

# TDM Switching of Voice and Data Calls on AS5400 Gateways Configuration Example

Document ID: 24644

## Contents

### Introduction

#### Prerequisites

- Requirements
- Components Used
- Conventions

#### Background Information

#### Configure

- Network Diagram
- Configurations

#### Verify

#### Troubleshoot

- Troubleshooting Commands

#### Related Information

## Introduction

This document details the theory and configuration of Time Division Multiplex (TDM) switching on the Cisco AS5400 platform.

## Prerequisites

### Requirements

It is assumed that the reader has a basic understanding of ISDN call signaling and the distribution of synchronous clock sources on TDM networks. Some background information on TDM clocking is provided in this document. Familiarity with Cisco IOS® configuration and debugging commands is also helpful.

### Components Used

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

- Cisco AS5400, AS5350, and AS5850 platforms
- Cisco IOS Software Release 12.2.2XB5 with IP Plus feature set

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

Refer to Cisco Technical Tips Conventions for more information on document conventions.

# Background Information

Fundamental to TDM switching is a synchronized clock source shared across all configured interfaces. If the clock reference is different across ports, the access server registers clock slips, which can be unnoticeable with voice calls, but almost certainly causes fax or modem calls to fail. Therefore, it is critical that the external devices (PBXes or central office (CO) switches) the access server interfaces with are synchronized to a common master clock reference. Most Telcos or service providers subscribe to, or originate a Stratum 1 clock reference and propagate this across their networks. Therefore, in the majority of cases, the clocking maintains synchronization even between different service providers. If all the configured T1/E1 interfaces on the access server maintain synchronization then there should be little likelihood of interface errors.

The TDM switching routes a call based on the Dialed Number Identification Service (DNIS) of the incoming call. Once the access server receives an incoming ISDN Q.931 setup message, it can then determine where the call is to be sent, select an appropriate bearer channel on the outgoing interface, and send an ISDN Q.931 setup message to signal the downstream device the presence of a new call. Once the terminating device sends an ISDN Q.931 connect message, the access server cross connects the pulse code modulation (PCM) streams across the backplane. As detailed in the previous paragraph, the two attached networks must have the same clock synchronization to ensure error free switching of the PCM audio streams or digital data from one interface to another. The network diagram shows the general concepts of the ISDN call coming in on particular PRI interfaces and being switched out through other interfaces based on matches made on configured plain old telephone service (POTS) dial-peers. If necessary, the called/calling numbers can be manipulated using IOS translation rules.

## Configure

AS5400 platforms are normally installed as dial-in data, voice, fax, or modem access servers. In order to terminate speech type calls (voice, fax, or modem) the access server needs appropriate any service, any port (ASAP) Digital Signal Processor (DSP) resources to be installed.

If the modem, fax, or voice calls are not actually required to be terminated on the access server, but for some reason need to be switched back out to alternate ports, it is possible to configure the AS5400 to act in a purely TDM application where voice call switching is controlled via ISDN D-channel signaling. Data or speech calls can be switched based on the DNIS (called number) through to another interface. Effectively, the access server becomes a TDM voice/data switch. This feature is often called TDM switching, although other names such as hairpinning, tromboning, or dial-grooming are also applied to the technique. Generally, the terms are interchangeable and for this document, the term TDM switching is used. There are no dual tone multifrequency (DTMF) or multifrequency (MF) signaling tones passed with ISDN. Call control is done with High-Level Data Link Control (HDLC) encapsulated D-channel messages. Therefore, there is no need for the DSP resources for voice calls when in the TDM mode of operation.

The access server uses an incoming DNIS (called number) to match on an outgoing POTS dial-peer destination pattern and routes the call out an appropriate port. It is possible to use IOS translation rules to manipulate the called and calling numbers for call routing decisions as well.

Applications of TDM switching can include an access server acting as a small ISDN data/voice exchange (using ISDN network side protocol emulation), or call rerouting via alternate carriers (least cost).

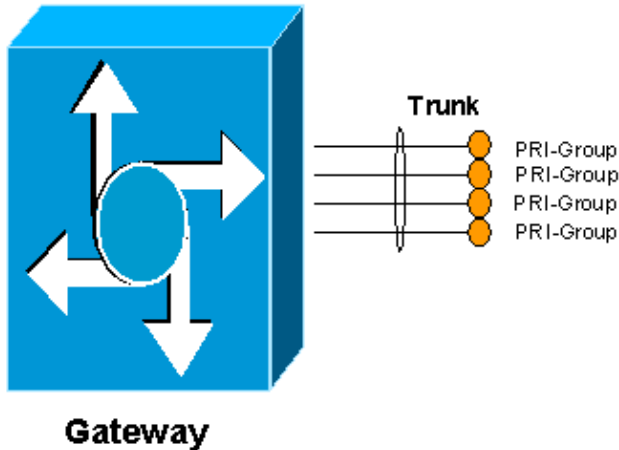
This document describes how to configure an AS5400 to perform TDM switching for voice and data calls. Based on matches made on the DNIS for the incoming call (provided in the ISDN Q.931 setup message), the call is switched from one interface to an alternate interface. The technique also works on other platforms that use TDM backplanes such as the AS5350 and the AS5850.

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

**Note:** Use the Command Lookup Tool (registered customers only) to find more information on the commands used in this document.

## Network Diagram

This document uses this network setup:



## Configurations

In order to allow the access server to perform TDM switching, resource pooling must be enabled and the available bearer channel resources placed into a pool. This pool of bearer channels is then tied to a DNIS group, which allows particular pools of resources to be associated with either certain incoming called numbers, or any incoming called numbers. This output shows an example:

```
AS5400
!
resource-pool enable
!
resource-pool group resource TDM_Voice
range limit 124

!--- Up to 124 speech channels can be switched.

!
resource-pool group resource TDM_Data
range limit 124

!--- Up to 124 data channels can be switched.

!
!
resource-pool profile customer TDM_Switching
limit base-size all
limit overflow-size 0
resource TDM_Voice speech

!--- Resources for speech calls.

!
resource TDM_Data digital
! resources for data calls
!
```

```
dnis group default

!--- Default DNIS group matches all called numbers.

!
```

The resource pooling feature must be enabled in order to allow TDM switching. A resource group called TDM\_Voice is defined which allows up to 124 channels available for speech calls. A second resource group called TDM\_Data allows up to 124 channels for data calls. These numbers are derived from the maximum number of E1 or T1 ports on the system. For example, an 8 port E1 card has 30 bearer channels plus one signaling channel per interface (31 channels) for 8 ports. The total is 248 (31 multiplied by 8). Half for data and half for voice calls is allocated here.

The resource group TDM\_Voice is then placed in a profile called TDM\_Switching and call types are defined as speech, while the resource group TDM\_Data is defined as digital. This effectively allows calls with speech and data bearer capabilities through the access server. The **dnis group default** command allows all incoming called numbers to be matched. It is possible to define DNIS groups that match on more specific called numbers. Refer to the Universal Port Resource Pooling for Voice and Data Services configuration guide for more information.

If necessary, the called numbers can be manipulated to prepend access codes as the call comes in on a particular port. For example:

```
AS5400

!
translation-rule 1
Rule 1 ^.% 555

!--- Match on any string, prepend with 555.

!
voice-port 6/0:D
translate called 1

!--- Apply translation rule 1 to port 6/0 so any
!--- incoming call is prepended with 555.

compand-type a-law
!
voice-port 6/3:D
compand-type a-law

!--- The translated called number is matched on POTS dial-peers
!--- to determine where it should be routed.

dial-peer voice 1 pots
description - enable DID (single stage dialing) on port 6/0
incoming called-number .
direct-inward-dial
port 6/0:D
!
dial-peer voice 2 pots
description - reroute calls from 6/0 to 6/3
destination-pattern 55598842304
port 6/3:D
prefix 0401890165
!
```

When a call comes in on port 6/0, it is prepended with 555. If the original called number is 98842304, the translated number becomes 55598842304 and matches on dial-peer 2. The call is then sent out on port 6/3. Since it is an explicit match, the original called number is stripped off and the prefix command replaces it with 0401890165.

Data call switching works in the same way. A POTS dial-peer matches on a called number and directs it out another port. As an example, if a call comes in on port 6/4 with a called number of 5551000, it is switched out port 6/7 with a new called number of 5552000. Likewise, if the call comes in on port 6/7 with a called number of 5552000, it is switched out on port 6/4 with a new called number of 5551000.

```
AS5400
!
dial-peer voice 3 pots
description - enable DID on port 6/4
incoming called-number
direct-inward-dial
port 6/4:D
!
dial-peer voice 4 pots
description - enable DID on port 6/7
incoming called-number
direct-inward-dial
port 6/7:D
!
dial-peer voice 12 pots
description - reroute calls from 6/4 to 6/7
destination-pattern 5551000
port 6/7:D
prefix 5552000
!
dial-peer voice 13 pots
description - reroute calls from 6/7 to 6/4
destination-pattern 5552000
port 6/4:D
prefix 5551000
!
```

## Verify

Use this section to confirm that your configuration works properly.

The Output Interpreter Tool (registered customers only) (OIT) supports certain **show** commands. Use the OIT to view an analysis of **show** command output.

- **show run** Displays the full configuration of the access server that operates as a TDM switch.

```
multi-5-19#show run
Building configuration...

Current configuration : 3110 bytes
!
! Last configuration change at 13:18:39 UTC Wed Jun 19 2002
! NVRAM config last updated at 20:45:12 UTC Sat Jan 8 2000
!
version 12.2
service timestamps debug datetime msec localtime
service timestamps log uptime
no service password-encryption
!
hostname multi-5-19
```

```
!  
enable password cisco  
!  
!  
!  
resource-pool enable  
!  
resource-pool group resource TDM_Voice  
range limit 124  
!  
resource-pool group resource TDM_Data  
range limit 124  
!  
resource-pool profile customer TDM_Switching  
limit base-size all  
limit overflow-size 0  
resource TDM_Data digital  
resource TDM_Voice speech  
dnis group default  
dial-tdm-clock priority 1 6/0  
!  
!  
!  
!  
ip subnet-zero  
ip cef  
!  
isdn switch-type primary-net5  
!  
!  
!  
!  
!  
!  
fax interface-type fax-mail  
mta receive maximum-recipients 0  
!  
controller E1 6/0  
pri-group timeslots 1-31  
!  
controller E1 6/1  
!  
controller E1 6/2  
!  
controller E1 6/3  
pri-group timeslots 1-31  
!  
controller E1 6/4  
pri-group timeslots 1-31  
!  
controller E1 6/5  
!  
controller E1 6/6  
!  
controller E1 6/7  
pri-group timeslots 1-31  
!  
translation-rule 1  
Rule 1 ^.% 555  
!  
translation-rule 2  
Rule 2 ^.% 666  
!  
!  
!
```

```
interface FastEthernet0/0
no ip address
duplex auto
speed auto
!
interface FastEthernet0/1
no ip address
duplex auto
speed auto
!
interface Serial0/0
no ip address
shutdown
clockrate 2000000
!
interface Serial0/1
no ip address
shutdown
clockrate 2000000
!
interface Serial6/0:15
no ip address
isdn switch-type primary-net5
isdn incoming-voice modem
no cdp enable
!
interface Serial6/3:15
no ip address
isdn switch-type primary-net5
isdn incoming-voice modem
no cdp enable
!
interface Serial6/4:15
no ip address
isdn switch-type primary-net5
isdn protocol-emulate network
no cdp enable
!
interface Serial6/7:15
no ip address
isdn switch-type primary-net5
isdn protocol-emulate network
no cdp enable
!
interface Group-Async0
physical-layer async
no ip address
!
ip classless
!
no ip http server
!
!
!
call rsvp-sync
!
voice-port 6/0:D
translate called 1
compand-type a-law
!
voice-port 6/3:D
translate called 2
compand-type a-law
!
voice-port 6/4:D
compand-type a-law
```

```
!  
voice-port 6/7:D  
compand-type a-law  
!  
!  
mgcp profile default  
!  
dial-peer cor custom  
!  
!  
!  
dial-peer voice 1 pots  
incoming called-number  
direct-inward-dial  
port 6/0:D  
!  
dial-peer voice 2 pots  
incoming called-number  
direct-inward-dial  
port 6/3:D  
!  
dial-peer voice 10 pots  
destination-pattern 55598842304  
port 6/3:D  
prefix 94344600  
!  
dial-peer voice 11 pots  
destination-pattern 66698842305  
port 6/0:D  
prefix 94344600  
!  
dial-peer voice 3 pots  
incoming called-number  
direct-inward-dial  
port 6/4:D  
!  
dial-peer voice 4 pots  
incoming called-number  
direct-inward-dial  
port 6/7:D  
!  
dial-peer voice 12 pots  
destination-pattern 5551000  
port 6/7:D  
prefix 5552000  
!  
dial-peer voice 13 pots  
destination-pattern 5552000  
port 6/4:D  
prefix 5551000  
!  
!  
line con 0  
line aux 0  
line vty 0 4  
password cisco  
login  
!  
scheduler allocate 10000 400  
ntp master  
end
```

multi-5-19#



# Troubleshoot

Use this section to troubleshoot your configuration.

## Troubleshooting Commands

The Output Interpreter Tool (registered customers only) (OIT) supports certain **show** commands. Use the OIT to view an analysis of **show** command output.

When you troubleshoot ISDN trunks, you can make B-channels busy out. Issue the **ds0 busyout X** command under controller configuration mode for the CAS trunk.

```
Router(config-controller)#ds0 busyout X
```

In order to busy out CCS or PRI ISDN trunks use the **isdn service b\_channel X state 2** command under interface configuration mode.

For T1:

```
Router(config)#interface serial 0:23
```

For E1:

```
Router(config)#interface serial 0:15
```

```
Router(config-if)#isdn service b_channel X state 2
```

The valid states are 0=Inservice, 1=Maint, 2=Outofservice, and X is the B-channel number in both CCS and CAS configurations.

The **show isdn service** command can be used in order to find the state of each B-channel.

**Note:** Refer to Important Information on Debug Commands before you use **debug** commands.

Normal ISDN and IOS translation rule debugs can be used in order to troubleshoot TDM switching.

- **debug translation detailed** Displays information about the operation of the IOS translation rules so that digit manipulations of called or calling numbers can be monitored.
- **debug isdn q931** Displays information about call setup and teardown of ISDN network connections (Layer 3) between the local router (user side) and the network.

These command outputs are traces for **debug translation detailed** (IOS translation rule debugging) and **debug isdn q931** enabled for a speech call on port 6/0 that is switched to port 6/3.

```
multi-5-19#debug translation detailed
*Jan 1 00:20:53.215: ISDN Se6/0:15: RX <- SETUP pd = 8 callref = 0x1D79
*Jan 1 00:20:53.215: Bearer Capability i = 0x8090A3
*Jan 1 00:20:53.215: Channel ID i = 0xA18395
*Jan 1 00:20:53.215: Called Party Number i = 0x80, '98842304',
Plan:Unknown, Type:Unknown

!--- Receive a setup message on interface 6/0:15 for a
!--- speech call with a called number of 98842304.
!--- Speech call is indicated by the bearer capability of 0x8090A3 :
!--- 64 Kbps A-law PCM audio/speech.
```

*!--- IOS Translation rule number 1 prepends '555' to the original  
!--- called number when it passes through port 6/0.*

```
*Jan 1 00:20:53.219: xrule_checking
*Jan 1 00:20:53.219: xrule_checking calling , called 98842304
*Jan 1 00:20:53.219: xrule_checking peer_tag 0, direction 1, protocol 6
*Jan 1 00:20:53.219: xrule_translation
*Jan 1 00:20:53.219: xrule_translation callednumber 98842304, strlen 8
*Jan 1 00:20:53.219: xrule_translation callednumber 98842304 xruleCalledTag=1
*Jan 1 00:20:53.219: xrule_translation called Callparms Numpertype 0x80, match_type 0x0
*Jan 1 00:20:53.219: xrule_translation Xrule index 1, Numpertype 0x9
*Jan 1 00:20:53.219: dpMatchString, target_number 98842304, match_number ^.%
*Jan 1 00:20:53.219: dpMatchString match_tmp , match_len 0
*Jan 1 00:20:53.219: dpMatchString beginning_replace 0, match_tmp ,target 98842304
*Jan 1 00:20:53.219: dpMatchString 1. target 98842304,match_tmp
*Jan 1 00:20:53.219: dpMatchString 1.1 compare_len 0, target 98842304, match_tmp
*Jan 1 00:20:53.219: dpMatchString 5. match_len=compare_len 0, target 98842304
*Jan 1 00:20:53.219: replace_string
*Jan 1 00:20:53.219: replace_string match ^.%, replace 555
*Jan 1 00:20:53.219: translation_format replace_rule ^.%, strip_proceeding 0
*Jan 1 00:20:53.219: replace_string match_tmp ^.%, strip_proceeding 0
*Jan 1 00:20:53.219: replace_string match_tmp
*Jan 1 00:20:53.219: replace_string direction 1, callparty 2
*Jan 1 00:20:53.219: replace_string direction 1, callparty 2, target 98842304
*Jan 1 00:20:53.219: replace_string match_tmp ,replace 555
*Jan 1 00:20:53.219: replace_string2.replacel,target98842304,current98842304,match_tmp
*Jan 1 00:20:53.219: replace_string2.1 compare_len 0,match_len 0
*Jan 1 00:20:53.219: replace_string 3. replacel , compare_len 0
*Jan 1 00:20:53.219: replace_string 4. replacel 5,compare_len -1,replace 55
*Jan 1 00:20:53.219: replace_string 4. replacel 55,compare_len -2,replace 5
*Jan 1 00:20:53.219: replace_string 4. replacel 555,compare_len -3,replace
*Jan 1 00:20:53.219: replace_string 5.replacel 555, compare_len -3,match_len 0
*Jan 1 00:20:53.219: replace_string 6. replacel 555,compare_len -3,current 98842304
*Jan 1 00:20:53.219: replace_string 7. replacel 5559
*Jan 1 00:20:53.219: replace_string 7. replacel 55598
*Jan 1 00:20:53.219: replace_string 7. replacel 555988
*Jan 1 00:20:53.219: replace_string 7. replacel 5559884
*Jan 1 00:20:53.219: replace_string 7. replacel 55598842
*Jan 1 00:20:53.219: replace_string 7. replacel 555988423
*Jan 1 00:20:53.219: replace_string 7. replacel 5559884230
*Jan 1 00:20:53.219: replace_string 7. replacel 55598842304
*Jan 1 00:20:53.219: replace_string buffer 55598842304
*Jan 1 00:20:53.219: xrule_translation index 1,xrule_number 55598842304, callparty 2
*Jan 1 00:20:53.219: xrule_translation Return rc = 0
*Jan 1 00:20:53.219: xrule_checking Return rc = 0
*Jan 1 00:20:53.223: ISDN Se6/0:15: TX -> CALL_PROC pd = 8 callref = 0x9D79
*Jan 1 00:20:53.223: Channel ID i = 0xA98395
```

*!--- Send a call proceeding back to the ISDN.*

```
*Jan 1 00:20:53.227: ISDN Se6/3:15: TX -> SETUP pd = 8 callref = 0x0005
*Jan 1 00:20:53.227: Bearer Capability i = 0x8090A3
*Jan 1 00:20:53.227: Channel ID i = 0xA9839F
*Jan 1 00:20:53.227: Called Party Number i = 0x80, '0401890165', Plan:Unknown, Type:Unknown
```

*!--- Match has been made on outgoing POTS dial-peer  
!--- and a new call is sent out on 6/3:15.*

```
*Jan 1 00:20:53.371: ISDN Se6/3:15: RX <- CALL_PROC pd = 8 callref = 0x8005
*Jan 1 00:20:53.371: Channel ID i = 0xA1839F
*Jan 1 00:20:53.371: ISDN Se6/3:15: RX <- ALERTING pd = 8 callref = 0x8005
```

*!--- Receive alerting on the second (outgoing) call leg.*

```
*Jan 1 00:20:53.375: ISDN Se6/0:15: TX -> ALERTING pd = 8 callref = 0x9D79
```

```

*Jan 1 00:20:53.375: Progress Ind i = 0x8188 - In-band info or appropriate now available

!--- Send alerting on the first (incoming) call leg.

*Jan 1 00:21:00.095: ISDN Se6/3:15: RX <- CONNECT pd = 8 callref = 0x8005
*Jan 1 00:21:00.095: ISDN Se6/3:15: TX -> CONNECT_ACK pd = 8 callref = 0x0005
*Jan 1 00:21:00.099: ISDN Se6/0:15: TX -> CONNECT pd = 8 callref = 0x9D79
*Jan 1 00:21:00.247: ISDN Se6/0:15: RX <- CONNECT_ACK pd = 8 callref = 0x1D79

!--- Both calls connect.

*Jan 1 00:21:00.247: ISDN Se6/0:15: CALL_PROGRESS:CALL_CONNECTED call id 0x5, bchan 20, ds
*Jan 1 00:21:37.591: ISDN Se6/0:15: RX <- DISCONNECT pd = 8 callref = 0x1D79
*Jan 1 00:21:37.591: Cause i = 0x8290 - Normal call clearing

!--- Receive a disconnect on incoming call leg.

*Jan 1 00:21:37.595: ISDN Se6/0:15: TX -> RELEASE pd = 8 callref = 0x9D79
*Jan 1 00:21:37.599: ISDN Se6/3:15: TX -> DISCONNECT pd = 8 callref = 0x0005
*Jan 1 00:21:37.599: Cause i = 0x8090 - Normal call clearing

!--- Send a disconnect on the outgoing call leg.

*Jan 1 00:21:37.631: ISDN Se6/0:15: RX <- RELEASE_COMP pd = 8 callref = 0x1D79
*Jan 1 00:21:37.723: ISDN Se6/3:15: RX <- RELEASE pd = 8 callref = 0x8005
*Jan 1 00:21:37.723: Cause i = 0x8290 - Normal call clearing
*Jan 1 00:21:37.723: ISDN Se6/3:15: TX -> RELEASE_COMP pd = 8 callref = 0x0005

!--- Both calls have cleared.

```

This is the command output from the **debug isdn q931** command. These traces show an ISDN data call switched from port 6/4 to port 6/7.

```

Jun 19 13:36:02.091: ISDN Se6/4:15: RX <- SETUP pd = 8 callref = 0x0005
Jun 19 13:36:02.091: Bearer Capability i = 0x8890
Jun 19 13:36:02.091: Channel ID i = 0xA9839F
Jun 19 13:36:02.095: Called Party Number i = 0x81, '5551000', Plan:ISDN, Type:Unknown

!--- Call comes in on port 6/4 for 5551000. Bearer Capability
!--- is 0x8890, which indicates 64 K data call.

Jun 19 13:36:02.095: ISDN Se6/4:15: TX -> CALL_PROC pd = 8 callref = 0x8005
Jun 19 13:36:02.095: Channel ID i = 0xA9839F

Jun 19 13:36:02.099: ISDN Se6/7:15: TX -> SETUP pd = 8 callref = 0x0085
Jun 19 13:36:02.099: Bearer Capability i = 0x8890
Jun 19 13:36:02.099: Channel ID i = 0xA98381
Jun 19 13:36:02.099: Called Party Number i = 0x81, '5552000', Plan:ISDN, Type:Unknown

!--- Redirect the call out on port 6/7, (new) called
!--- number is 5552000 with data bearer capability.

Jun 19 13:36:02.155: ISDN Se6/7:15: RX <- CALL_PROC pd = 8 callref = 0x8085
Jun 19 13:36:02.155: Channel ID i = 0xA98381
Jun 19 13:36:02.159: ISDN Se6/7:15: RX <- CONNECT pd = 8 callref = 0x8085
Jun 19 13:36:02.159: Channel ID i = 0xA98381

!--- Second call leg connects.

Jun 19 13:36:02.159: ISDN Se6/7:15: TX -> CONNECT_ACK pd = 8 callref = 0x0085
Jun 19 13:36:02.163: ISDN Se6/4:15: CALL_PROGRESS:CALL_CONNECTED call id 0x7,bchan 30, dsl
Jun 19 13:36:02.163: ISDN Se6/4:15: TX -> CONNECT pd = 8 callref = 0x8005

!--- First call leg connects.

```

Jun 19 13:36:02.215: ISDN Se6/4:15: RX <- CONNECT\_ACK pd = 8 callref = 0x0005  
Jun 19 13:38:12.783: ISDN Se6/4:15: RX <- DISCONNECT pd = 8 callref = 0x0005  
Jun 19 13:38:12.783: Cause i = 0x8090 - Normal call clearing

*!--- Remote device drops the call, first call leg disconnects.*

Jun 19 13:38:12.787: ISDN Se6/4:15: TX -> RELEASE pd = 8 callref = 0x8005  
Jun 19 13:38:12.787: ISDN Se6/7:15: TX -> DISCONNECT pd = 8 callref = 0x0085  
Jun 19 13:38:12.787: Cause i = 0x8290 - Normal call clearing

*!--- Second call leg is dropped.*

Jun 19 13:38:12.807: ISDN Se6/7:15: RX <- RELEASE pd = 8 callref = 0x8085  
Jun 19 13:38:12.851: ISDN Se6/4:15: RX <- RELEASE\_COMP pd = 8 callref = 0x0005

*!--- Both calls have cleared.*

## Related Information

- **Universal Port Resource Pooling for Voice and Data Services**
- **Voice Technology Support**
- **Voice and Unified Communications Product Support**
- **Troubleshooting Cisco IP Telephony** [↗](#)
- **Technical Support & Documentation – 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: Aug 20, 2006

Document ID: 24644

---