Troubleshooting Dial Technology Connectivity – IOS DDR Initiates Call

Document ID: 14954

Contents

Introduction

This document provides methods of troubleshooting different types of dial connections and is not intended to be read from start to finish. The structure is designed to allow the reader to skip forward to the sections of interest, each of which are variations on the overall troubleshooting theme for a specific case.

Prerequisites

Requirements

This document covers three main scenarios; before you begin to troubleshoot, determine what type of call is being attempted and go to that section:

- Callin
- Cisco IOS dial–on–demand routing (DDR)
- Non–DDR callout

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 you are working in a live network, ensure that you understand the potential impact of any command before using it.

History

Dialup is simply the application of the public switched telephone network (PSTN) that carries data on behalf of the end user. It involves a customer premises equipment (CPE) device sending the telephone switch a phone number to which to direct a connection. The AS3600, AS5200, AS5300, and AS5800 are all examples of routers that have the ability to run a Primary Rate Interface (PRI) along with banks of digital modems. The AS2511, on the other hand, is an example of a router that communicates with external modems.

The carrier market has grown significantly, and the market now demands higher modem densities. The answer to this need is a higher degree of interoperation with the telephone company equipment and the development of the digital modem. This is a modem that is capable of direct digital access to the PSTN. As a result, faster CPE modems have now been developed that take advantage of the clarity of signal that the digital modems enjoy. The fact that the digital modems connecting into the PSTN through a PRI or Basic Rate Interface (BRI) can transmit data at over 53k using the V.90 communication standard, attests to the success of the idea.

The first access servers were the AS2509 and AS2511. The AS2509 could support 8 incoming connections using external modems, and the AS2511 could support 16. The AS5200 was introduced with 2 PRIs and could support 48 users using digital modems, and it represented a major leap forward in technology. Modem densities have increased steadily with the AS5300 supporting 4 and then 8 PRIs. Finally, the AS5800 was introduced to fill the needs of carrier class installations needing to handle dozens of incoming T1s and hundreds of user connections.

A couple of outdated technologies bear mentioning in a historical discussion of dialer technology. 56Kflex is an older (pre−V.90) 56k modem standard that was proposed by Rockwell. Cisco supports version 1.1 of the 56Kflex standard on its internal modems, but recommends migrating the CPE modems to V.90 as soon as possible. Another outdated technology is the AS5100. The AS5100 was a joint venture between Cisco and a modem manufacturer. The AS5100 was created as a way to increase modem density through the use of quad modem cards. It involved a group of AS2511s built as cards that inserted into a backplane shared by quad modem cards, and a dual T1 card.

Conventions

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

Cisco IOS DDR Initiates the Call

While the troubleshooting approach for incoming calls starts at the bottom, troubleshooting an outbound connection starts at the top.

The general flow of reasoning resembles the following:

1. Does the Dial on Demand Routing (DDR) initiate a call? (A yes answer advances to the next question.)
2. If this is an async modem, do the chat scripts issue the expected commands?
3. Does the call make it out to the PSTN?
4. Does the remote end answer the call?
5. Does the call complete?
6. Is data passing over the link?
7. Is the session established? (PPP or Terminal)

To see whether the dialer is trying to make a call to its remote destination, use the command **debug dialer events**. More detailed information can be gained from **debug dialer packet**, but the **debug dialer packet** command is resource−intensive and should not be used on a busy system that has multiple dialer interfaces

operating.

The following line of **debug dialer events** output for an IP packet lists the name of the DDR interface and the source and destination addresses of the packet:

```
Dialing cause: Async1: ip (s=172.16.1.111 d=172.16.2.22)
```

If traffic does not initiate a dial attempt, the most common reason is improper configuration (either of the interesting traffic definitions, the state of the dialer interface, or the routing).

Possible Causes Suggested Actions Missing or incorrect "interesting traffic" definitions

1. Using the command **show running−config**, ensure that the interface is configured with a *dialer−group* and that there is a global level *dialer−list* configured with a matching number.
2. Ensure that the *dialer−list* command is configured to permit either an entire protocol or to permit traffic matching an access list
3. Verify that the *access−list* declares packets going across the link to be interesting. One useful test is to use the privileged exec command **debug ip packet [list number]** using the number of the pertinent access list. Then attempt to ping, or otherwise send traffic, across the link. If the interesting traffic filters have been properly defined, you will see the packets in the debug output. If there is no debug output from this test, the *access−list* is not matching the packets.

Interface state Use the command **show interfaces** *<interface name>* to ensure that the interface is in the state "up/up (spoofing)."

- Interface in "standby" mode

Another (primary) interface on the router has been configured to use the dialer interface as a backup interface. Furthermore, the primary interface is not in a state of "down/down", which is required to bring the dialer interface out of standby mode. Also, a *backup delay* must be configured on the primary interface, or the *backup interface* command will never be enforced. To check that the dialer interface will change from "standby" to "up/up (spoofing)," it is usually necessary to pull the cable from the primary interface. Simply shutting down the primary interface with the configuration command **shutdown** will not put the primary interface into "down/down," but instead will put it into "administratively down" ? not the same thing. In addition, if the primary connection is via Frame Relay, the Frame Relay configuration must be done on a point−to−point Serial sub−interface, and the telephone company must be passing the "Active" bit. This practice is also known as "end−to−end Local Management Interface (LMI)."

- Interface is "administratively down"

The dialer interface has been configured with the command **shutdown**. This is also the default state of any interface when a Cisco router is booted for the very first time. Use the interface configuration command **no shutdown** to remove this impediment. Incorrect routing Issue the exec command **show ip route [*a.b.c.d*]**, where *a.b.c.d* is the address of the dialer interface of the remote router. If **ip unnumbered** is used on the remote router, use the address of the interface listed in the **ip unnumbered** command. The output should show a route to the remote address via the dialer interface. If there is no route, ensure that static or floating static routes have been configured by examining the output of **show running−config**. If there is a route via an interface other than the dialer interface, the implication is that DDR is being used as a backup. Examine the router configuration to make sure that static or floating static routes have been configured. The surest way to test the routing, in this case, is to disable the primary connection and execute the **show ip route [*a.b.c.d*]** command to verify that the proper route has been installed in the routing table.

Note: If you attempt this during live network operations, a dial event may be triggered. This sort of testing is

best accomplished during scheduled maintenance cycles.

Placing the Call

If the routing and the interesting traffic filters are correct, a call should be initiated. This can be seen by using **debug dialer events**:

```
Async1 DDR: Dialing cause ip (s=10.0.0.1, d=10.0.0.2)



Async1 DDR: Attempting to dial 5551212
```

If the dialing cause is seen but no attempt is made to dial, the usual reason is a misconfigured dialer map or dialer profile.

Possible problem Suggested actions Misconfigured dialer map Use the command **show running−config** to ensure that the dialing interface is configured with at least one *dialer map* statement that points to the protocol address and called number of the remote site. Misconfigured dialer profile Use the command **show running−config** to ensure that the Dialer interface is configured with a **dialer pool X** command and that a dialer interface on the router is configured with a matching *dialer pool−member X* . If dialer profiles are not properly configured, you may see a debug message like:

```
Dialer1: Can't place call, no dialer pool set
```

Make sure that a *dialer string* is configured.

Next, identify the type of media that is being used:

- External async modem DDR
- Channel Associated Signaling (CAS) T1/E1 async modem DDR
- PRI async modem DDR
- BRI async modem DDR
- PRI ISDN DDR
- BRI ISDN DDR

External Async Modem DDR

1. To identify an external async modem DDR, use the following commands, then try to make a call:

   ```
   router# debug modem
   router# debug chat line <n>
   ```

2. For modem calls, a chat script must execute in order for the call to proceed. For dialer map−based DDR, the chat script is invoked by the **modem−script** parameter in a dialer map command. If the DDR is dialer profile−based, this is accomplished by the command **script dialer**, configured on the TTY line. Both methods rely on a chat script existing in the router's global configuration, for example:

   ```
   chat-script callout AT OK atdt\T TIMEOUT 60 CONNECT \c
   ```

   In either event, the command to view the chat script activity is **debug chat**. If the dial string (that is, phone number) used in the **dialer map** or **dialer string** command were 5551212, the debug output would look like the following:

```
CHAT1: Attempting async line dialer script


CHAT1: Dialing using Modem script: callout & System script: none
CHAT1: process started
CHAT1: Asserting DTR
CHAT1: Chat script callout started
CHAT1: Sending string: AT
CHAT1: Expecting string: OK
CHAT1: Completed match for expect: OK
CHAT1: Sending string: atdt5551212
CHAT1: Expecting string: CONNECT
CHAT1: Completed match for expect: CONNECT
CHAT1: Chat script callout finished, status = Success
```

3. Ensure that the chat script attempts the expected call (i.e. the right number) based on the "Sending string." If the chat script does not attempt to make the expected call, verify the configuration of the chat script. Use the **start–chat** command at the exec prompt to initiate the chat script manually.
4. Seeing "Timeout expecting: CONNECT" can describe several different possibilities:

   ♦ **Possibility 1**: The local modem is not actually placing the call. Verify that the modem can place a call by performing a reverse telnet to the modem and manually initiating a dial. If the remote end doesn't seem to be answering, verify that the call is being placed by the originating modem by calling a local number manually with the command **ATDT** *<number>* and listening for the ring.
   ♦ **Possibility 2**: The remote modem is not answering. Test this by dialing the remote modem with an ordinary POTS telephone. Try this:

      a. Ensure that the called phone number is correct. Use a handset to call the receiving number. Be sure to use the same cable to the wall that the modem was using. If a manual call is able to reach the receiving async modem, the originating modem may not be working correctly. Verify the modem and replace as needed.
      b. If a manual call is not able to reach the answering async modem, change the phone cables on the receiving modem and try a regular phone on the receiving modem's line. If the call can be received by the regular phone, there is likely a problem with the receiving modem.
      c. If the manual call is still not able to reach the regular phone on the line in question, try another (known good) line in the receiving facility. If that connects OK, have the telco check the phone line going to the receiving modem.
      d. If this is a long distance call, have the originating side try another (known good) long distance number. If that works, the receiving facility or line may not be provisioned to receive long distance calls. If the originating line cannot reach any other long distance numbers, it may not have long distance enabled. Try 1010 codes for different long distance companies.
      e. Finally, try another (known good) local number from the originating line. If the connection still fails, have the telco check the originating line.
   ♦ **Possibility 3**: The number being dialed is incorrect. Verify the number by dialing it manually. Correct the configuration, if necessary.
   ♦ **Possibility 4**: The modem trainup is taking too long or the TIMEOUT value is too low. Try increasing the TIMEOUT value in the **chat–script** command. If the TIMEOUT is already 60 seconds or more, there may be a cable problem between the modem and the DTE to which it is attached. Trainup failures can also indicate a circuit problem or modem incompatibility.

   To get to the bottom of an individual modem problem, go to the AT prompt at the originating modem with reverse telnet. If possible, get to the AT prompt of the receiving modem as well. Most modems will indicate a ring to the terminal session attached to their DTE connection. Use ATM1 to have the modem send sounds to its speaker so that people at each end can hear what is happening on the line.

Does the music have noise in it? If so, clean up the circuit. If the async modems do not train up, call the number and listen for static. There may be other factors interfering with train up. Reverse telnet to the async modem and debug.

5. If everything is working fine and you still cannot connect on your CAS async modem DDR, try PPP debugging. Use the commands:

```
router# debug ppp negotiation
      router# debug ppp authentication
```

If the chat script completes successfully, the modems are connected. Consult the "Troubleshooting PPP" section in Chapter 17 of the Internetwork Troubleshooting Guide for the next step in troubleshooting the conection.

CAS T1/E1 Async Modem DDR

1. To identify a CAS T1/E1 async modem DDR, use the following commands, then try to make a call:

Warning: *Running debugs on a busy system could crash the router by overloading the CPU or over−running the console buffer!*

```
router# debug modem
router# debug chat or debug chat line n

router# debug modem csm
router# debug cas
```

Note: The **debug cas** command is available on Cisco AS5200 and AS5300 platforms running Cisco IOS?? Software Release 12.0(7)T and later. In prior versions of IOS, the command **service internal** would have to be entered into the main level of the router's configuration and **modem−mgmt csm debug−rbs** would need to be entered at the exec prompt. Debugging on the Cisco AS5800 requires connecting to the trunk card. (Use **modem−mgmt csm no−debug−rbs** to turn off the debug.)

2. For modem calls, a chat script must execute in order for the call to proceed. For dialer map−based DDR, the chat script is invoked by the **modem−script** parameter in a dialer map command. If the DDR is dialer profile−based, this is accomplished by the command **script dialer**, configured on the TTY line. Both uses rely on a chat script existing in the router's global configuration, such as:

```
chat-script callout AT OK atdt\T TIMEOUT 60 CONNECT \c
```

In either event, the command to view the chat script activity is **debug chat**. If the dial string (that is, phone number) used in the **dialer map** or **dialer string** command were 5551212, the debug output would look like the following:

```
CHAT1: Attempting async line dialer script


CHAT1: Dialing using Modem script: callout & System script: none
CHAT1: process started
CHAT1: Asserting DTR
CHAT1: Chat script callout started
CHAT1: Sending string: AT
CHAT1: Expecting string: OK
CHAT1: Completed match for expect: OK
CHAT1: Sending string: atdt5551212
CHAT1: Expecting string: CONNECT
CHAT1: Completed match for expect: CONNECT
CHAT1: Chat script callout finished, status = Success
```

3. Ensure that the chat script attempts the expected call (i.e. the right number) based on the "Sending string". If the chat script does not attempt to make the expected call, verify the configuration of the chat script. Use the **start–chat** command at the exec prompt to initiate the chat script manually.
4. Seeing "Timeout expecting: CONNECT" can describe several different possibilities:

   ♦ **Possibility 1**: The local modem is not actually placing the call. Verify that the modem can place a call by performing a reverse telnet to the modem and manually initiating a dial. If the remote end doesn't seem to be answering, verify the call is being placed by the modem by calling a local number manually with the command **ATDT**<*number*> and listening for the ring.

   For outbound calling via CAS T1 or E1 and integrated digital modems, much of the troubleshooting is similar to other DDR troubleshooting. The same holds true, as well, for outbound integrated modem calls over a PRI line. The unique features involved in making a call in this manner require special debugging in the event of a call failure.

   As for other DDR situations, you must ensure that a call attempt is demanded. Use debug dialer events for this purpose. Refer to *IOS DDR* , earlier in this article.

   Before a call can be placed, a modem must be allocated for the call. To view this process and the subsequent call, use the following debug commands:

   ```
   router# debug modem
   router# debug modem csm
   router# debug cas
   ```

   Note: The **debug cas** command first appeared in IOS version 12.0(7)T for the AS5200 and AS5300. Earlier versions of IOS use a system–level configuration command **service internal** along with the exec command **modem–mgmt debug rbs**:

   Turning on the Debugs:

   ```
   router#conf t
   Enter configuration commands, one per line.  End with CNTL/Z.
   router(config)#service internal
   router(config)#^Z
   router#modem-mgmt csm ?
     debug-rbs      enable rbs debugging
     no-debug-rbs   disable rbs debugging
   router#modem-mgmt csm debug-rbs
   router#
   neat msg at slot 0: debug-rbs is on
   neat msg at slot 0: special debug-rbs is on
   ```

   Turning off the Debugs:

   ```
   router#modem-mgmt csm no-debug-rbs
   neat msg at slot 0: debug-rbs is off
   ```

   Debugging this information on an AS5800 requires connecting to the trunk card. The following is an example of a normal outbound call over a CAS T1 that is provisioned and configured for FXS–Ground–Start:

   ```
   Mica Modem(1/0): Rcvd Dial String(5551111)
   [Modem receives digits from chat script]

   CSM_PROC_IDLE: CSM_EVENT_MODEM_OFFHOOK at slot 1, port 0
   ```

```
CSM_RX_CAS_EVENT_FROM_NEAT:(A003):
EVENT_CHANNEL_LOCK at slot 1 and port 0

CSM_PROC_OC4_DIALING:
CSM_EVENT_DSX0_BCHAN_ASSIGNED at slot 1, port 0

Mica Modem(1/0): Configure(0x1)

Mica Modem(1/0): Configure(0x2)

Mica Modem(1/0): Configure(0x5)

Mica Modem(1/0): Call Setup

neat msg at slot 0: (0/2): Tx RING_GROUND

Mica Modem(1/0): State Transition to Call Setup

neat msg at slot 0: (0/2): Rx TIP_GROUND_NORING
[Telco switch goes OFFHOOK]

CSM_RX_CAS_EVENT_FROM_NEAT:(A003):
EVENT_START_TX_TONE at slot 1 and port 0

CSM_PROC_OC5_WAIT_FOR_CARRIER:
CSM_EVENT_DSX0_START_TX_TONE at slot 1, port 0

neat msg at slot 0: (0/2):
Tx LOOP_CLOSURE [Now the router goes OFFHOOK]

Mica Modem(1/0): Rcvd Tone detected(2)

Mica Modem(1/0): Generate digits:called_party_num=5551111 len=8

Mica Modem(1/0): Rcvd Digits Generated

CSM_PROC_OC5_WAIT_FOR_CARRIER:
CSM_EVENT_ADDR_INFO_COLLECTED at slot 1, port 0

CSM_RX_CAS_EVENT_FROM_NEAT:(A003):
EVENT_CHANNEL_CONNECTED at slot 1 and port 0

CSM_PROC_OC5_WAIT_FOR_CARRIER:
CSM_EVENT_DSX0_CONNECTED at slot 1, port 0

Mica Modem(1/0): Link Initiate

Mica Modem(1/0): State Transition to Connect

Mica Modem(1/0): State Transition to Link

Mica Modem(1/0): State Transition to Trainup

Mica Modem(1/0): State Transition to EC Negotiating

Mica Modem(1/0): State Transition to Steady State

Mica Modem(1/0): State Transition to Steady State Speedshifting

Mica Modem(1/0): State Transition to Steady State
```

Debugs for T1s and E1s with other signaling types are similar.

Getting to this point in the debugging indicates that the calling and answering modems have trained and connected and that higher−layer protocols can begin to negotiate. If a modem is

properly allocated for the outbound call but the connection fails to get this far, the T1 must be examined. Use the **show controller t1/e1** command to verify that T1/E1 is working. See Troubleshooting Serial Lines for an explanation of **show controller** output. If the T1/E1 is not working properly, T1/E1 troubleshooting is necessary.

♦ **Possibility 2**: The remote modem is not answering. Test this by dialing the remote modem with an ordinary telephone. Try this:

   a. Ensure that the called phone number is correct. Use a handset to call the receiving number.
   b. Ensure that a manual call is able to reach the answering async modem. If a manual call is able to reach the answering async modem, the CAS line may not be provisioned to allow outbound voice calls.
   c. If a manual call is not able to reach the answering async modem, change the phone cables on the receiving modem and try a regular phone on the receiving modem's line. If the call can be received by the regular phone, there is likely a problem with the receiving modem.
   d. If the manual call is still not able to reach the regular phone on the line in question, try another (known good) line in the receiving facility. If that connects, have the telco check the phone line going to the receiving modem.
   e. If this is a long–distance call, have the originating side try another (known good) long–distance number. If that works the receiving facility or line may not be provisioned to receive long–distance calls. If the originating (CAS) line cannot reach any other long distance numbers, it may not have long distance enabled. Try 10–10 codes for different long distance companies.
   f. Finally, try another (known good) local number from the originating CAS line. If the connection still fails, have telco check the CAS.

♦ **Possibility 3**: The number being dialed is incorrect. Verify the number by dialing it manually. Correct the configuration, if necessary.

♦ **Possibility 4**: The modem trainup is taking too long, or the TIMEOUT value is too low. Try increasing the TIMEOUT value in the **chat–script** command. If the TIMEOUT is already 60 seconds or more, there may be a cable problem between the modem and the DTE to which it is attached. Trainup failures can also indicate a circuit problem or modem incompatibility.

To get to the bottom of an individual modem problem, go to the AT prompt at the originating modem with reverse telnet. If possible, use reverse telnet to get to the AT prompt of the receiving modem as well. Use ATM1 to have the modem send sounds to its speaker so that people at each end can hear what is happening on the line.

Does the music have noise in it? If so, clean up the circuit. If the async modems do not train up, call the number and listen for static. There may be other factors interfering with train up. Reverse telnet to the async modem and debug.

5. If everything is working fine and you still cannot connect on your CAS async modem DDR, try PPP debugging. If the chat script completes successfully and the PPP debugs indicate a failure, consult the "Troubleshooting PPP" section in Chapter 17 of the Internetwork Troubleshooting Guide.

PRI Async Modem DDR

1. To identify a PRI async modem DDR, use the following commands, then try to make a call:

Warning: *Running debugs on a busy system could crash the router by overloading the CPU or over–running the console buffer!*

```
router# debug modem
```

```
router# debug chat
router# debug modem csm
router# debug isdn q931
router# debug isdn
router# debug ppp negotiate
router# debug ppp authenticate
```

2. For modem calls, a chat script must execute in order for the call to proceed. For dialer map–based DDR, the chat script is invoked by the **modem–script** parameter in a dialer map command. If the DDR is dialer profile–based, this is accomplished by the command **script dialer**, configured on the TTY line. Both methods rely on a chat script existing in the router's global configuration, such as:

```
chat-script callout AT OK atdt\T TIMEOUT 60 CONNECT \c
```

In either event, the command to view the chat script activity is **debug chat**. If the dial string (phone number) used in the **dialer map** or **dialer string** command were 5551212, the debug output would look like the following:

```
CHAT1: Attempting async line dialer script


CHAT1: Dialing using Modem script: callout & System script: none
CHAT1: process started
CHAT1: Asserting DTR
CHAT1: Chat script callout started
CHAT1: Sending string: AT
CHAT1: Expecting string: OK
CHAT1: Completed match for expect: OK
CHAT1: Sending string: atdt5551212
CHAT1: Expecting string: CONNECT
CHAT1: Completed match for expect: CONNECT
CHAT1: Chat script callout finished, status = Success
```

3. Ensure that the chat script attempts the expected call (the right number) based on the "Sending string." If the chat script does not attempt to make the expected call, verify the configuration of the chat script. Use the **start–chat** command at the exec prompt to initiate the chat script manually.

4. Seeing "Timeout expecting: CONNECT" can describe several different possibilities:

   ♦ **Possibility 1**: The local modem is not actually placing the call. Verify that the modem can place a call by performing a reverse telnet to the modem and manually initiating a dial. If the remote end doesn't seem to be answering, verify that the call is being placed by the modem by calling a local number manually with the command **ATDT**<*number*> and listening for the ring. If no call gets out, turn on ISDN debugs. Upon the first suspicion of an ISDN failure on a BRI, always check the output from **show isdn status**. The key things to note are that Layer 1 should be *Active* and Layer 2 should be in a state of *MULTIPLE_FRAME_ESTABLISHED*. See the Internetwork Troubleshooting Guide Chapter 16, "Interpreting Show ISDN Status" for information on reading this output, as well as for corrective measures.

   For outbound ISDN calls, **debug isdn q931** and **debug isdn events** are the best tools to use. Fortunately, debugging outbound calls is very similar to debugging incoming calls. A normal successful call might look like this:

```
*Mar 20 21:07:45.025: ISDN SE0:23: Event:
Call to 5553759 at 64 Kb/s

*Mar 20 21:07:45.033: ISDN SE0:23: TX ->  SETUP pd = 8
callref = 0x2C
*Mar 20 21:07:45.037:           Bearer Capability i = 0x8890
*Mar 20 21:07:45.041:           Channel ID i = 0x83
*Mar 20 21:07:45.041:           Keypad Facility i = 0x35353533373539
*Mar 20 21:07:45.141: ISDN SE0:23: RX <-  CALL_PROC pd = 8
callref = 0xAC
```

```
*Mar 20 21:07:45.145:          Channel ID i = 0x89
*Mar 20 21:07:45.157: ISDN SE0:23: received HOST_PROCEEDING
        Channel ID i = 0x0101
*Mar 20 21:07:45.161:    -------------------
        Channel ID i = 0x89
*Mar 20 21:07:45.313: ISDN SE0:23: RX <-  CONNECT pd = 8
callref = 0xAC
*Mar 20 21:07:45.325: ISDN SE0:23: received HOST_CONNECT
```

Note that the CONNECT message is the key indicator of success. If a CONNECT is not received, you may see a DISCONNECT or a RELEASE_COMP (release complete) message followed by a cause code:

```
*Mar 20 22:11:03.212: ISDN SE0:23:
RX <-RELEASE_COMP pd = 8 callref = 0x8F
*Mar 20 22:11:03.216:          Cause i = 0x8295 – Call rejected
```

The cause value indicates two things:

◊ The second byte of the 4– or 6–byte value indicates the point in the end–to–end call path from which the DISCONNECT or RELEASE_COMP was received. This can help you to localize the problem.
◊ The third and fourth bytes indicate the actual reason for the failure. See Table 9 for the meanings of the different values.

♦ **Possibility 2**: The remote modem is not answering. Test this by dialing the remote modem with an ordinary telephone. Try this:

a. Ensure that the called phone number is correct. Use a handset to call the receiving number.
b. Ensure that a manual call is able to reach the answering async modem. If a manual call is able to reach the answering async modem, the BRI line may not be provisioned to allow outbound voice calls.
c. If a manual call is not able to reach the answering async modem, change the phone cables on the receiving modem and try a regular phone on the receiving modem's line. If the call can be received by the regular phone, there is likely a problem with the receiving modem.
d. If the manual call is still not able to reach the regular phone on the line in question, try another (known good) line in the receiving facility. If that connects, have the telco check the phone line going to the receiving modem.
e. If this is a long–distance call, have the originating side try another (known good) long–distance number. If that works, the receiving facility or line may not be provisioned to receive long–distance calls. If the originating (BRI) line cannot reach any other long–distance numbers, it may not have long distance enabled. Try 1010 codes for different long–distance companies.
f. Finally, try another (known good) local number from the originating BRI line. If the connection still fails, have the telco check the BRI.

♦ **Possibility 3**: The number being dialed is incorrect. Verify the number by dialing it manually. Correct the configuration, if necessary.
♦ **Possibility 4**: The modem trainup is taking too long or the TIMEOUT value is too low. Try increasing the TIMEOUT value in the chat–script command. If the TIMEOUT is already 60 seconds or more, there may be a cable problem between the modem and the DTE it is attached to. Trainup failures can also indicate a circuit problem or modem incompatibility.

To get to the bottom of an individual modem problem, go to the AT prompt at the originating modem with reverse telnet. If possible, use reverse telnet to get to the AT prompt of the receiving modem as well. Use ATM1 to have the modem send sounds to its speaker so that

people at each end can hear what is happening on the line.

Does the music have noise in it? If so, clean up the circuit. If the async modems do not train up, call the number and listen for static. There may be other factors interfering with train up. Reverse telnet to the async modem and debug.

5. If everything is working fine and you still cannot connect on your BRI async modem DDR, try PPP debugging. If the chat script completes successfully and the PPP debugs indicate a failure, consult the "Troubleshooting PPP" section in Chapter 17 of the Internetwork Troubleshooting Guide.

BRI Async Modem DDR

This feature only works on the Cisco 3640 platform using Cisco IOS Software Release 12.0(3)T or later. It requires a later hardware revision of the BRI network module. This will not work with a WAN Interface Card (WIC).

1. Ensure that the country code is correct with the show modem command. Use the following commands, then try to make a call:

   Warning: *Running debugs on a busy system could crash the router by overloading the CPU or over−running the console buffer!*

   ```
   router# debug modem
   router# debug chat
   router# debug modem csm
   router# debug isdn q931
   router# debug bri
   router# debug ppp negotiate
   router# debug ppp authenticate
   ```

2. For modem calls, a chat script must execute in order for the call to proceed. For dialer map−based DDR, the chat script is invoked by the **modem−script** parameter in a dialer map command. If the DDR is dialer profile−based, this is accomplished by the command **script dialer**, configured on the TTY line. Both uses rely on a chat script existing in the router's global configuration, such as:

   ```
   chat-script callout AT OK atdt\T TIMEOUT 60 CONNECT \c
   ```

   In either event, the command to view the chat script activity is **debug chat**. If the dial string (phone number) used in the **dialer map** or **dialer string** command were 5551212, the debug output would look like the following:

   ```
   CHAT1: Attempting async line dialer script

   CHAT1: Dialing using Modem script: callout & System script: none
   CHAT1: process started
   CHAT1: Asserting DTR
   CHAT1: Chat script callout started
   CHAT1: Sending string: AT
   CHAT1: Expecting string: OK
   CHAT1: Completed match for expect: OK
   CHAT1: Sending string: atdt5551212
   CHAT1: Expecting string: CONNECT
   CHAT1: Completed match for expect: CONNECT
   CHAT1: Chat script callout finished, status = Success
   ```

3. Ensure that the chat script attempts the expected call (the right number) based on the "Sending string." If the chat script does not attempt to make the expected call, verify the configuration of the chat script. Use the **start−chat** command at the exec prompt to initiate the chat script manually.

4. Seeing "Timeout expecting: CONNECT" can describe several different possibilities:

- ◆ **Possibility 1**: The local modem is not actually placing the call. Verify that the modem can place a call by performing a reverse telnet to the modem and manually initiating a dial. If the remote end doesn't seem to be answering, verify that the call is being placed by the modem by calling a local number manually with the command **ATDT***<number>* and listening for the ring. If no call gets out, turn on ISDN debugs. Upon the first suspicion of an ISDN failure on a BRI, always check the output from **show isdn status**. The key things to note are that Layer 1 should be *Active* and Layer 2 should be in a state of *MULTIPLE_FRAME_ESTABLISHED* . See the Internetwork Troubleshooting Guide Chapter 16, "Interpreting Show ISDN Status" for information on reading this output and corrective measures.

  For outbound ISDN calls, **debug isdn q931** and **debug isdn events** are the best tools to use. Fortunately, debugging outbound calls is very similar to debugging incoming calls. A normal successful call might look like this:

  ```
  *Mar 20 21:07:45.025: ISDN BR0: Event:
  Call to 5553759 at 64 Kb/s

  *Mar 20 21:07:45.033: ISDN BR0: TX ->  SETUP pd = 8
  callref = 0x2C
  *Mar 20 21:07:45.037:          Bearer Capability i = 0x8890
  *Mar 20 21:07:45.041:          Channel ID i = 0x83
  *Mar 20 21:07:45.041:          Keypad Facility i = 0x35353533373539
  *Mar 20 21:07:45.141: ISDN BR0: RX <-  CALL_PROC pd = 8
  callref = 0xAC
  *Mar 20 21:07:45.145:          Channel ID i = 0x89
  *Mar 20 21:07:45.157: ISDN BR0: received HOST_PROCEEDING
          Channel ID i = 0x0101
  *Mar 20 21:07:45.161:   -------------------
          Channel ID i = 0x89
  *Mar 20 21:07:45.313: ISDN BR0: RX <-  CONNECT pd = 8
  callref = 0xAC
  *Mar 20 21:07:45.325: ISDN BR0: received HOST_CONNECT
  ```

  Note that the CONNECT message is the key indicator of success. If a CONNECT is not received, you may see a DISCONNECT or a RELEASE_COMP (release complete) message followed by a cause code:

  ```
  *Mar 20 22:11:03.212: ISDN BR0: RX <-  RELEASE_COMP pd = 8
  callref = 0x8F
  *Mar 20 22:11:03.216:          Cause i = 0x8295 - Call rejected
  ```

  The cause value indicates two things.

  - ◊ The second byte of the 4– or 6–byte value indicates the point in the end–to–end call path from which the DISCONNECT or RELEASE_COMP was received. This can help you to localize the problem.
  - ◊ The third and fourth bytes indicate the actual reason for the failure. See Table 9 for the meanings of the different values.
- ◆ **Possibility 2**: The remote modem is not answering. Test this by dialing the remote modem with an ordinary telephone. Try this:

  - a. Ensure that the called phone number is correct. Use a handset to call the receiving number.
  - b. Ensure that a manual call is able to reach the answering async modem. If a manual call is able to reach the answering async modem, the BRI line may not be provisioned to allow outbound voice calls.
  - c. If a manual call is not able to reach the answering async modem, change the phone cables on the receiving modem and try a regular phone on the receiving modem's

line. If the call can be received by the regular phone, there is likely a problem with the receiving modem.

    d. If the manual call is still not able to reach the regular phone on the line in question, try another (known good) line in the receiving facility. If that connects, have the telco check the phone line going to the receiving modem.

    e. If this is a long–distance call, have the originating side try another (known good) long–distance number. If that works, the receiving facility or line may not be provisioned to receive long–distance calls. If the originating (BRI) line cannot reach any other long–distance numbers, it may not have long distance enabled. Try 10–10 codes for different long–distance companies.

    f. Finally, try another (known good) local number from the originating BRI line. If the connection still fails, have the telco check the BRI.

♦ **Possibility 3**: The number being dialed is incorrect. Verify the number by dialing it manually. Correct the configuration, if necessary.

♦ **Possibility 4**: The modem trainup is taking too long, or the TIMEOUT value is too low. Try increasing the TIMEOUT value in the chat–script command. If the TIMEOUT is already 60 seconds or more, there may be a cable problem between the modem and the DTE it is attached to. Trainup failures can also indicate a circuit problem or modem incompatibility.

To get to the bottom of an individual modem problem, go to the AT prompt at the originating modem with reverse telnet. If possible, use reverse telnet to get to the AT prompt of the receiving modem as well. Use ATM1 to have the modem send sounds to its speaker so that people at each end can hear what is happening on the line.

Does the music have noise in it? If so, clean up the circuit. If the async modems do not train up, call the number and listen for static. There may be other factors interfering with train up. Reverse telnet to the async modem and debug.

5. If everything is working fine and you still cannot connect on your BRI async modem DDR, try PPP debugging. If the chat script completes successfully and the PPP debugs indicate a failure, consult the "Troubleshooting PPP" section in Chapter 17 of the Internetwork Troubleshooting Guide.

PRI ISDN DDR

1. Upon the first suspicion of an ISDN failure on a PRI, always check the output from **show isdn status**. The key things to note are that Layer 1 should be Active and Layer 2 should be in a state of *MULTIPLE_FRAME_ESTABLISHED*. See the Internetwork Troubleshooting Guide Chapter 16, "Interpreting Show ISDN Status" for information on reading this output and corrective measures.

For outbound ISDN calls, **debug isdn q931** and **debug isdn events** are the best tools to use. Fortunately, debugging outbound calls is very similar to debugging incoming calls. A normal successful call might look like this:

```
*Mar 20 21:07:45.025: ISDN SE0:23: Event:
Call to 5553759 at 64 Kb/s

*Mar 20 21:07:45.033: ISDN SE0:23: TX ->  SETUP pd = 8
callref = 0x2C
*Mar 20 21:07:45.037:          Bearer Capability i = 0x8890
*Mar 20 21:07:45.041:          Channel ID i = 0x83
*Mar 20 21:07:45.041:          Keypad Facility i = 0x35353533373539
*Mar 20 21:07:45.141: ISDN SE0:23: RX <-  CALL_PROC pd = 8
callref = 0xAC
*Mar 20 21:07:45.145:          Channel ID i = 0x89
*Mar 20 21:07:45.157: ISDN SE0:23: received HOST_PROCEEDING
       Channel ID i = 0x0101
*Mar 20 21:07:45.161:    -------------------
```

```
                    Channel ID i = 0x89
      *Mar 20 21:07:45.313: ISDN SE0:23: RX <-  CONNECT pd = 8
      callref = 0xAC
      *Mar 20 21:07:45.325: ISDN SE0:23: received HOST_CONNECT
```

Note that the CONNECT message is the key indicator of success. If a CONNECT is not received, you may see a DISCONNECT or a RELEASE_COMP (release complete) message followed by a cause code:

```
      *Mar 20 22:11:03.212: ISDN SE0:23: RX <-  RELEASE_COMP pd = 8
      callref = 0x8F
      *Mar 20 22:11:03.216:           Cause i = 0x8295 - Call rejected
```

The cause value indicates two things.

- ♦ The second byte of the 4– or 6–byte value indicates the point in the end–to–end call path from which the DISCONNECT or RELEASE_COMP was received. This can help you to localize the problem.
- ♦ The third and fourth bytes indicate the actual reason for the failure. See Table 9 for the meanings of the different values.

Note: The following printout indicates a higher–protocol failure:

```
      Cause i = 0x8090 - Normal call clearing
```

PPP authentication failure is a typical reason. Turn on **debug ppp negotiation** and **debug ppp authentication** before assuming that the connection failure is necessarily an ISDN problem.

2. If the ISDN CONNECT message is seen and the PPP debugs indicate a failure, consult the "Troubleshooting PPP" section in Chapter 17 of the Internetwork Troubleshooting Guide.

BRI ISDN DDR

1. Upon the first suspicion of an ISDN failure on a BRI, always check the output from show isdn status. The key things to note are that Layer 1 should be Active and Layer 2 should be in a state of MULTIPLE_FRAME_ESTABLISHED. See the Internetwork Troubleshooting Guide Chapter 16, "Interpreting Show ISDN Status" for information on reading this output and corrective measures.

For outbound ISDN calls, **debug isdn q931** and **debug isdn events** are the best tools to use. Fortunately, debugging outbound calls is very similar to debugging incoming calls. A normal successful call might look like this:

```
      *Mar 20 21:07:45.025: ISDN BR0: Event: Call to 5553759 at 64 Kb/s

      *Mar 20 21:07:45.033: ISDN BR0: TX -> SETUP pd = 8  callref = 0x2C
      *Mar 20 21:07:45.037:         Bearer Capability i = 0x8890
      *Mar 20 21:07:45.041:         Channel ID i = 0x83
      *Mar 20 21:07:45.041:         Keypad Facility i = 0x35353533373539
      *Mar 20 21:07:45.141: ISDN BR0: RX <-  CALL_PROC pd = 8  callref = 0xAC
      *Mar 20 21:07:45.145:         Channel ID i = 0x89
      *Mar 20 21:07:45.157: ISDN BR0: received HOST_PROCEEDING
              Channel ID i = 0x0101
      *Mar 20 21:07:45.161:   ------------------
              Channel ID i = 0x89
      *Mar 20 21:07:45.313: ISDN BR0: RX <-  CONNECT pd = 8  callref = 0xAC
      *Mar 20 21:07:45.325: ISDN BR0: received HOST_CONNECT
```

Note that the CONNECT message is the key indicator of success. If a CONNECT is not received, you may see a DISCONNECT or a RELEASE_COMP (release complete) message followed by a cause

code:

```
*Mar 20 22:11:03.212: ISDN BR0: RX <-  RELEASE_COMP pd = 8
callref = 0x8F
*Mar 20 22:11:03.216:          Cause i = 0x8295 - Call rejected
```

The cause value indicates two things.

♦ The second byte of the 4– or 6–byte value indicates the point in the end–to–end call path from which the DISCONNECT or RELEASE_COMP was received. This can help you to localize the problem.
♦ The third and fourth bytes indicate the actual reason for the failure. See Table 9 for the meanings of the different values.

Note: The following printout indicates a higher–protocol failure:

```
Cause i = 0x8090 - Normal call clearing
```

PPP authentication failure is a typical reason. Turn on **debug ppp negotiation** and **debug ppp authentication** before assuming that the connection failure is necessarily an ISDN problem.
2. If the ISDN CONNECT message is seen and the PPP debugs indicate a failure, consult the "Troubleshooting PPP" section in Chapter 17 of the Internetwork Troubleshooting Guide.

Related Information

- **Cisco IOS Dial Services Quick Configuration Guide**
- **Cisco IOS Dial Services Configuration Guide: Network Services**
- **Cisco IOS Dial Services Configuration Guide: Terminal Services**
- **Cisco IOS Dial Services Command Reference**
- **Dial Case Study Overview**
- **Access Technology Pages**
- **Technical Support & Documentation – Cisco Systems**