off-hook and on-hook signaling, the only other thing that the routers pass between the PBXs (besides voice) are DTMF tones. There is also an audio path so this is usually not a problem, but there is a problem. The problem arises with how you do audio over that path. It is sometimes preferable to use the low bit rate codecs in order to save bandwidth. The issue comes up that these low bit rate codecs are designed by means of algorithms that were written for human speech. DTMF tones do not conform to these algorithms very well and need some other method to convey unless the customer uses g711 codec. The answer lies in the **dtmf-relay** command. This feature allows the DSPs at the end, starts the tone, to recognize the DTMF tone and separate it from the regular audio stream. Based upon how it is configured, the DSP then codes this tone as either a different type of Real Time Protocol (RTP) packet or as an h245 message to be sent across the link separately from the audio stream. This is the same process behind the **fax-relay** and **modem-relay** commands.

This feature poses another debug problem for trunk troubleshooting. How do you verify what digits are passed if there is no call setup and you have to extract that information from the packet stream between the routers? How to do this depends upon what type of **dtmf-relay** command is employed.

As shown in this example, the **dtmf-relay cisco-rtp** command, uses a proprietary Cisco payload type, so you must look down at the DSPs to see this. You can issue the **debug vpm signal** command in conjunction with the **debug vpm port x/x:y.z** command (to limit output to the port in question) to see the digits passed to the DSPs at the originating side. This output is displayed at the originating side, not at the terminating side.

```
*Mar 1 00:22:39.592: htsp_digit_ready: digit = 31
```

```

*Mar 1 00:22:39.592: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:40.021: htsp_digit_ready: digit = 32
*Mar 1 00:22:40.021: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:40.562: htsp_digit_ready: digit = 33
*Mar 1 00:22:40.562: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:40.810: [1/0:1(2)] rcv from dsp SIG DCBA state 0xF
*Mar 1 00:22:41.131: htsp_digit_ready: digit = 34
*Mar 1 00:22:41.131: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:41.499: [1/0:1(2)] Signaling RTP packet has no partical
*Mar 1 00:22:41.499: [1/0:1(2)] send to dsp SIG DCBA state 0xF
*Mar 1 00:22:41.672: htsp_digit_ready: digit = 35
*Mar 1 00:22:41.672: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:42.192: htsp_digit_ready: digit = 36
*Mar 1 00:22:42.192: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:42.789: htsp_digit_ready: digit = 37
*Mar 1 00:22:42.789: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:43.350: htsp_digit_ready: digit = 38
*Mar 1 00:22:43.350: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:44.079: htsp_digit_ready: digit = 39
*Mar 1 00:22:44.079: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:45.249: htsp_digit_ready: digit = 30
*Mar 1 00:22:45.249: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:45.810: [1/0:1(2)] rcv from dsp SIG DCBA state 0xF
*Mar 1 00:22:46.007: htsp_digit_ready: digit = 2A
*Mar 1 00:22:46.011: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:46.572: [1/0:1(2)] Signaling RTP packet has no partical
*Mar 1 00:22:46.572: [1/0:1(2)] send to dsp SIG DCBA state 0xF
*Mar 1 00:22:46.628: htsp_digit_ready: digit = 23
*Mar 1 00:22:46.628: [1/0:1(2), S_TRUNKED, E_VTSP_DIGIT]
*Mar 1 00:22:50.815: [1/0:1(2)] rcv from dsp SIG DCBA state 0xF
all digits 0-9 are represented by 30-39, * = 2A and # = 23.

```

You can verify what digits are sent from the originating side with the **dtmf-relay h245-alphanumeric** command. The **dtmf-relay h245-alphanumeric** command uses the alphanumeric portion of h.245 to convey the tones. As shown in this example, the digits can easily be seen at both the originating and the terminating sides of the trunk when the **debug h245 asn1** command is enabled:

#### Originating side:

```

*Mar 1 00:34:17.749: H245 MSC OUTGOING PDU ::=
value MultimediaSystemControlMessage ::= indication : userInput : alphanumeric : "1"

*Mar 1 00:34:17.749: H245 MSC OUTGOING ENCODE BUFFER::= 6D 400131
*Mar 1 00:34:17.753:
*Mar 1 00:34:18.350: H245 MSC OUTGOING PDU ::=
value MultimediaSystemControlMessage ::= indication : userInput : alphanumeric : "2"

*Mar 1 00:34:18.350: H245 MSC OUTGOING ENCODE BUFFER::= 6D 400132
*Mar 1 00:34:18.350:
*Mar 1 00:34:18.838: H245 MSC OUTGOING PDU ::=
value MultimediaSystemControlMessage ::= indication : userInput : alphanumeric : "3"

*Mar 1 00:34:18.838: H245 MSC OUTGOING ENCODE BUFFER::= 6D 400133

```

#### Terminating side:

```

*Mar 1 17:45:16.424: H245 MSC INCOMING ENCODE BUFFER::= 6D 400131
*Mar 1 17:45:16.424:
*Mar 1 17:45:16.424: H245 MSC INCOMING PDU ::=
value MultimediaSystemControlMessage ::= indication : userInput : alphanumeric : "1"

*Mar 1 17:45:17.025: H245 MSC INCOMING ENCODE BUFFER::= 6D 400132
*Mar 1 17:45:17.025:
*Mar 1 17:45:17.025: H245 MSC INCOMING PDU ::=


```

```
value MultimediaSystemControlMessage ::= indication : userInput : alphanumeric : "2"

*Mar 1 17:45:17.514: H245 MSC INCOMING ENCODE BUFFER::= 6D 400133
*Mar 1 17:45:17.514:
*Mar 1 17:45:17.514: H245 MSC INCOMING PDU ::=
value MultimediaSystemControlMessage ::= indication : userInput : alphanumeric : "3"
```

The **dtmf-relay h245-signal** command is very similar and can be seen when the same debugs as the **dtmf-relay h245-alphanumeric** command are used. Overall, to troubleshoot the connection trunks with the **dtmf-relay** command is rather difficult without the debugs mentioned.

## Related Information

- [Configuring and Troubleshooting Transparent CCS](#)
- [Voice Technology Support](#)
- [Voice and IP Communications Product Support](#)
- [Troubleshooting Cisco IP Telephony](#) 
- [Technical Support – Cisco Systems](#)

---

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2014 – 2015 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

---

Updated: Feb 02, 2006

Document ID: 23443

---