Callgen User Manual

Release 4.2T

Document Revision 1.1
Last Modified: 1/5/05

CONTENTS

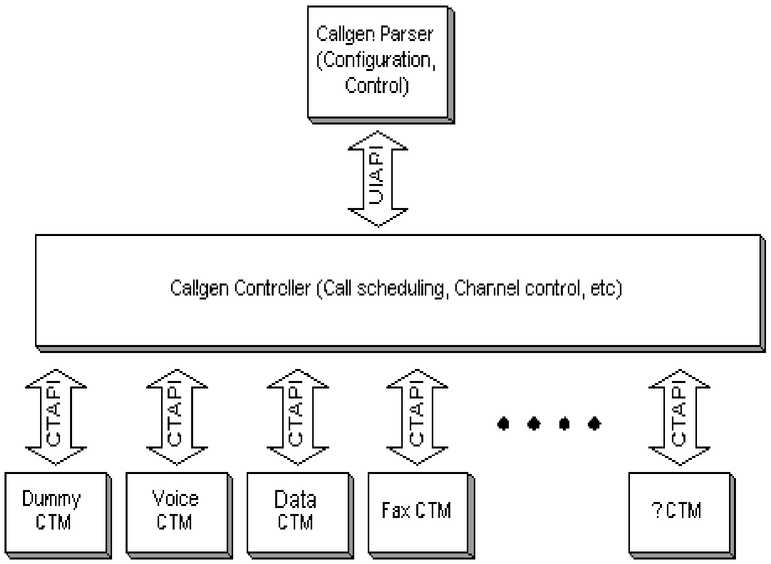
Introduction

Callgen is an IOS-based bulk-call tool that generates voice calls over T1/E1 PRI/CAS, analog, VoIP, and VoFR interfaces. It currently runs on 1760, 3640, 3660, 3725, 3745, 5400, 5850, and 7200 router/access servers. Callgen places fax calls on both analog (FXO and FXS ports) and digital interfaces (T1/E1) using libretto code.

Callgen is designed as a generic bulk-call generator so that it can be extended to incorporate other call types, some of which are not strictly in the telephony/dial area, for example, IP-based video streams.

Overview

The following diagram shows a high-level view of Callgen's major functional blocks: the parser, controller, and one or more call type modules (CTMs).


Callgen Parser (Configuration, Control)
UIAPI
Callgen Controller (Call scheduling, Channel control, etc)
CTAPI
CTAPI
CTAPI
CTAPI
CTAPI
Dummy CTM
Voice CTM
Data CTM
Fax CTM
? CTM

- Parser—Callgen uses the IOS command-line parser for all user interaction. However, the User Interface API (UIAPI) has been designed for developing other user interfaces in the future, such as a web front end.
- Controller—This runs as a separate IOS process and is responsible for receiving configuration and control commands from the user, scheduling the execution of all calls, and collecting various call-specific and global statistics.

- Call Type Module—The CTMs provide a modular way to capture all the call type-specific behavior. They are implemented as a separate subsystem within IOS. The Call Type API (CTAPI) is defined to describe the way in which the Callgen controller and CTMs communicate. This release of Callgen includes voice, data, and fax CTMs. A dummy CTM is also included for testing purposes.

Note The Callgen **show version** command gives information about the versions of the controller and all registered call type modules.

Callgen uses the concept of channels to generate calls. Each channel represents a single caller; only one call can occur at a time on a channel. Using configuration parameters, you define the type of call placed on the channel and whether it is a originating or terminating channel.

Each channel has a specific call type, such as voice, data, or fax. Each call type has it own configuration parameters. For information on how to configure channels, see Chapter 4, "Configuring Channels."

The next chapter contains a few examples for getting started using Callgen. For more information on how to use Callgen commands, hardware and security requirements, and logging capabilities, see Chapter 3, "Using Callgen."

Quick Start

This chapter provides a quick overview of how to use Callgen through a few simple examples.

Generating a Series of Originate Calls

This example generates a sequence of calls. It uses a special Callgen call type called “dummy,” created to aid internal testing during Callgen development. Dummy places calls that do nothing but pretend to connect, wait a specified amount of time, and then hang up. This example provides an overview of running Callgen and introduces certain basic concepts. It also offers a quick introduction to the Callgen user interface.

Step 1 Load the Callgen IOS image into the router. After booting the IOS image, you should see a message similar the following:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!                                                     !!!!
!!!!                 CISCO CONFIDENTIAL.                 !!!!
!!!!  DO NOT COPY OR DISTRIBUTE WITHOUT AUTHORIZATION.   !!!!
!!!!                                                     !!!!
!!!!  WARNING: You have loaded a Callgen IOS image.      !!!!
!!!!         Only secure Callgen images with license     !!!!
!!!!         key may be run outside of Cisco facilities. !!!!
!!!!                                                     !!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

If you missed the boot banner message, to make sure you are running a Callgen-enabled image issue a simple Callgen command from the IOS exec prompt, for example, **show version**:

```
clash-5300# callgen show version

Callgen Version:        3.0.0
Controller Version:     2.0.0
VoiceCTM Version:       2.2.0
DummyCTM Version:       1.0.0
FaxCTM Version:         2.1.0
DataCTM Version:        1.1.0
clash-5300#
```

Step 2 Enter **callgen** at the IOS exec prompt to go to Callgen command mode:

```
wizards-3640# callgen
wizards-3640(callgen)#
```

Step 3 Add a “dummy, originate” channel to place the call.

The channel is the basic building block within Callgen and represents a single endpoint capable of originating or terminating calls. Each channel has a type (for example, voice, fax, modem) and a mode (originate or terminate). If neither are specified, the channel defaults to **voice, originate**. To add a dummy, originate channel:

```
wizards-3640(callgen)# channel 1 type dummy
wizards-3640(cgch_1_du_o)#
```

Each channel is identified by a number. In this case, we have arbitrarily chosen “1” for the channel number. Any number from 1 to 10000 is valid. Channels do not have to be added or defined in any particular order.

After you enter the channel command, the prompt changes to channel mode. Any subsequent configuration commands apply to the specified channel. The “du” in the prompt indicates that this channel is of type “dummy.” The “o” indicates that this is an “originate” channel (the default).

Step 4 Configure the channel to place calls at a rate of 10 per minute, with each call lasting 3 seconds:

```
wizards-3640(cgch_1_du_o)# rate 10 per minute
wizards-3640(cgch_1_du_o)# duration 3 seconds
```

Step 5 Verify the Callgen configuration using the **show config** command:

```
wizards-3640(cgch_1_du_o)# show config

channel 1 type dummy mode originate
  rate 10 per minute
  duration 3 seconds

wizards-3640(callgen)#
```

Step 6 Begin placing calls for a total of 10 minutes.

```
wizards-3640(callgen)# start test-duration 10 minutes
```

The goal here is to place 10 calls per minute, for a total of about 100 calls. Each call will connect, last 3 seconds, and then hang up. Approximately 3 seconds later (the rate of 10 per minute indicates a call placed once every 6 seconds: 3 seconds of call duration, plus 3 seconds idle), the next call will be placed.

Note If this were a real call, such as voice, you would need to supply more information before actually placing calls, such as the outgoing interface and the number you are calling.

Step 7 Periodically check the status using the Callgen **show** command. The following output confirms the 3-second hold time and the 3-second average idle time for each call. A call’s idle time is when the channel is not actively in a call.

```
wizards-3640(callgen)# show

Aggregate Call Statistics
  Elapsed time of session: 36176ms and counting
  Originate Statistics
    max# of concurrent calls: 1
    active channels: 1 of 1
    setup attempts: 7
    accepts: 6
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    setup rate: 0 calls per millisecond
    accept rate: 0 calls per millisecond
    setup time: min: 0ms, max: 0ms, avg: 0ms
    hold time: min: 3000ms, max: 3000ms, avg: 3000ms
    disconnect time: min: 0ms, max: 0ms, avg: 0ms
    idle time: min: 3000ms, max: 3000ms, avg: 3000ms
  Terminate Statistics
    active channels: 0 of 0
    setup attempts: 0
    accepts: 0
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    setup rate: 0 calls per millisecond
    accept rate: 0 calls per millisecond
    hold time: min: 0ms, max: 0ms, avg: 0ms
```

The output also displays how many originate and terminate channels have been defined and are active, and indicates the total number of call attempts and accepts. A Callgen router can have both originate and terminate channels active simultaneously, and call statistics are aggregated separately for originate and terminate channels. Dummy calls are fake and are automatically answered. If these were real calls, you would have to have a device to accept and terminate the call. This could be Callgen, either running on the same or a different router, a telephony device, or a loopback dial peer.

After 10 minutes, we can confirm that the expected 100 calls were generated and accepted. We also see that the channel is no longer active and that the session lasted for 10 minutes (600,000 msec). All the calls were answered: setup attempts (100) equals accepts (100).

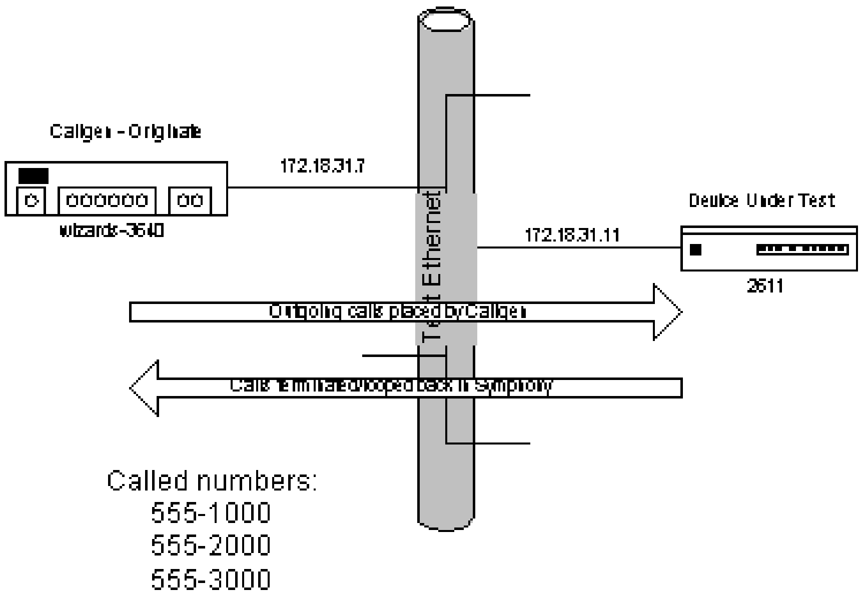
```
wizards-3640(callgen)# show

Aggregate Call Statistics
  Elapsed time of session: 600000ms
  Originate Statistics
    max# of concurrent calls: 1
    active channels: 0 of 1
    setup attempts: 100
    accepts: 100
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    setup rate: 0 calls per millisecond
    accept rate: 0 calls per millisecond
    setup time: min: 0ms, max: 0ms, avg: 0ms
    hold time: min: 3000ms, max: 3000ms, avg: 3000ms
    disconnect time: min: 0ms, max: 0ms, avg: 0ms
    idle time: min: 3000ms, max: 3000ms, avg: 3000ms
  Terminate Statistics
    active channels: 0 of 0
    setup attempts: 0
    accepts: 0
```

```
confirms: 0
setup-fails: 0
aborts: 0
abnormal disconnects: 0
confirmed errors: 0
other errors: 0
setup rate: 0 calls per millisecond
accept rate: 0 calls per millisecond
hold time: min: 0ms, max: 0ms, avg: 0ms
```

Placing Multiple, Simultaneous VoIP Calls

The following example places voice over IP (VoIP) calls from a 3640 router to a 2600 router with a loopback dial peer. Using the dial-peer loopback mechanism in Symphony to “wrap” the voice path back to the originator is one way to terminate calls. This example places three simultaneous VoIP calls to three different numbers, each having different characteristics.



The characteristics of the calls are:

- Channel 1: Calls to 555-1000 at a rate of 60 call attempts per hour, each lasting 1 minute
- Channel 2: Calls to 555-2000 at a rate of 3 per minute, each with a random duration between 5 and 15 seconds
- Channel 3: Calls to 555-3000, started 5 minutes into the run (after Channel 1 and 2 have started), at a rate of 3 per hour, each lasting 5 minutes

Step 1 Create a dial peer.

Before we can originate or terminate VoIP calls or any type of VoXX call, we must first create a dial peer as part of the router’s configuration. The dial peer contains information on how calls are placed over the IP or packet network, such as the address of the call target and desired CODEC. We then specify this dial peer in Callgen’s interface command. This is the dial peer we are using for our example:

```
dial-peer voice 100 voip
  session target ipv4:172.18.31.11
```

Step 2 Configure Callgen on the 3640.

Each command is entered at the Callgen mode prompt.

```
wizards-3640(callgen)# show config

channel 1
  rate 50 per hour
  duration 1 minutes
  calling-number 5550001
  called-number 5551000
  interface voip:100       # Note, ‘100’ corresponds to dial-peer with tag,
‘100.’

channel 2
  rate 3 per minute
  duration random 5 15 seconds
  calling-number 5550001
  called-number 5552000
  interface voip:100

channel 3
  rate 3 per hour
  duration 5 minutes
  start-time-delay 5 minutes
  calling-number 5550001
  called-number 5553000
  interface voip:100
```

Step 3 Configure the remote 2600 with loopback peers.

The dial peers for each of the called numbers are configured as loopback, which indicates that when the Callgen-generated call arrives, it will be answered and the voice path will be looped back to the originator.

```
hostname tte2600
!
!
dial-peer voice 1000 voip
destination-pattern 5551000
session target loopback:rtp
!
dial-peer voice 2000 voip
destination-pattern 5552000
session target loopback:rtp
!
dial-peer voice 3000 voip
destination-pattern 5553000
session target loopback:rtp
!
!
```

Step 4 Start call generation.

We will start just channel 2 for a specified amount of time to make sure that VoIP calls are being placed successfully:

```
wizards-3640(callgen)# start 2 test-duration 10 minutes
wizards-3640(callgen)#
```

Step 5 Check whether calls are being placed with a random hold time between 5 and 15 seconds, using **show channel 2**.

```
Channel 2 Call Statistics
  elapsed channel run time: 00w0d00:02:35.932 and counting
  channel state: HOLD
  setup attempts: 9
  accepts: 8
  confirms: 0
  setup-fails: 0
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  setup rate: 3 calls per minute
  accept rate: 3 calls per minute
  setup time: min: 228ms, max: 228ms, avg: 228ms
  hold time: min: 5000ms, max: 14000ms, avg: 10000ms
  disconnect time: min: 4ms, max: 4ms, avg: 4ms
  idle time: min: 5768ms, max: 14768ms, avg: 10196ms
  inter-digit-delay: min: 0ms, max: 0ms, avg: 0ms
  last disconnect cause: 16 normal call clearing.
  last abnormal disconnect cause: 0
  Script Stopwatch: N/A
Channel 2 Current Settings
  channel type is voice
  channel mode is originate
  maximum call rate is 3 calls per minute
  call duration has a random distribtion with min:5 and max:15 seconds
  minimum inter-call-delay is 0 seconds
  start-to-start-delay is 0 seconds
  start-time-delay is 0 seconds
  calling-number is 5550001
  redirection info is not configured
  called-number is 5552000
  setup-timeout is 300 seconds
  teardown-timeout is 300 seconds
  minimum path-confirmation time-out is 15 seconds
  minimum path-confirmation digit-on-time is 50 milliseconds
  minimum path-confirmation digit-off-time is 150 milliseconds
  minimum path-confirmation post-sending-delay is 600 milliseconds
  path-confirmation type is not configured
  voice-quality psqm-server is not configured, using udp protocol
  voice-quality recording path is not configured
  voice-quality max-audio-silence is 1600 milliseconds
  voice-quality hit-threshold is 200 milliseconds
  voice-quality logging latency is off
  voice-quality phase-locking algorithm is off
  minimum voice-quality psqm-inter-delay is 5 seconds
  qsig-message is off
  qsig-calltype is basic-call
  interface is voip:100
  call script is not configured
```

The output shows that eight calls have been accepted on channel 2, the minimum duration has been 5 seconds, maximum has been 14, and the average was 10 seconds.

Step 6 Limit the total number of calls Callgen generates. First, stop channel 2, clear out and reset all counters, and then start all three channels:

```
router(callgen)# stop 2
router(callgen)# clear counters
This will reset current statistics of calls. [confirm]y
router(callgen)# start total-calls 10000
router(callgen)#
```

Using the **start** command without specifying a particular channel starts all channels. The **total-calls** parameter specifies that Callgen places 10,000 calls across the three channels before stopping. Since this can take several hours to complete, use the **stop** command with no arguments to stop all channels.

Originating and Terminating Multiple, Simultaneous PRI Calls

This example uses Callgen back-to-back between two 5300 routers using an ISDN Primary Rate Interface (PRI) connection. For actual test scenarios, you would probably have a device or network under test between the two Callgen routers, such as a VoIP or VoATM gateway, or PBX.

In this example, a Callgen router named as5300-9 generates voice calls, each lasting 1 second, with 1 second pauses between each call. It does this on 10 of the 23 available B channels. A Callgen router named as5300-8 terminates these calls.

```
+-----------+                +-------------+
| as5300-9  |t1 0        t1 1| as5300-8    |
|callgen NET+------PRI--------+ callgen     |
|originate  |                | terminate   |
+-----------+                +-------------+
```

Note that the D-channel for t1 0 above is configured to be the network side of the connection, otherwise, a switch would be required. To configure t1 0 as the network side, use **isdn network**, **shut**, **no shut** in the D-channel configuration of t1 0, **serial0:23**, to enable back-to-back connection. We must also make sure that the t1 controllers and ISDN signaling are configured properly for voice.

Note This example assumes that the basic router configuration of the ISDN PRI interface and D channel are correct and working. The router's **show isdn status** command should indicate that all three layers are up and the **show controller t1** command should not indicate any framing errors after 15 minutes of use.

The following shows the relevant router configuration statements:

```
isdn switch-type primary-5ess

controller T1 0
  framing esf
  clock source line primary
  linecode b8zs
  pri-group timeslots 1-24

interface Serial0:23
  no logging event link-status
  isdn switch-type primary-5ess
  isdn protocol-emulate network
  isdn incoming-voice modem
```

Step 1 Configure the as5300-9 (originate side).

Note that on channel 1, we have overridden the class-specified duration of 1 second to be 5 minutes. Channel 1 is also configured to verify that the voice/bearer channel can pass voice traffic and that it has been connected to the right party, using the **path-confirmation** command to ping the called number in-band via DTMF tones.

```
class pri-orig type voice mode originate
  duration 1 seconds
  inter-call-delay 1 seconds
  called-number 19194721000
  interface port:0:D

channel 1 class pri-orig
  calling-number 19191111212
  duration 5 minutes
  path-confirmation type ping called-number

channel 2 class pri-orig
  calling-number 19192221212

channel 3 class pri-orig
  calling-number 19193331212

channel 4 class pri-orig
  calling-number 19194441212

channel 5 class pri-orig
  calling-number 19195551212

channel 6 class pri-orig
  calling-number 19196661212

channel 7 class pri-orig
  calling-number 19197771212

channel 8 class pri-orig
  calling-number 19198881212

channel 9 class pri-orig
  calling-number 19199991212

channel 10 class pri-orig
  calling-number 19190001212
```

Step 2 Configure the as5300-8 (terminate side).

```
class pri-term type voice mode terminate
  called-number 19194721000
  interface port:1:D
  exit

channel 1 class pri-term
  path-confirmation cut-through-time 2 seconds
  path-confirmation type ping called-number

channel 2 - 10 class pri-term
```

The **path-confirmation cut-through-time** command on channel 1 delays the start of the configured pings by 2 seconds to take into account that the call might be connected on the terminate side from a signalling perspective but the originate side might not be able to receive any in-band traffic for a short amount of time. If either side does not receive the ping, or the wrong digits are received, the path confirmation failures counter increments.

Using Callgen

This chapter describes how to use Callgen, including the different Callgen command modes and logging and debugging facilities.

Hardware Requirements

Voice, fax, and data call generation require the following:

- 5300—One to two Voice Feature Cards (VFC) (stop support after v4.1T)
- 5400—One NP Voice Feature Card
- 5350—Supports T1/E1 digital voice card
- 3600—Up to 12 FXS, FXO, E&M VIC cards or up to six digital voice cards
- 3700—Up to 8 FXS, FXO, E&M VIC cards or up to four digital voice cards
- 5800—One to two 192-DSP voice cards (stop support after v3.2 T)
- 5850—Supports T1/E1, T3
- 7200—Up to six digital voice cards

Fax calls can also be generated using Microcom modems. Data call generation requires MICA modems.

Security Issues

Starting with Callgen 2.2, all released Callgen images enforce a security key checking before starting IOS. This protection prevents Callgen images being given to customers without first registering the machine ID. To obtain a security key, go to http://wwwin-vts.cisco.com/protected-cgi/get_key.cgi. This centralized database of machine IDs helps us track the distribution of Callgen images as well as provide statistics on each platform.

Note Giving customers Callgen images without the Network Verification Services (NVS) contract is against the rules of Cisco Systems’ intellectual property (confidential) copyright act.

Command Line Prompt Mode

Callgen uses the IOS parser to accept configuration and control commands. All the IOS parser features (command completion, history, context-sensitive help, emacs-style editing, and so on) work exactly the same way in Callgen as they do at the IOS exec prompt.

To use the callgen commands, either prefix the command with the word **callgen** at the IOS exec prompt or type the command from within one of the following callgen modes.

Note By default, Callgen changes the prompt to identify channel or class name when in channel or class mode. To force Callgen to use fixed prompts in these modes, use the **set prompt static** command. To change back to the default, use **set prompt dynamic**.

Callgen Mode

Callgen mode is the main mode, used to control Callgen operations, turn on debugs, and show configuration and statistics.To enter callgen mode, type **callgen** at the IOS exec prompt:

```
router# callgen
router(callgen)#
```

Note Callgen is a single-user tool that is invoked from the router console on which it is running. Future versions will include code to ensure that only one user is logged into the router and actively using callgen.

Channel Mode

Channel mode is used to configure channel-specific parameters. You can enter channel mode in one of two ways:

- From the IOS exec prompt by entering **callgen channel** *num*, where *num* is the channel number you are defining:

```
router# callgen channel 4
router(cgch_4_vo_o)#
```

- From callgen mode by entering **channel** *num*, where *num* is the channel number you are defining or editing:

```
router(callgen)# channel 5
router(cgch_5_vo_o)#
```

The channel mode prompt reflects the current channel number, the call type, and call mode. For example, “cgch” indicates that this is the callgen channel mode, 4 is the channel number, “vo” is the call type (voice), and “o” is the call mode (originate). The current values for type and mode are:

Table 3-1 Type and Mode Abbreviations

Call Types	Call Modes
vo—voice call	**t—**terminate
du—dummy call	**o—**originate

For more information on channels, see Chapter 4, “Configuring Channels.”

Class Mode

Class mode is used to configure class-specific parameters. You can enter class mode in one of two ways:

- From the IOS exec prompt by entering **callgen channel** *name*, where, where *name* is the 1-15 character class name you are defining:

  ```
  router# callgen class voice
  router(cgcc_voice_vo_o)#
  ```

- From callgen mode by entering **class** *name*, where *name* is the 1 to 15 character class name you are defining or editing:

  ```
  router(callgen)# class foobar
  router(cgcc_foobar_vo_o)#
  ```

The class mode prompt reflects the current class name, call type, and call mode for the class you are defining. For example, “cgcc” indicates the callgen class mode, “foobar” is the current class name, “vo” is the call type (voice), and “o” is the call mode (originate). Valid type and mode abbreviations are the same as those in channel mode (see Table 3-1).

For more information on classes, see Creating Classes (page 4-11).

Global Mode

Global mode is used to configure call-type-specific global definitions of parameters used by channels and classes. To enter global mode, type **callgen global type** *call-type* **mode** *call-mode* in callgen mode. For example:

```
router(callgen)# global type voice mode originate
router(cgset_vo_o)#
```

The global mode prompt reflects the current call type and call mode. For example, “cgset” indicates the callgen global mode, “vo” is the call type (voice), and “o” is the call mode (originate). Valid type and mode abbreviations are the same as those in channel mode (see Table 3-1). For some call types, you do not need to specify the call mode to enter global mode. In this case, the prompt specifies “all,” indicating that the global definitions apply to both originate and terminate modes. For example:

```
router(callgen)# global type voice
router(cgset_vo_all)#
```

Exiting a Mode

To drop back to the previous mode, use the **exit** command. For example:

```
clash-5300(cgch_4_vo_o)# exit
clash-5300(callgen)#
```

All current Callgen configuration and state information is saved even if you exit out of callgen mode back to the IOS exec prompt. Channels continue originating and terminating calls, statistics are updated, and so on.

Returning to the IOS Exec Prompt

To go to the IOS exec prompt regardless of the current callgen mode, use the **end** command. For example:

```
clash-5300(cgch_4_vo_o)# end
clash-5300#
```

Entering Commands from the IOS Exec Prompt

You can run any callgen command from the IOS exec prompt by proceeding the command with **callgen**. After the command executes, you return to the IOS exec prompt. For example, if you enter **callgen show version** at the IOS exec prompt, the current callgen version is displayed without entering callgen mode. Entering **callgen channel 64 inter-call-delay 10 seconds** from the IOS exec prompt performs the exact same function as entering **callgen channel 64** to get into channel mode, followed by a separate **inter-call-delay 10 seconds** command.

Saving and Loading Callgen Configurations

You can save and load Callgen configurations to the IOS File System (IFS).

Saving a Configuration to IFS

save-config *url*

Saves the Callgen configuration to IFS. The saved configuration can be loaded later with the **load-config** command.

Examples

If you enter **save-config ?**, the program displays which file systems are available on the router.

```
(callgen)#save-config ?
           bootflash:       Save config to bootflash:
           disk0:           Save config to disk0:
           disk1:           Save config to disk1:
           flash:           Save config to flash:
           lex:             Save config to lex:
           null:            Save config to null:
           nvram:           Save config to nvram:
           pram:            Save config to pram:
           rcp:             Save config to rcp:
           slavebootflash:  Save config to slavebootflash:
           slavenvram:      Save config to slavenvram:
           slaveslot0:      Save config to slaveslot0:
           slaveslot1:      Save config to slaveslot1:
           slot0:           Save config to slot0:
           slot1:           Save config to slot1:
           system:          Save config to system:
           tftp:            Save config to tftp:
```

If you enter just the file system name, the program prompts you for the remaining information. For example, if you want to save the configuration to TFTP server 192.1.1.2 and to the filename */tftpboot/test/filters*.

```
(callgen)#save-config tftp
             Address or name of remote host []? 192.1.1.2
             IFS filename []? test/filters
             !!
             Save complete.
```

If the complete URL is entered, the program does not prompt for more information. In a TCL script, you must use the complete URL, because CSCCON does not know how to respond to callgen IFS prompts.

The following example shows using the command from the router exec with a complete URL.

```
(callgen)#save-config tftp://192.1.1.2/test/filters
             !!
             Save complete.
```

Loading a Configuration from IFS

load-config *url*

Loads a Callgen configuration file from IFS. It first deletes all existing Callgen configurations. It then reads in and executes the commands in the requested configuration file to create a new Callgen configuration.

The Callgen configuration file was created with the **save-config** command.

Examples

If you enter **load-config ?**, the program displays which file systems are available on the router:

```
(callgen)#load-config ?
       bootflash:       Load config from bootflash:
       disk0:           Load config from disk0:
       disk1:           Load config from disk1:
       flash:           Load config from flash:
       null:            Load config from null:
       nvram:           Load config from nvram:
       pram:            Load config from pram:
       rcp:             Load config from rcp:
       slavebootflash:  Load config from slavebootflash:
       slavenvram:      Load config from slavenvram:
       slaveslot0:      Load config from slaveslot0:
       slaveslot1:      Load config from slaveslot1:
       slot0:           Load config from slot0:
       slot1:           Load config from slot1:
       system:          Load config from system:
       tftp:            Load config from tftp:

(callgen)#load-config
```

If you enter just the file system name, the program prompts you for the remaining information. For example, if you want to load a filter from TFTP server 192.1.1.2, and read the file */tftpboot/test/filters*.

```
(callgen)#load-config tftp
     Address or name of remote host []? 192.1.1.2
     IFS filename []? test/filters
        Please wait until ‘Load Complete’ message.

     (callgen)#
     Loading test/filters from 192.1.1.2 (via Ethernet0/0/0): !
     [OK - 2360/4096 bytes]

        Load Complete.
```

If the complete URL is entered, the program does not prompt for more information. In a TCL script, you must use the complete URL, because CSCCON does not know how to respond to Callgen IFS prompts.

Logging Messages

You can have Callgen messages logged to an external server using the IOS Syslog messaging feature. To use this feature, start syslog daemon on the external server and configure IOS to send messages to the server. Callgen sends its messages to IOS, and IOS logs them to the server. The following is an example of how to set up the IOS configuration:

```
cg-5300-3#conf t
Enter configuration commands, one per line. End with CNTL/Z.
cg-5300-3(config)#logg on
cg-5300-3(config)#logg 10.0.1.51
cg-5300-3(config)#logg facility local0
cg-5300-3(config)#logg trap debugg
cg-5300-3(config)#end
```

For more information on IOS message classification, formats, and the above commands, see http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/dialnms/syslog.htm.

For information on configuring and starting syslog daemon on the external server, see http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/dialnms/syslog.htm#xtocid228378.

The following is a sample output from the external server regarding Callgen messages:

```
Mar 27 16:56:37 [10.0.2.8.208.173] 30: 21:55:40: %SYS-5-CONFIG_I: Configured from
console by console
Mar 27 16:56:59 [10.0.2.8.208.173] 31: 21:56:02: %CALLGEN_MSG-7-DEBUG: << Entered
newVoice>>
Mar 27 16:56:59 [10.0.2.8.208.173] 32: 21:56:02: %CALLGEN_MSG-7-DEBUG: << Exited
newVoice (0) >>
Mar 27 16:57:03 [10.0.2.8.208.173] 33: 21:56:08: %CALLGEN_MSG-7-DEBUG: << Entered
setParameter>>
Mar 27 16:57:03 [10.0.2.8.208.173] 34: 21:56:08: %CALLGEN_MSG-7-DEBUG: << Exited
setParameter (0) >>c
```

To see the messages on the console, use the command **logging console** in IOS config mode. To log messages to the external server, use the **logging** *ip_address* command in addition to the ones described above. To stop logging messages to the external server, use the **no logging** *ip_address* command.

Callgen maintains a timestamped log of messages that have occurred since the last **clear log** command. You can also log server radius messages.

By default, Callgen only displays and records messages considered to be errors. Like IOS debug commands, you can prevent Callgen messages from appearing on the console using the global configuration command **no logg console debug** message:

```
router# config t
router(config)# no logg console debug
```

Logging Options

To control the types of messages you get, use the following commands.

set log verbosity error

Displays and records error messages only (the default).

set log verbosity warn

Displays and records error messages or significant events only.

set log verbosity info

Displays and records any messages, including those that indicate normal and routine state transitions.

set log radius {**on** | **off**}

Turns logging radius server messages on or off. The default is off. This log buffer is also emptied when use the **clear log** command.

Setting the Log Size

Callgen maintains a buffer up to *n* K bytes of log messages. Once the buffer becomes full, a warning message is displayed and no more messages are recorded. Use the callgen **set log size** command to specify the number of K (1024) bytes that Callgen will preserve.

Displaying Log Messages

Use the **show log** and **show log radius** commands to display messages. To only display the messages containing a specific string, use **show log grep** *string*.

For example, **show log** produces the following output:

```
wizards-3640(callgen)# show log
ErrMod_05:22:45.282 Error Log Verbosity set to 2
CH1_05:23:27.206 control(start): instance 3 received duration of 3 sec, 3000 msec.
CH1_05:23:30.206 finishDummyCall(): instance 3 state: ‘teardown’, callStatus:
‘accepted’, callError: ‘no-error’,
setupTime:0 msecs, activeTime:3000 msecs, teardownTime:0 msecs
CH1_05:23:30.206 control(start): instance 3 received duration of 3 sec, 3000 msec.
CH1_05:23:33.206 finishDummyCall(): instance 3 state: ‘teardown’, callStatus:
‘accepted’, callError: ‘no-error’,
setupTime:0 msecs, activeTime:3000 msecs, teardownTime:0 msecs
CH1_05:23:33.206 control(start): instance 3 received duration of 3 sec, 3000 msec.
CH1_05:23:36.206 finishDummyCall(): instance 3 state: ‘teardown’, callStatus:
‘accepted’, callError: ‘no-error’,
setupTime:0 msecs, activeTime:3000 msecs, teardownTime:0 msecs
CH1_05:23:36.206 control(start): instance 3 received duration of 3 sec, 3000 msec.
CH1_05:23:39.206 finishDummyCall(): instance 3 state: ‘teardown’, callStatus:
‘accepted’, callError: ‘no-error’,
setupTime:0 msecs, activeTime:3000 msecs, teardownTime:0 msecs
CH1_05:23:39.206 control(start): instance 3 received duration of 3 sec, 3000 msec.
CH1_05:23:42.206 finishDummyCall(): instance 3 state: ‘teardown’, callStatus:
‘accepted’, callError: ‘no-error’,
setupTime:0 msecs, activeTime:3000 msecs, teardownTime:0 msecs
ErrMod_05:26:20.994 Error Log Size set to 65536 bytes
```

Here is sample output generated by executing **show log radius**:

```
cg-5300-5(callgen)#show log radius
16:08:19.575 Client-id: 110.1.0.1 Client-port-id: CH1 Status: Setup
16:08:19.675 Client-id: 110.1.0.1 Client-port-id: CH1 Status: Active
16:08:34.675 Client-id: 110.1.0.1 Client-port-id: CH1 Status: Disconnect
16:08:34.687 Client-id: 110.1.0.1 Client-port-id: CH1 Status: InActive
```

Debugging Options

Callgen has three types of debugs that you can optionally turn on: controller, parser, or CTM. Use the debug options only if needed to debug certain Callgen operational problems, because debugging can impact the system. Debugging should not be used on systems with heavy call generation volume.

To view the available debug options, use **debug ?**:

```
clash-5300(callgen)# debug ?
ctlr Debug controller
ctm Debug call type module
parser Debug callgen CSB and parm blocks
```

Turn debugging on with the **debug** *type* command:

```
clash-5300(callgen)# debug ctlr
Callgen controller debug enabled
```

Turn debugging off with the **no debug** *type* command:

```
clash-5300(callgen)# no debug ctlr
Callgen controller debug disabled
```

To view the current debug level settings, use the **show debug** command:

```
router(callgen)# show debug
Parser debug disabled
Controller debug disabled
CTM debug disabled
```

To help the technical support efforts, the command **show debug all** has been implemented. This command displays the output of the following commands all at once: **show running** (IOS), **show version** (IOS), **show version** (Callgen), and **show config** (Callgen). Or you can just use **show report**, capture the output, and send it to callgen-support (see show report (page 4-15)).

For example:

```
router(callgen)#sh debug all
IOS RUNNING CONFIGURATION

Building configuration...

Current configuration:
!
version 12.1
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname cg-train-7
!
no logging console
enable password lab
!
!
!
!
!
memory-size iomem 25
voice-card 2
!
ip subnet-zero
ip host CALLGEN-SECURITY-V2 21.63.69.5 0.90.0.0
ip host boot 10.0.2.10
ip dhcp smart-relay
!
cns event-service server
!
!
!
!
```

```
!
!
!
controller T1 2/0
 framing esf
 linecode b8zs
 ds0-group 1 timeslots 1-24 type e&m-wink-start
!
controller T1 2/1
 framing esf
 linecode b8zs
 ds0-group 1 timeslots 1-24 type e&m-wink-start
!
!
!
interface Ethernet0/0
 no ip address
 shutdown
 no cdp enable
!
interface Ethernet0/1
 ip address 10.0.2.37 255.255.255.0
 no cdp enable
!
interface Ethernet0/2
 no ip address
 shutdown
 no cdp enable
!
interface Ethernet0/3
 no ip address
 shutdown
 no cdp enable
!
ip kerberos source-interface any
ip classless
ip route 10.0.2.0 255.255.255.0 Ethernet0/0
no ip http server
!
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
!
voice-port 1/0/0
!
voice-port 1/0/1
!
voice-port 1/1/0
!
voice-port 1/1/1
!
voice-port 2/0:1
!
voice-port 2/1:1
!
!
line con 0
 exec-timeout 0 0
 transport input none
line aux 0
line vty 0 4
 login
!
end
```

```
IOS VERSION
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3640-TCGEN-M), ExperimentalVersion12.1(20000710:205900)
[afaruqui-ddts 100]
Copyright (c) 1986-2000 by cisco Systems, Inc.
Compiled Mon 10-Jul-00 13:59 by afaruqui
Image text-base: 0x60008948, data-base: 0x614FC000

ROM: System Bootstrap, Version 11.1(20)AA2, EARLY DEPLOYMENT RELEASE SOFTWARE(fc1)
ROM: 3600 Software (C3640-TSCGEN-M), Experimental Version
12.1(20000313:033814)[spchang-callgen2_2 240]

cg-train-7 uptime is 21 hours, 40 minutes
System returned to ROM by reload
System image file is "tftp://10.0.2.10/afaruqui/c3640-tcgen-mz"

cisco 3640 (R4700) processor (revision 0x00) with 49152K/16384K bytes of memory.
Processor board ID 17632726
R4700 CPU at 100Mhz, Implementation 33, Rev 1.0
Bridging software.
X.25 software, Version 3.0.0.
SuperLAT software (copyright 1990 by Meridian Technology Corp).
TN3270 Emulation software.
Primary Rate ISDN software, Version 1.1.
4 Ethernet/IEEE 802.3 interface(s)
2 Channelized T1/PRI port(s)
4 Voice FXO interface(s)
DRAM configuration is 64 bits wide with parity disabled.
125K bytes of non-volatile configuration memory.
32768K bytes of processor board System flash (Read/Write)

Configuration register is 0x0

CALLGEN VERSION INFORMATION

Callgen Version:        2.2.0
Controller Version:     2.0.0
VoiceCTM Version:       2.2.0
DummyCTM Version:       1.0.0
DataCTM Version:        1.1.0

CALLGEN CONFIGURATION


class v type voice mode originate
  start-called-number 514000

channel 1 class v
  called-number 514000
  voice-quality type psqm audio-file 1 no-sync
```

Technical Support

Please send support questions to callgen-support@cisco.com. For general questions and informal discussions among other Callgen users, send email to callgen-users@cisco.com. For a concise summary of all Callgen commands, refer to the Callgen Command Reference Card. All Callgen documents, images, and updates are located at http://wwwin-vts.cisco.com/bulkCall.

To help the technical support efforts, the command **show debug all** has been implemented. This command displays the output of the following commands all at once: **show running** (IOS), **show version** (IOS), **show version** (Callgen), and **show config** (Callgen). Or you can use **show report**, which displays the output of the following commands: **show aggregate** (Callgen), **show config** (Callgen), and **show running** (IOS)**.** Also, the commands **show log** and **show channel** also provide good information. Please capture as much information as possible and send it to callgen-support.

Defect Notification and Feature Requests

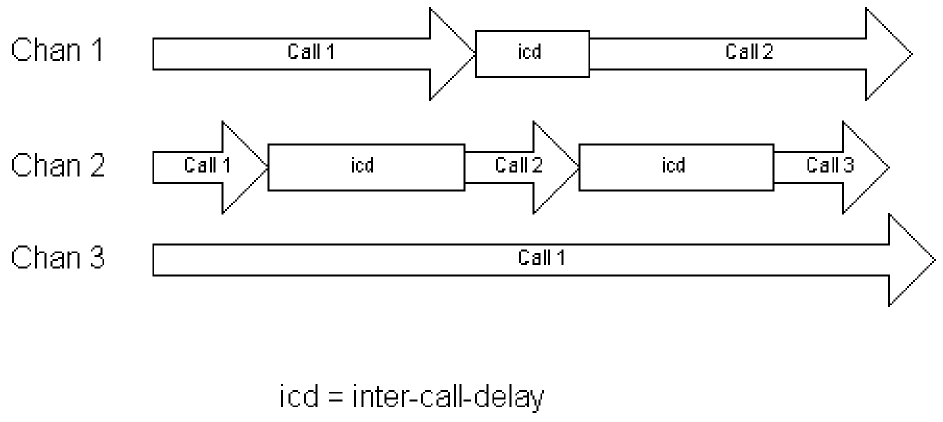
A list of the current defects is on the Known Callgen Defect (QDDTS query) page. Check here to make sure that the defect you want to report is not on the list. To report a new defect in either Callgen or this documentation, or to request a new Callgen feature, open a DDTS in the CSC.labtrunk project with a component name of callgen.

Configuring Channels

In Callgen, channels are the basic building block for placing calls. Each channel represents a single call execution; only one call at a time is occurring on a channel. Channel configuration parameters define the characteristics of the calls placed on the channel. To run multiple simultaneous calls, you configure multiple channels, each with its own set of parameters (call duration, call rate, and so on).

This chapter describes the generic configuration options available for all call types. For more information on configuration options for a specific call type, see the sections on the various call type modules.

The example below shows three channels, each with a different call duration and inter-call-delay setting.



Creating Channels

You can create a channel in either callgen mode (see Channel Mode (page 3-2)) or from the IOS prompt (Entering Commands from the IOS Exec Prompt (page 3-4)).When creating a channel, you can specify the type, mode, or class. You can also specify a range of channels.

The syntax is:

channel *firstNumber* [**-** *lastNumber*] [**type** *type* [**mode** {**originate** | **terminate**}]]
channel *firstNumber* [**-** *lastNumber*] [**class** *name*]

firstNumber—A number from 1 to 10000 that identifies the channel. The channel number does not affect the order of how actual calls are generated. Channel numbers do not have to be in sequence; for example, you can define channels 1, 7, 64, 245, and 599. However, it is helpful to group channels numerically. For example, make all voice calls on channels 100 to 199, fax calls on channels 200 to 299, and so on.

lastNumber—The last channel number when defining a range of channels.

When specifying a range, you must include spaces between the range value and the hyphen for the command to parse properly.

type—The type of call the channel represents. Options are **voice**, **data**, **fax**, and **dummy**. If not specified, the default is voice. A type is specified when initially creating the channel. To change a channel's call type, you must delete the channel and add it back again with the new type.

Note Callgen also provides configuration commands that apply to a particular call type only. For information on how to configure a specific call type, refer to the chapters on the different call type modules: Chapter 6, "Voice Call Type Module," Chapter 7, "Data Call Type Module," and Chapter 8, "Fax Call Type Module."

mode—The call mode, either **originate** or **terminate**. Originate channels place calls and terminate channels answer them. If not specified, the default is originate. To change a channel's mode, you must delete the channel and add it back again with the new mode.

class—The name of defined class. Classes offer an efficient way to group configuration parameters together and apply them to a number of channels. For information on creating classes, see Creating Classes (page 4-11).

Only one class can be assigned to a channel. The channel inherits all the configuration parameters specified in that class. If a class is not specified, the generic default values of parameters are used. To change a channel's class, you must delete the channel and add it back again with the new class.

After a channel is created, you can associate the channel with an interface (see Associating a Channel with an Interface (page 4-3)) and control the timing of calls (see Call Timing Parameters (page 4-4)).

Note You must specify an interface for voice channels.

You can define up to 10,000 channels, but you need to take CPU power, memory, and physical interfaces into account. For example, if you are placing POTS calls on a 3600 and have four physical FXO ports, four channels is sufficient to drive these.

Examples

Creates a channel of type voice and mode originate:

```
channel 1
```

Creates 10 channels of type voice and mode originate:

```
channel 1 - 10
```

Creates a channel of type voice and mode terminate:

```
channel 45 mode terminate
```

Creates a channel that uses the parameters defined in the class named mycall:

```
channel 3 class mycall
```

Deleting Channels

To delete a specified channel, use the prefix **no** prefix before the **channel** command:

no channel *number*

For example, from callgen mode:

```
router(callgen)# no channel 3
```

From IOS exec mode:

```
router# callgen no channel 3
```

You can erase all current channel and class configurations from memory using the callgen **clear config** command. For example:

```
router(callgen)# clear config
This will erase current configuration of Callgen. [confirm]y
router(callgen)#
```

Disabling Channels

To place a channel into a disabled state, use the **shutdown** command (this is similar to the way an interface is shutdown in IOS):

[**no**] **channel** *number* **shutdown**

The **no** option enables a channel.

By default, newly added channels are placed in a ready (no shutdown) state. To verify the state of configured channels, use the **show channel** command (see Displaying Channel Information (page 4-12)).

The following example shuts down a channel, displays its status, and then enables it again:

```
clash-5300(callgen)# channel 4
clash-5300(cgch_4_vo_o)# shutdown
clash-5300(callgen)# sh chan

ch-4-vo-o, state: DOWN, attempts: 0, accepts: 0, confirms: 0, if: ,
  setup-fails: 0, aborts: 0, disconnects: 0, confirm-fails: 0, other-fails: 0
clash-5300(callgen)# no channel 4 shutdown
ch-4-vo-o, state: INACTIVE, attempts: 0, accepts: 0, confirms: 0, if: ,
  setup-fails: 0, aborts: 0, disconnects: 0, confirm-fails: 0, other-fails: 0
```

Associating a Channel with an Interface

The **interface** command associates a channel with a physical interface on which to originate or terminate the call:

[**no**] **channel** *number* **interface** *interfacename*

Valid interfaces depend on the call type and mode of the channel. For instance, some interfaces on the router are capable of generating voice calls, while others are capable of generating modem calls.

interface is a required parameter for voice channels.

To see a list of valid interfaces for the channel’s type, use the **show interface** command. For example:

```
router(callgen)# show interface voice

Interfaces supporting voice/originate
voip:<dial-peer tag>
vofr:<dial-peer tag>
port:0:1
port:1:D
Interfaces supporting voice/terminate
voip:<dial-peer tag>
vofr:<dial-peer tag>
port:0:1
port:1:D
router(callgen)# channel 20 interface port:1:D
```

Call Timing Parameters

There are several configuration commands that control the timing of the call for the channel.

Specifying the Length of a Call

To specify the amount of time a call lasts, use the **duration** parameter:

[**no**] **duration** {*durationInSeconds* | **random** [**unique-seed**] *minSeconds maxSeconds*} [{**minutes** | **seconds** | **hours**}]

For example:

```
router(callgen)# ch 1
router(cgch_1_vo_o)# duration 3 minutes

router(callgen)# ch 2- 6
router(cgch_1_vo_o)# duration random unique-seed 30 50 seconds
```

The duration specified does not include the time required to set up or tear down the call, which varies depending on the call type and signaling. You can specify either an absolute value or a random value within a specific range. Units can be specified in seconds, minutes, or hours (the default is seconds). If keyword **unique-seed** is used together with keyword **random** in different channels, each channel generates a different pattern of random duration. The default is without **unique-seed**.

A duration of zero immediately disconnects the call once it is established. However, a duration of zero is not recommended for VoIP and analog calls, since the state transition of two ends might not be in sync.

The default value for originate channels is 0. The default value for terminate channels is 596 hours. We suggest not configuring duration for terminate channels and instead, allowing the originate channel to hang up the call. This prevents calls from being hung up before the originate channel's expected duration has expired. If the terminate channel then does hang up before it should, it means that something went wrong with the connection.

Note Random duration does not work if configured under the class configuration.

Delaying Call Generation

The **start-time-delay** parameter specifies the length of time to wait before call generation on the channel begins after the **start** or **run** command is issued. The default is 0.

[**no**] **start-time-delay** *delayInSeconds* [{**minutes** | **seconds** | **hours**}]

For example:

```
cg-train-6(cgch_1_vo_o)#start-time-delay 10
```

Specifying the Time Between Calls

The **inter-call-delay** parameter specifies the amount of time to wait between placing calls on the same channel. More specifically, it is the amount of time between tearing down the current call and starting to set up the subsequent call. The default is 0.

[**no**] **inter-call-delay** *delayInSeconds* [**minutes** | **seconds** | **hours**}]

For example:

```
cg-train-6(cgch_1_vo_o)#inter-call-delay 10
```

Delaying the Time Between Starting Calls

The **call-to-call-delay** parameter specifies the amount of time between the start of a call to the start of the subsequent call on the same channel. The default is 0.

[**no**] **call-to-call-delay** *delayInSeconds* [{**minutes** | **seconds** | **hours**}]

There is a potential conflict between the **call-to-call-delay** and **inter-call-delay** parameters. Callgen requires consistent timing configuration when **inter-call-delay**, **duration**, and **call-to-call-delay** are all specified. A valid configuration must equal the following:

call-to-call-delay == **duration** + **inter-call-delay**.

If the duration is random, the following rule applies:

min duration + **inter-call-delay** <= **call-to-call-delay** <= **max duration + inter-call-delay**.

For example, in the following, the call-to-call delay is 5*60+10:

```
cg-train-6(cgch_1_vo_o)#call-to-call-delay 310
```

Required Delay Time Between Starting Calls

The **reqd-call-to-call-delay** parameter specifies the maximum amount of time between the start of a call to the start of the subsequent call on the same channel. The default is 0.

[**no**] **reqd-call-to-call-delay** *delay* [{**minutes** | **seconds** | **hours**}]

There is a potential conflict between **reqd-call-to-call-delay** and the **inter-call-delay** and **call-to-call-delay** parameters. **reqd-call-to-call-delay** cannot co-exist with these other two parameters. A valid configuration must meet the following condition:

reqd-call-to-call-delay >= setup-timeout + duration + teardown-timeout + inter-call-delay

To maintain the absolute maximum call-to-call delay, call scripts or path confirmations are not allowed when **reqd-call-to-call-delay** is configured. In addition, variable duration is not supported.

reqd-call-to-call-delay is closely related to the **constant rate** parameters when using the **run** and **start** commands. If **reqd-call-to-call-delay** and **constant rate** are not configured, Callgen might not be able to maintain a constant call rate throughout the test duration (see Starting Call Generation (page 5-1)). **reqd-call-to-call-delay** by itself does not affect the call rates. This feature is only supported in the voice CTM.

Specifying the Number of Calls to Place

To specify the number of calls to place, use the **rate** parameter:

[**no**] **rate** *numberOfCalls* [**per** {**hour** | **minute** | **second**}]

If no time units are specified, the default is per second.

rate is closely related to the **call-to-call-delay** parameter in that they are different ways of expressing the same thing: how many calls to place during a certain duration. Because of this, only one of these parameters can be included in a channel's configuration; the most recent parameter specified is used. For example, if you have configured a channel with a call-to-call-delay and then specify a rate, the **rate** command appears in the configuration and the **call-to-call-delay** does not.

The same configuration checking applies to **rate** as to **call-to-call-delay** by replacing **call-to-call-delay** with **1/rate**.

Delaying the Start Between Channels

The **start-to-start-delay** parameter specifies the delay between the start of the previous channel to the start of the channel you are configuring.

[**no**] **start-to-start-delay** *delayInSeconds* [{**minutes** | **seconds** | **hours**}]

The default is 0.

In the following example, the configuration specifies that channel 2 starts 10 seconds after channel 1, and that channel 3 starts 5 minutes after channel 2. In terms of absolute start times, once you issue the **start** or **run** command, channel 1 would start immediately, channel 2 10 seconds later, and channel 3 in 5 minutes, 10 seconds into the run.

```
channel 1

channel 2
  start-to-start-delay 10 seconds

channel 3
  start-to-start-delay 5 minutes
```

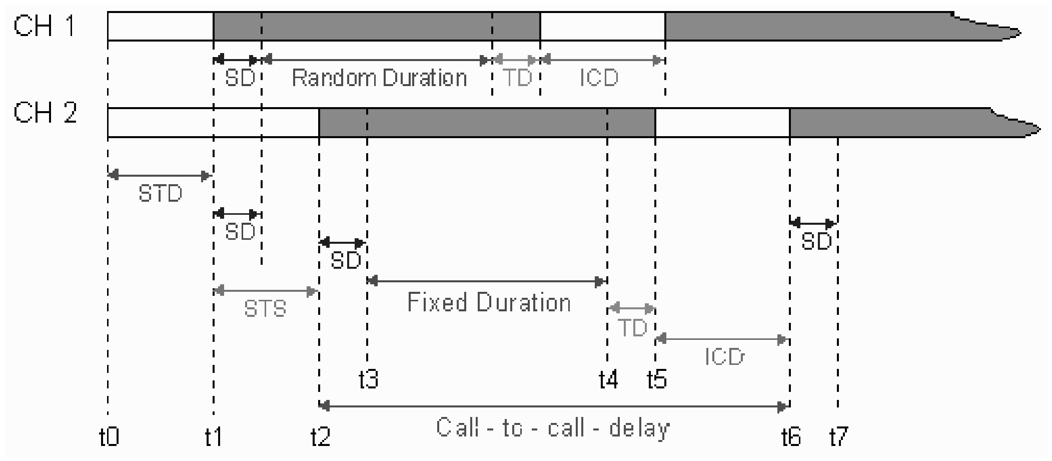
How the Time Parameters Work Together

The following diagram shows the various channel and call timing parameters. Time is on the horizontal axis, with specific times labeled t0, t1,t 2, t3, and t4. The following abbreviations are also used:

- SD = setup delay
- TD = tear-down delay
- STD = start-time delay
- STS = start-to-start delay
- ICD = inter-call delay

t0 represents the start of the test (the user has entered the start or run command). The start-time delay (time from t0 to t1) represents the time between the start of the test and when the first active channel (not shutdown) starts placing calls. The start-to-start delay (configured under channel 2), the time from t1 to t2, represents the delay between the start of the first channel and the start of the second channel. This allows the user to stagger or ramp up call generation. The setup delay (time from t2 to t3) represents the time between the start of a channel and when the call connection is established.

The duration (time from t3 to t4) represents the duration of the call on channel 2. The tear-down delay (time from t4 to t5) represents the time between the start of disconnecting a call and when the call is completely disconnected. The length of the setup delay and the tear-down delay can be bounded through the configurations of setup timeout and tear-down timeout.



Once started, the channel continues to place calls by waiting the specified inter-call-delay time, placing the call, holding the call active for the specified duration, and then terminating the call. Note that start-time delay and start-to-start delay are one time delays that apply only when the channel is first started. Furthermore, all of the above call timing parameters, with the exception of duration, can only be configured for originate channels (they do not apply to terminate channels).

Note In most cases, a channel configured on an FXO interface is not notified when its opposite endpoint disconnects, so it remains in a connected or hold state until its own duration time expires. Therefore, when terminating a call on an FXO interface, be sure to specify a duration that is a little longer than that of its corresponding originate channel. The actual added delay depends on the cut-through time of the call setup. Also make sure to specify an adequate inter-call delay on the originate channel.

Displaying Channel Configurations

show config [**channel** *beginchannel* [**-** *endchannel*]]

Displays the configurations of all currently configured channels, unless a single channel or a range of channels is specified. When specifying a range, you must include spaces between the range value and the hyphen for the command to be parsed properly.

If you need to reconstruct a channel configuration, you can cut and paste the output of this command into a callgen mode prompt. Future Callgen versions will allow saving this configuration to a TFTP server or Flash. Here is some sample output:

```
router(callgen)# show config

channel 1 type dummy mode originate
  duration 5 seconds
  dummy-signalling-mode duration
  dummy-random-delay setup 0 300000

channel 2 type voice mode originate
  duration 10 minutes
  calling-number 19194722900
  called-number 19194722824

router(callgen)#
```

Configuring Channel Pairs

By creating a channel pair, you can control the calls set up between two original channels.

[**no**] **ch-pair** {*channelNum peer_channelNum* | *channelNum*} - *end_channelNum*
[*peer_channelNum* - *end_peer_channelNum*]

Currently, when a channel starts, it places calls over and over again on the channel until the call duration expires. With channel pairing, when a call on a channel completes, a call on the paired channel is set up (the first channel remains idle). When the call on the second channel completes, a call on the first channel is set up. This continues until the session duration expires, or either of the channels are stopped.

For example, in the following, channel 1 and 2 are paired:

```
channel 1 type voice mode orig
   duration 5 seconds
   interface port:0:D

channel 2 type voice mode orig
   duration 5 seconds
   interface port:0:D

ch-pair 1 2
```

Pairing a Range of Channels

The **ch-pair** command can also pair a range of originate channels. For example, the following pairs channels 3 to 5 with channels 6 to 8 (that is, the pairs are 3 and 6, 4 and 7, 5 and 8):

```
ch-pair 3 - 5 6 - 8
```

When one of the channels in a pair is started or stopped, the other channel is automatically started or stopped. If both channels in a pair are specified as arguments to a **start** or **stop** command, the lower-numbered channel is started or stopped first. For example, the following starts channels 3 and 6. Channel 3 starts first.

```
ch-pair 3 - 5 6 - 8
start 3
```

This command starts channels 3 and 6, however, channel 6 starts first.

```
start 6
```

This command starts channels 3 to 5 and their paired channels 6 to 8.

```
start 3 – 5
```

Using Channel Pairing for Bi-directional Testing

The **ch-pair** command is also useful for setting up a bi-directional calling testbed. For example, assume the following testbed:

```
Callgen Router (port 0) ----------> UUT
   (port 1)                          |
         ^                                |
         |________________________________|
```

In the following, the originate and terminate channels are on same router. Channel 1 sets up a call that is answered by channel 2. When the call on channel 1 completes, channel 3 does a setup that is gets answered by channel 4 (note that this call is in the reverse direction from the first call). When channel 3 completes, a call on channel 1 is generated. For bi-directional calling to work, the originate channels must be on the same router.

```
chan 1 mode originate
  called-number 5678
  duration 5 seconds
  interface port:0:D

chan 2 mode terminate
  called-number 5678
  interface port:1:D

chan 3 mode originate
  called-number 1234
  duration 5 seconds
  interface port:1:D

chan 4 mode terminate
  called-number 1234
  interface port:0:D

ch-pair 1 3

callgen> start
```

Configuring Early Disconnected Calls

The **accept-premature-disc** command handles an early disconnected call that has no other error as a passed call and updates the passed-calls counter accordingly when generating channel statistics.

[**no**] **accept-premature-disc**

You can configure this option on both a channel and a class. It should be enabled if you expect that the terminate channel could also initiate call disconnection. This option is disabled by default.

Note Currently, this command is only applicable to **voice** calls.

Configuring Thresholds

The **threshold** command sets the threshold on call counters associated with a channel. You can configure thresholds on a channel or a class.

[**no**] **threshold** *callCounterName* [**in-percent**] *operator* {*value* | *percent*}

You can set thresholds on the following call counters: accepts, confirms, setup-fails, aborts, disconnects, confirm-fails, and other-fails. The threshold value can be specified either in percentage or absolute value. If the value is specified in percentage, use the keyword **in-percent**.

The operator for accepts and confirms is <=. For all other call counters, it is >=. The range of the threshold is 1-MAXINT when absolute values are used. When percentages are used, the range is 1-100. The **show channel** command displays the channels that have exceeded the configured threshold (see Displaying Channel Information (page 4-12)).

Examples

Sets the threshold to see all channels with setup-fails above 10 percent:

```
threshold setup-fails in-percent >= 11
```

Sets the threshold to see all channels with accepts below 90%:

```
threshold accepts in-percent <= 89
```

Sets the threshold to see all channels with aborts above 4:

```
threshold aborts >= 5
```

Sets the threshold to see all channels with aborts above 0:

```
threshold aborts >= 1
```

Enabling Standard Deviation

[**no**] **stddev**

Enables the channel to gather call setup and tear-down information for standard deviation calculation at a later time. The command **show aggregate** triggers the calculation of standard deviation for all the channenls in the specified group of the command. If one of the channels in the group does not enable the stddev information gathering, **show aggregate** does not start the standard deviation calculation. The standard deviation calculation is based on all the calls of all the originate channels specified by **show aggregate**.

Displaying Aggregate Statistics (page 5-3)

Caution This feature demands memory of the Callgen router to store extra information for each call. It also slows down system performance. Try to execute **show aggregate** after Callgen finishs making calls.

```
show agg

 Elapsed time of session: 00w0d00:13:22.692 and counting
  Originate Statistics
    max# of concurrent calls: 1
    active channels: 0 of 1
    setup attempts: 3
    accepts: 3
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    setup rate: 9 calls per hour
    accept rate: 9 calls per hour
    setup time: min: 4056ms, max: 4064ms, avg: 4061ms, stddev: 3ms
    hold time: min: 10000ms, max: 10000ms, avg: 10000ms
    disconnect time: min: 800ms, max: 800ms, avg: 800ms, stddev: 0ms
    idle time: min: 2000ms, max: 2000ms, avg: 2000ms
```

Creating Classes

If you have many channels with common parameters, you can define a class that contains the shared configuration values. The configuration parameters can then be applied to each channel with the **class** configuration command when the channel is created.

class *classname* [**type** *type* [**mode** {**originate** | **terminate**}]]

Each class has a text name (up to 15 characters) that is used to identify it. You can give the class a name that describes the call.

All configuration options that are valid for channel configuration can be included in a class, with the exception of the **class** and **shutdown** commands.

To change a class's type or mode, you must first delete the class and then redefine it.

The following example defines a class named short-voice. The class is then assigned to channels 1 and 2:

```
class short-voice
  rate 5 per minute
  duration random 5 10 seconds
  calling-number 919-472-2900

channel 1 - 2 class short-voice
```

Deleting Classes

You cannot delete a class if there are channels that refer to the class. You must first delete all channels that refer to the class, and then delete the class. To delete all channels and classes, use the **clear config** command (see Deleting Channels (page 4-3)).

To delete a specific class, use the prefix **no** before the **class** command:

no class *classname*

For example, from callgen mode:

```
router(callgen)# no class ISPcalls
```

From IOS exec mode:

```
router# callgen no class ISPcalls
```

Overriding Class Parameters

Any class parameter can be overridden by specifying it under the channel. For example, we define the following class:

```
class foobar
  duration 3 minutes
```

We then assign the class to the following channels:

```
channel 1 class foobar
  duration 1 minutes
channel 2 class foobar
```

Since channel 1 also has a duration configured under the channel configuration, the channel value overrides the class-specified value. So channel 1 places calls with a duration of 1 minute, while channel 2 places calls with a duration of 3 minutes (as specified in the class).

Auto Called/Calling Number Generation

To generate called numbers automatically for a large number of channels, you need to create a fax or voice originate/terminate class and use the command **start-called-number** *number* to specify the starting called number.

Once this has been specified, you can add the increment value that each subsequently generated called number should be increased by. This is done with the **called-increment-step** *step* command. If this command is not specified, the default increment is 1, meaning that the called numbers are generated sequentially with a difference of one between them. Make sure that the called number has no spaces or dashes (-).

The calling number can also be generated in the same way in voice and fax classes. The only difference is that it is only applicable for an originating class. The command **start-calling-number** *number* is specified to indicate the starting calling number from where to generate the numbers. Similarly, the **calling-increment-step** *step* command specifies an increment value for the generation of the subsequent calling numbers.

After making these changes in the class, when a channel is created with the class, it will have this feature. For example, in the following class v_o has the following parameters:

```
class v_o type voice mode originate
   start-called-number 5140000
   start-calling-number 6150000
   calling-increment-step 3
   called-increment-step 2
```

Then we issue the following command:

```
class v_o type voice mode originate
   router(callgen)#chan 1 - 3 class v_o
```

The following channels are created in the configuration as a result of this. These channels can be used normally for making and terminating calls.

```
channel 1 class v_o
  calling-number 6150000
  called-number 5140000

channel 2 class v_o
  calling-number 6150003
  called-number 5140002

channel 3 class v_o
  calling-number 6150006
  called-number 5140004
```

Note Currently, the called and calling numbers cannot have more than 10 digits and cannot have any characters other than digits. In addition, these numbers cannot start with 0.

Displaying Channel Information

The **show channel** command, entered with no arguments, displays a summary of all configured channels, including the current state and various call counters.

```
clash-5300(callgen)# show channel

ch-1-du-o, state: IDLE, attempts: 1, accepts: 1, confirms: 0, if: ,
setup-fails: 0, aborts: 0, disconnects: 0, confirm-fails: 0, other-fails: 0
passed-calls: 1, failed-calls: 0
```

```
ch-2-vo-o, state: INACTIVE, attempts: 0, accepts: 0, confirms: 0, if: ,
setup-fails: 0, aborts: 0, disconnects: 0, confirm-fails: 0, other-fails: 0
passed-calls: 0, failed-calls: 0
```

The output includes the following information:

Field	Description
ch-*#*-*type*-*mode*	Indicates the channel number, call type, and mode. *type* is either **vo** for a voice channel or **du** for a dummy channel. Mode is either **o** for originate or **t** for terminate.
state: *STATE*	Indicates the current state of the channel. Can be one of the following: **DOWN**—channel has been shutdown **INACTIVE**— channel has not been started **IDLE**—channel is running but currently not in a call **SETUP**—channel currently setting up a call **HOLD**—call has been setup and accepted **DISC**—call is currently being torn down
attempts	Number of call attempts on this channel. Callgen increments the attempt counter after it sucessfully initiates a setup request to the underlying IOS layer. It is possible that the attempt counter is not incremented after one “try-to-attempt,” but one or more of the error counters, usually the other-errors counter, are incremented after one “try-to-attempt.” Also, it is possible to issue the command **start total-calls 10**, but Callgen only attempts to make five calls. There could be a “pre- call-setup request error” or “configuration error.” **Note** If a “try-to-attempt” is not counted toward an “attempt”, it does not counted toward “setup-fails” either. Since the value of the “attempt” counter does not always match the number of calls you specify in the **start** command, do not check the attempt counter in a loop in your automation scrip. If you do, an infinity loop might happen when Callgen has a problem with initiating call setups. To check whether Callgen has completed the test after issuing **start total-calls** *n*, or any **start** command, you should check if all the Callgen originate channels have gone back to INACTIVE state and all the terminate channels have gone back to IDLE state, or an INACTIVE state if **stop** is issued. The easiest way is to do this is to check the output of **Callgen show aggregate**.
accepts	Number of call accepts for this channel. For example, number of times the call was answered by the called party.
confirms	Number of calls with successful path confirmation. Path confirmation is considered successful if no errors occur for the duration of the call and at least one path confirmation cycle completes. Thus, it is possible for a call to be abnormally disconnected or aborted and still have path confirmation succeed.
if	Indicates the interface port name or dial-peer tag number associated with this channel.
setup-fails	Number of calls failed during call setup state.
aborts	Number of calls terminated by the user. For example, the user entered a **stop** command.
disconnects	Number of calls abnormally disconnected. For an originate channel, any disconnect from the remote endpoint is considered abnormal.
confirm-fails	Number of calls with path confirmation failures. A path confirmation failure occurs when an incorrect tone or DTMF digit is received or when a timeout occurs.
other-fails	Number of calls with failures not meeting the above classes. Currently, the conditions that cause this counter to increment are channel misconfiguration failures; if path-confirmation type is configured and a call is disconnected before path-confirmation finishes one trial; and someone stops/aborts a channel while the current trial of path-confirmation is ongoing and there is no previous path-confirmation error for this call.

Field	Description
passed-calls	Number of calls without any failure indicated by any of the error counters. If **accept-premature-disc** is enabled, an early disconnect call without signaling error and without path-confirmation error is treated as a passed call. This counter gathers aggregate information, so you can look at this counter instead of individual counters if you are only interested in knowing whether a call passed. See Configuring Early Disconnected Calls (page 4-9).
failed-calls	Number of calls with any failure indicated by any of the error counters. If **accept-premature-disc** is not enabled, an early disconnect call, with or without signaling error and without path-confirmation error, is treated as a failed call. This counter gathers aggregate information, so you can look at this counter instead of individual counters if you are only interested in knowing whether a call passed. See Configuring Early Disconnected Calls (page 4-9).

With the **show channel** command, you can also specify a numerical range, a class name, or a specific call state to limit the display. For example,

```
wizards-3640(callgen)# show channel 3 - 9
wizards-3640(callgen)# show channel 3 -
wizards-3640(callgen)# show channel in-class CAS_orig
wizards-3640(callgen)# show channel in-hold-state
```

If you specify a specific channel number, the output is broken up into two sections. The first section lists the call statistics, and the second section details how the channel is configured. The format and content vary with call type and mode. For example:

```
wizards-3640(callgen)# show channel 1

Channel 1 Call Statistics
  elapsed channel run time: 00w0d00:00:09.004
  channel state: INACTIVE
  setup attempts: 3
  accepts: 3
  confirms: 0
  setup-fails: 0
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  passed-calls: 3
  failed-calls: 0
  setup rate: 13 calls per minute
  accept rate: 13 calls per minute
  setup time: min: 0ms, max: 0ms, avg: 0ms
  hold time: min: 3000ms, max: 3000ms, avg: 3000ms
  disconnect time: min: 0ms, max: 0ms, avg: 0ms
  idle time: min: 0ms, max: 0ms, avg: 0ms
  idleTimeInMilliseconds 0

Channel 1 Current Settings
  channel type is dummy
  channel mode is originate
  minimum call-to-call-delay is 0 seconds
  call duration is 3 seconds
  minimum inter-call-delay is 0 seconds
  start-to-start-delay is 0 seconds
  start-time-delay is 0 seconds
  dummy originate channel channel #1 instance #1 signalling-mode(Default):duration
if(Default):’’
  channel-state:idle call-status:accepted error-status:no-error
  current call - setupMsec:0 activeMsec:3000 terminateMsec:0 idleMsec:0
  random config parameters:
```

```
setup(Default) min:0 msec max:0 msec
active(Default) min:0 msec max:0 msec
teardown(Default) min:0 msec max:0 msec
```

show report

The **show report** command lists call statistics and configuration information, which can be used to archive the current results and all the configuration settings for a test run. You can also choose the amount of detail to display, using one of the following options:

show report [**brief**]

Displays concatenated output of the callgen **show** and **show config** commands, and the IOS **show run** command. For example:

```
cg-5300-7(callgen)#show report
   Building configuration...
   Aggregate Call Statistics
      Elapsed time of session: 00w1d06:17:30.020 and counting
      Originate Statistics
      active channels: 0 of 0
      setup attempts: 0
      accepts: 0
      confirms: 0
      setup-fails: 0
      aborts: 0
      abnormal disconnects: 0
      confirmed errors: 0
      other errors: 0
      setup rate: 0 calls per millisecond
      accept rate: 0 calls per millisecond
      setup time: min: 0ms, max: 0ms, avg: 0ms
      hold time: min: 0ms, max: 0ms, avg: 0ms
      disconnect time: min: 0ms, max: 0ms, avg: 0ms
```

show report summary

Displays concatenated output of the callgen **show channel** and **show config** commands, and the IOS **show run** command. For example:

```
cg-5300-7(callgen)#show report summary
   Building configuration...

   ! Callgen Config:

   channel 5001 type voice mode terminate
      called-number 7250001
      path-confirmation type ping called-number
      path-confirmation cut-through-time 2 seconds
      path-confirmation digit-on-time 150 milliseconds
      path-confirmation digit-off-time 50 milliseconds
      interface port:3:1

   !Current router configuration:
   !
   ! Last configuration change at 21:46:55 UTC Sat Jan 1 2000
   !
```

show report detail

Displays concatenated output of callgen **show channel** *num* command for all channels as well as the callgen **show config** and IOS **show run** commands. For example:

```
cg-5300-7(callgen)#show report detail
   Building configuration...

   Channel 5001 Call Statistics
      elapsed channel run time: 00w0d15:24:52.432 and counting
      channel state: IDLE
      setup attempts: 0
      accepts: 0
      confirms: 0
      setup-fails: 0
      aborts: 0
      abnormal disconnects: 0
      confirmed errors: 0
      other errors: 0
      passed-calls: 0
      failed-calls: 0
      setup rate: 0 calls per millisecond
      accept rate: 0 calls per millisecond
      hold time: min: 0ms, max: 0ms, avg: 0ms
      inter-digit-delay: min: 0ms, max: 0ms, avg: 0ms
      last disconnect cause: 0
      last abnormal disconnect cause: 0
      Script Stopwatch: N/A
```

Displaying Threshold Information

The **show threshold-exceeded** displays the channels exceeding the configured thresholds. The output reflects the current status of the call counters.

show threshold-exceeded [{*callCounterName* | **channel** *channelNum*}

The command can be used in the following three ways:

show threshold-exceeded *callCounterName*

Displays the channels exceeding the configured threshold for the specified call counter.

show threshold-exceeded channel *channelNum*

Displays the thresholds that have been exceeded on the specified channel number.

show threshold-exceeded

Displays a list of channels exceeding the configured thresholds for all call counters.

For example:

```
cg-train-9(callgen)#show threshold-exceed setup-fails

Channel#        Threshold: setup-fails         Current Value
-------------------------------------------------------------
2               >=     90%                        94%
4               >=     90%                        92%
5               >=     90%                       100%
6               >=     90%                       100%

cg-train-9(callgen)#show threshold-exceed channel 4

Channel 4                      Threshold                Current Value
----------------------------------------------------------------------
setup-fail threshold           >=     90%                        92%
accepts threshold              <=      3                          1

cg-train-9(callgen)#show threshold-exceed
```

```
Channel#        Threshold: confirmed-error     Current Value
-------------------------------------------------------------


Channel#        Threshold: aborts              Current Value
-------------------------------------------------------------
1               >=      1                         1


Channel#        Threshold: setup-fails         Current Value
-------------------------------------------------------------
2               >=     90%                       94%
4               >=     90%                       92%
5               >=     90%                      100%


Channel#        Threshold: abnormal-disconnect Current Value
-------------------------------------------------------------


Channel#        Threshold: other-error         Current Value
-------------------------------------------------------------


Channel#        Threshold: accepts             Current Value
-------------------------------------------------------------
1               <=      3                         3
2               <=      3                         1
4               <=      3                         1


Channel#        Threshold: confirms            Current Value
-------------------------------------------------------------
```

show log threshold

The **show log threshold** displays the information contained in the threshold log. A message is recorded in the log whenever a channel exceeds the set threshold for a call counter. Messages are logged on a channel and counter basis.

For example:

```
cg-train-9(callgen)#show log threshold
CH2_00:01:35.067 setup-fail threshold, current threshold 100% is >= configured
threshold 90%
CH2_00:01:35.067 accepts threshold, current threshold 0 is <= configured threshold 3
CH3_00:01:35.071 setup-fail threshold, current threshold 100% is >= configured
threshold 90%
CH3_00:01:35.071 accepts threshold, current threshold 0 is <= configured threshold 3
CH4_00:01:35.071 setup-fail threshold, current threshold 100% is >= configured
threshold 90%
CH4_00:01:35.071 accepts threshold, current threshold 0 is <= configured threshold 3
CH5_00:01:35.071 setup-fail threshold, current threshold 100% is >= configured
threshold 90%
CH5_00:01:35.071 accepts threshold, current threshold 0 is <= configured threshold 3
CH1_00:02:07.755 setup-fail threshold, current threshold 90% is >= configured threshold
90%
CH1_00:02:35.043 aborts threshold, current threshold 1 is >= configured threshold 1
```

Placing Calls

After you have configured channels, you can start placing calls. This chapter explains how to start and stop call generation.

Starting Call Generation

Callgen has two commands for placing calls: **run** and **start**. The **run** command is synchronous in that it does not return until call generation is complete. The **start** command is asynchronous; it immediately returns you to the callgen mode prompt so that you can interactively query call status as calls are placed.

The parameters for both commands are exactly the same:

run | **start** [*beginchannel* [- *endchannel*]] [**total-calls** *callcount*]
[**test-duration** *duration* [**hours** | **minutes** | **seconds**] [**end-gracefully**]]
[[**constant**] **rate** *number_of_calls* **per** [**hour** | **minute** | **second**]]

beginchannel [- *endchannel*]—You can specify a single channel or a range of channels. If a channel is not specified, the command applies to all configured, active (not shutdown) channels.

When specifying a range, you must include spaces between the range value and the hyphen for the command to parse properly.

total-calls—Number of call attempts (across all the channels currently placing calls) before stopping.

test-duration—Number of minutes (default), seconds, or hours to run before stopping.

end-gracefully—Defers disconnecting the current call until its expected duration has expired.

constant—Generates constant call set-up rates across active originate channels. This option only applies to the **rate** parameter. To enable this option, you must also configure **reqd-call-to-call-delay** for all the channels in a test (see Required Delay Time Between Starting Calls (page 4-5)). The test is aborted if there are not enough channels to generate the specified call rate. This option is only implemented for the voice CTM.

rate—Number of calls to place in the specified time. Callgen attempts to provide this aggregate call rate across the active originate channels that the **start** or **run** command applies to.

Note You should always start terminate channels first so that they will be ready to answer incoming calls from originate channels.

For example, the following generates 10,000 calls on all configured originate channels:

```
start total-calls 10000
```

The following generates calls for eight hours on channels 3 through 10:

```
start 3 - 10 test-duration 8 hours
```

If no time parameters are specified, Callgen places calls repeatedly on each channel until either a callgen **stop** command is entered at the prompt or Control + shift + 6 is pressed.

Note Callgen automatically creates a dial peer for each terminated called number. These dial peers are necessary for incoming calls to be handed off to Callgen properly. Do not remove or modify these auto-generated dial peers.

Stopping Call Generation

You can stop call generation using the Callgen **stop** command. If you use the **stop** command without any parameters, it terminates call generation on all active channels immediately.

stop [*beginchannel* [- *endchannel*] [**deferrably** | **gracefully | immediately**] [**call**]

beginchannel [- *endchannel*]—You can specify a single channel or a range of channels. If a channel is not specified, the command stops all currently running channels.

When specifying a range, you must include spaces between the range value and the hyphen for the command to parse properly.

deferrably—Stops all channels as soon as possible, as if normal duration expired. No call script is aborted.

gracefully—Stops all channels when the call’s duration expires. No channel is aborted.

call—Stops the channel and instead of bringing the channel state to INACTIVE state, it changes to IDLE state. This option is only applicable for terminate channels.

For example, the following stops channels 1 through 90 when normal call duration has expired:

```
stop 1 - 90 gracefully
```

Clearing Call Statistics Before Placing Calls

The Callgen **clear counters** command resets all generic and call type specific statistics and counters. By default, Callgen does not reset statistics before starting call generation. If you want statistics to be zeroed out before placing calls, issue this command before the callgen **run** or **start** command.

The options are as follows. If no options are specified, all counters are cleared.

clear counters *beginchannel* [- *endchannel*]

Clears counters in the specified channel number or a range of channels.

clear counters in-class *classname*

Clears counters in channels belonging to the specified class.

Displaying Call Statistics

The callgen **show** command, entered without any additional arguments, displays a summary of aggregate call statistics for originate and terminate sessions, including the number of active channels, elapsed time of the current run, and general call statistics. For example:

```
wizards-3640(callgen)# show

Aggregate Call Statistics
  Elapsed time of session: 00w6d22:36:53.364
  Originate Statistics
     max# of concurrent calls: 6
    active channels: 0 of 6
    setup attempts: 251
    accepts: 251
    confirms: 0
    setup-fails: 0
    aborts: 1
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    passed-calls: 250
    failed-calls: 1
    setup rate: 1 calls per hour
    accept rate: 1 calls per hour
    setup time: min: 228ms, max: 4208ms, avg: 243ms
    hold time: min: 3840ms, max: 300592ms, avg: 10308ms
    disconnect time: min: 4ms, max: 440ms, avg: 5ms
    idle time: min: 5768ms, max: 14768ms, avg: 10599ms
  Terminate Statistics
    active channels: 0 of 3
    setup attempts: 251
    accepts: 251
    confirms: 0
    setup-fails: 0
    aborts: 1
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    passed-calls: 250
    failed-calls: 1
    setup rate: 1 calls per hour
    accept rate: 1 calls per hour
    hold time: min: 4052ms, max: 302584ms, avg: 10533ms
```

Note The **show** command without any other option shows the call statistics accumulated over a period of time. The **show aggregate** command shows the statistics calculated on the fly. The **show** command might give inconsistent statistics because of a timing glitch but has little impact on router performance. The **show aggregate** command has a lot of impact on router performance if there are many channels and many calls per channel. It is not advisable to execute **show aggregate** while there are active calls.

Displaying Aggregate Statistics

The **show aggregate** command displays a summary of call aggregate statistics for both originate and terminate sessions. This command also provides detail statistics on active channels. For example:

```
cg-5300-8(callgen)# show aggregate

 Elapsed time of session: 00w0d00:00:10.980 and counting
```

```
   Originate Statistics
    max# of concurrent calls: 2
    active channels: 2 of 2
      HOLD: 2
    setup attempts: 2
    accepts: 2
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    passed-calls: 0
    failed-calls: 0
    setup rate: 0 calls per millisecond
    accept rate: 0 calls per millisecond
    setup time: min: 212ms, max: 212ms, avg: 212ms
    hold time: min: 0ms, max: 0ms, avg: 0ms
    disconnect time: min: 0ms, max: 0ms, avg: 0ms
    idle time: min: 0ms, max: 0ms, avg: 0ms
   Terminate Statistics
    active channels: 2 of 2
       HOLD: 2
    setup attempts: 2
    accepts: 2
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    passed-calls: 0
    failed-calls: 0
    setup rate: 0 calls per millisecond
    accept rate: 0 calls per millisecond
    hold time: min: 0ms, max: 0ms, avg: 0ms
```

If standard deviation is enabled for all the channels that are used to calculate the aggregated statistics for **show aggregate**, the output of **show aggregate** shows the standard deviation entries as well (see Enabling Standard Deviation (page 4-10)).

```
cgt-3660-4(callgen)# how agg

 Elapsed time of session: 00w0d00:13:22.692 and counting
  Originate Statistics
    max# of concurrent calls: 1
    active channels: 0 of 1
    setup attempts: 3
    accepts: 3
    confirms: 0
    setup-fails: 0
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    passed-calls: 3
    failed-calls: 0
    setup rate: 9 calls per hour
    accept rate: 9 calls per hour
    setup time: min: 4056ms, max: 4064ms, avg: 4061ms, stddev: 3ms
    hold time: min: 10000ms, max: 10000ms, avg: 10000ms
    disconnect time: min: 800ms, max: 800ms, avg: 800ms, stddev: 0ms
    idle time: min: 2000ms, max: 2000ms, avg: 2000ms
```

You can also specify a numerical range, a class name, or a specific mode to limit the display. Here are the options:

> **show aggregate mode** {**cgModeOriginate** | **cgModeTerminate**}
>
> **show aggregate channel** *startChannel* **-** [**mode** {**cgModeOriginate** | **cgModeTerminate**}]
>
> **show aggregate channel** *startChannel* **-** *endChannel* [**mode** {**cgModeOriginate** | **cgModeTerminate**}]
>
> **show aggregate channel in-class** *className*

In addition, it is possible to see the aggregate voice-related statistics by issuing the command in callgen mode:

> **show aggregate voice**—Shows the aggregate voice statistics for all channels.
>
> **show aggregate voice channel** *n* **-** [*m*]—Shows the aggregate voice statistics for channels ranging from *n* to *m*.

The output of the above commands looks like this:

```
cg-as5400-3(callgen)#sh aggregate voice

Originate Voice Channels Statistics
  Avg. Jitter Lead: 0ms
  Avg. Jitter Lag : 0ms
  Avg. E2E Delay  : 0ms
  Avg. RTT        : 0ms
  Hits            : 0
  Clips           : 0
  Max_Hit_Erros   : 0
```

Voice Call Type Module

The voice call type module (CTM) is written on top of IOS Symphony and, therefore, inherits Symphony's underlying functionality and capabilities. The voice CTM supports origination and termination of calls on T1/E1 ISDN PRI trunks, T1/CAS trunks, POTS interfaces (FXS, FXO, and E&M), VoIP, and VoFR.

Voice CTM Configuration Parameters

The following sections detail the voice-specific CTM parameters and commands.

Hanging Up a Call

The **hang-up** command forces the current call to disconnect. It has the same effect as if the configured call duration expired.

Example:

```
clash-5300(callgen)# channel 64 hang-up
```

Specifying the Calling Address

To establish a call, Callgen must know what number to dial as well as what number to associate with a channel. This is accomplished with the following two parameters.

called-number—For originate channels, this specifies the number called. For terminate channels, this specifies the number associated with the specified channel. This is a required parameter.

Example:

```
clash-5300(callgen)# channel 64 called-number 19195551212
```

For ISDN calls, you can specify the type of number and numbering plan identification of the called-party number information element (IE) by giving the oct3 byte in hex value. For example, the Q.931 standard specifies that a national number type and an ISDN numbering plan has an oct3 of 21:

```
clash-5300(callgen)# channel 64 called-number 5551212 oct3 21
```

calling-number—For originate channels only. Specifies the number you are calling from. This information is passed to the called party during call setup for those signaling interfaces that support this capability (such as ISDN PRI, VOIP H323).

For example:

```
clash-5300(callgen)# channel 64 calling-number 19192221212
```

For ISDN calls, in addition to the oct3 byte, you can configure the oct3a byte, which specifies the presentation and screening indicators. For example, if the indicators are "presentation restricted," "user-provided," "verified," and "failed":

```
clash-5300(callgen)# channel 64 calling-number 2221212 oct3 21 oct3a A2
```

Specifying a Redirecting Number

redirect {**redirecting-number** | **original-called-number**} *number* [**reason** *n1*] [**counter** *n2*]

Before a call reaches a Cisco access server, it might have previously been redirected by a 5ESS switch and have redirecting information on the original called number (OCN) or the redirect number (RDN) attached. The OCN is used in H.323 tunneling, and the RDN is used for other services.

This new feature enables including OCN or RDN IE in an ISDN SETUP message of a PRI call as well as the redirecting reason and counter. Note that in IOS, switch-type primary-dms100 only supports OCN, and the rest of the types only support RDN. The parameter is converted to the other form if it is not supported. For example, for primary-5ess, an OCN is converted to an RDN, while primary-dms100 always converts RDN to OCN.

Specifying Setup Timeout

setup-timeout *value* [**seconds**]

Enables forced disconnection of a setup attempt if it takes too long. This ensures the continuity of testing and prevents a channel hanging in the SETUP phase. This configuration applies to originate channels only. The default is 16 seconds. The configurable range is [1, 300] seconds.

Note The default is 300 seconds for Callgen v4.1 and earlier.

Specifying Tear-down Timeout

teardown-timeout *value* [**seconds**]

Enables forced disconnection of a tear-down attempt if it takes too long. This ensures the continuity of testing and prevents a channel hanging in the DISC phase. This configuration applies to originate and terminate channels. The default is 5 seconds. The configurable range is [1, 300] seconds.

Note The default is 300 seconds for Callgen v4.1T and earlier.

Setting the Ringing Duration

ringing-duration *duration time_unit*

Simulates a deferred answering of a call, like when a user picks up after a couple of rings. For PRI or CAS calls, you can specify a ringing duration before answering a call for the terminate configuration.

Note When E1R2 signaling type is used, Callgen terminate channels need to have “ringing-duration 2 seconds” configured. Otherwise, you will likely see the following scenario: the terminate channel is in a HOLD state and the originate channel is in a SETUP state.

Detecting Call Progression Tones (CPtone)

Note Some documents also refer to call progression tones as supervisory tones.

Configuring Call Progression Tone Detection

[**no**] **cptone-detect** [**custom-cptone** *tone-name*] [**dualtone-detect-params** *dualtone-tag*] [**timeout** *timeout-value* [**seconds**]]

cptone-detect is equivalent to the current **cptone-detect** default, that is, the IOS voice-port mode command **cptone** *locale*, which defines the CPtone set. If no **cptone** *locale* is configured under the specific voice port that a Callgen channel is associated with, the default is **cptone us**.

Note This feature is only applied to the originating channel or class.

To configure a custom tone, use **cptone-detect custom-cptone** *tone-name*.

To configure a custom dual tone detect parameter, use **cptone-detect dualtone-detect-params** *dualtone-tag*.

To configure both a custom CPtone and a custom dual tone, use **cptone-detect custom-cptone** *tone-name* **dualtone-detect-params** *dualtone-tag*.

You can configure a timeout value with any of the above options. The default is 15 seconds. Path confirmation or a script does not begin before the configured cptone is detected or the timeout timer expires.

To remove this configuration, use **no cptone-detect**.

Note This feature is not supported on platforms using NextPort DSP, such as as5400, as5400HPX, and as5850.

Note If script and/or path confirmation is configured when cptone detection is enabled, Callgen is likely to pick up the unwanted ringback tone. To work around this problem, configure the script command **wst** *tone* or **ps** *n-seconds*. The specified tone should be high frequency and different from the tones used in the 3-tone path confirmation. If the **ps** option is used, the configured number of seconds depends on how long ringing duration is configured on the terminating gateway. If the default ringing duration is used, **ps 5** should be good. See Defining a Call Script (page 6-4) for more information on the **script** command usages.

Note If a call originates from a PRI interface, only the ringback tone can be detected.

Configuring Custom Call Progression Tones in IOS

You can configure a custom CPtone set using the **custom-cptone** and **dualtone-detect-params** options. *tone-name* is defined by the IOS global configuration **voice class custome-cptone** *tone-name*. *dualtone-tag* is defined by the IOS global configuration **voice class dualtone-detect-params** *dualtone-tag*.

Sample configuration for a predefined CPtone set:

IOS configuration (note, you do not have to configure **cptone us**, since it is the default value):

```
voice-port 0:D
cptone us
```

Callgen configuration:

```
channel 1
cptone-detect
```

Sample configuration for a custom CPtone set:

IOS configuration:

```
voice class custome-cptone uk-custom
dualtone ringback
   frequency 400 450
   cadence 400 200 400 2000

voice class dualtone-detect-params 50
   freq-max-deviation 10
   freq-max-power 6
   freq-min-power 25
   freq-power-twist 15
   freq-max-delay 16
   cadence-min-on-time 50
   cadence-max-on-time 2400
   cadence-variation 8
```

Callgen configuration:

```
channel 1
cptone-detect custom-cptone uk-custom dualtone-detect-params 50
```

Defining a Call Script

Call Script Commands

You can use the **script** command to define a simple script that is executed when the call is established. This gives you complete control over the timing of when DTMF tones are sent and/or expected. The following options can be used in the script. All script instructions and arguments must be separated by spaces.

{	Indicates the beginning of the call script.
sd *dtmf_string*	Sends a string of DTMF tones (legal values: 0123456789ABCD*#). For example, **sd 12345**
rd *dtmf_string*	Expects to receive a string of DTMF tones. For example, **rd 12345**
don *duration*	Specifies the tone duration (in milliseconds) of DTMF tones dialed by following **sd** commands. For example, **don 50**

doff *duration*	Specifies the inter-digit duration (in milliseconds) of DTMF tones dialed by following **sd** commands. For example, **doff 100**
dts *frequency1* [*frequency2* | **0**] *duration*	Sends a dual tone of the specified frequencies for a duration in milliseconds. If *freqency2* is zero, a monotone is sent.
dtr *frequency1* [frequency2 | **0**]	Detects a dual tone of the specified frequencies using the DSP.
idle	Idle until the call duration expires. If the duration is not configured, or the call duration already expires before reaching the idle instruction, the instruction has no effect on the call. If present, this must always be the last instruction.
pms *milliseconds*	Pauses the specified number of milliseconds. For example, **pms 100**
ps *seconds*	Pauses the specified number of seconds. For example, **ps 10**
vad	Pauses until voice activity detected.
flsh	Generates a hookflash.
lc *n_times*	Loops on the previous instruction *n* times. For example, **lc 1** executes the previous instruction one more time.
ls [*n_times*]	Loops on the script *n* times. If the number of times is not specified, it loops on script until the call duration expires (call will not be disconnect until script completes). If present, this instruction must always be the last instruction. For example, **ls 5** executes the script five times and then disconnects. If path confirmation is also desired on the call, then **ls** needs to be specified with a loop count.
ms	Sets next instruction as a new entry point of script. The **ls** instruction uses this instruction as the return point when looping.
pl *audio-file-tag*	Plays an audio file onto the channel. The file tag must have been defined in global mode. This instruction ends when the playing finishes. For example, **pl 1**
rc *audio-file-tag duration*	Records the incoming voice on the channel to an audio file. The file tag must have been defined in global mode. This instruction ends when the *duration* in seconds of voice has been recorded. For example, **rc 2 10**. **Note** Recording audio into a file in a router's flash is not supported.
st *tone-freq duration*	Sends tone of specified *tone-freq* for *duration* in milliseconds. For example, **st 1000 50**
rt *tone-freq*	Detects the tone of the specified *tone-freq*.
wst *tone-freq*	Wait for a sync tone. The difference between **wst** and **rt** is that **wst** ignores the unexpected incoming tones, and **rt** treats unexpected tones as an error.
dat *n-milliseconds*	Detects arbitrary tones for the specified period of time.
rstt	Resets the script stopwatch.
stpt	Stops the script stopwatch and calculates the elapsed time after resetting the script stopwatch. The **rstt** command should precede this instruction. For example, **rstt vad stpt** determines the elapsed time to detect voice activity.
}	Indicates the end of the call script.

Examples

In the following, when the call is established, the script first sends the DTMF sequence, 45678, and then waits to receive the DTMF sequence, 1234. Once these tones are received, the script pauses for

5 seconds and then sends the DTMF sequence, 123. This instruction is then repeated two more times. Thus, the DTMF sequence actually sent is 123123123. If path confirmation is configured on the channel, it is done; otherwise, the call is disconnected.

```
script {sd 45678 rd 1234 ps 5 sd 123 lc 2}
```

This script is the same as above, expect that it continues to repeat until the call duration expires. If the call duration expires in the middle of a script, the current script iteration completes and the call is disconnected. Additionally, you will not be able to configure path confirmation on this channel.

```
script {sd 45678 rd 1234 ps 5 sd 123 lc 2 ls}
```

This script operates the same as the previous script on the first pass; however, instead of repeating from the start of the script, the **ls** instruction causes control to return to the **rd 1234** instruction.

```
script {sd 45678 ms rd 1234 ps 5 sd 123 lc 2 ls}
```

The ping form of path confirmation can also be specified as a script (in fact, this is how Callgen implements it). For example, the following two path-confirmation configurations are equivalent for an originate channel:

```
path-confirmation type ping string 1234
script {rd 1234 sd 1234 ls}
```

The following two path-confirmation configurations are equivalent for a terminate channel:

```
path-confirmation type ping string 1234
script {sd 1234 rd 1234 ls}
```

Scripts also help to enable generating RTP traffic to a specific VoIP gateway by repeatedly wrapping an audio file into RTP packets and sending them out. For example (you should first copy an audio file to the Flash):

IOS config:

```
dial-peer voice 1001 voip
 destination-pattern 525....
 codec g711ulaw
 no vad
 session target ipv4:10.0.1.5
```

Callgen config:

```
global type voice
 audio-file 1 flash:reference.au

channel 1 type voice mode originate
 calling-number 5270001
 called-number 5250001
 duration 3 minutes
 inter-call-delay 5 seconds
 script {pl 1 ps 1 ls}
 interface voip:1001
```

wst Script Command

wst is especially useful when you want to synchronize both CG_originate and CG_terminate before taking the next action, for example, path confirmation. It is also used to avoid Callgen mistaking the call progression tones as the expected tones/path confirmation tones sent by the far end. Before this command is implemented, you should configure **script {ps 5}**, which allows Callgen to wait for 5 seconds to bypass the call progression tones before the path confirmation starts. By default, **wst** waits for 15 seconds for a tone. You can change this time value with the command **path-confirmation time-out** *n*. When **wst** detects a tone, the wait timer is reset.

For example:

```
channel 1 type voice mode originate
  duration 30 seconds
  called-number 8886666
  path-confirmation type 3-tone-slope
  script { wst 2230 ps 1 }
  interface port:2/0:0

channel 2 type voice mode terminate
  called-number 8886666
  path-confirmation type 3-tone-slope
  script { st 2230 100 ps 2 }
  interface port:2/1:0
```

dat Script Command

dat *n-milliseconds*

Records the arbitrary tones Callgen detected within the specified period of time. The internal tone buffer can hold 20 entries at most. The buffer is wrapped around from the beginning of the buffer if more than 20 tones have been detected. Use **show channel** *n* to see which tones are being detected. The advantage of using this command is that if an expected tone (with respect to your test case) is received, Callgen does not consider it an error and does not disconnect the call because of the unexpected tone. It is good to use this feature rather than **rt** if you know that the network is busy and you are expecting the tone to be garbled.

Note If the far end sends a long duration tone, Callgen thinks that multiple tones are being sent from the far end. To avoid that, if you are sending a high frequency tone (> 1000Hz), a tone duration of 100 milliseconds is enough. If you are sending a low frequency tone (< 1000Hz), a tone of 100 to 150 milliseconds is good. Basically, the lower the frequency, the longer the time is needed to detect the tone, for example, for 400Hz, you need about 150 milliseconds.

For example:

```
Channel 1 mode originate
  Called-number 123
  Script { dat 5000 }
  Interface port:1/0:0

Channel 101 mode terminate
   Called-number 123
  Script { st 500 150 pms 50 st 1005 100 pms 500 st 2005 150 idle }
  Interface port:1/1:0

cg-3660-1(callgen)#sh ch 1

Channel 1 Call Statistics
  elapsed channel run time: 00w0d00:00:07.756
  channel state: INACTIVE
  setup attempts: 1
  accepts: 1
  confirms: 0
  setup-fails: 0
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  passed-calls: 1
```

```
failed-calls: 0
setup rate: 0 calls per millisecond
accept rate: 0 calls per millisecond
setup time: min: 1956ms, max: 1956ms, avg: 1956ms
hold time: min: 5000ms, max: 5000ms, avg: 5000ms
disconnect time: min: 800ms, max: 800ms, avg: 800ms
idle time: min: 0ms, max: 0ms, avg: 0ms
inter-digit-delay: min: 0ms, max: 0ms, avg: 0ms
last disconnect cause: 16 normal call clearing (16)
last abnormal disconnect cause: 0
script completed: YES
Detected arb tones: 500 1000 2000 2012
Script Stopwatch: N/A
```

Note 4-tone is detected instead of 3-tone. The reason the last two tones are detected instead of one tone is because of a long tone duration (for a high frequency tone).

Specifying a Timeout Value for Script Commands

[**no**] **script time-out** *n* [{[**seconds**] | **milliseconds**}]

Configures a timeout value for the **rd**, **rt**, and **wst** script commands. It can be configured on both a channel and a class. The default is 15 seconds, where seconds is the default timeout unit.

Note Use the command **path-confirmation timeout** *n* [**seconds**] for version 4.1T or earlier**.**

Executing a Call Script On Demand

channel *startChannel* [**-** *endChannel*] **script** {...} **exec**

Configures and executes a Callgen script on demand. To do this, the following conditions much be satisfied:

- An infinite loop-script instruction cannot be configured in the current script if the channel is already configured with a call script.
- A path-confirmation type cannot be configured for the channel.
- A channel must be in the **hold** state at the time this command is entered.

You can reconfigure a script on demand many times if each script is configured with the **idle** instruction. For example:

```
cg-5300-8(callgen)#channel 1 script {ps 10 sd 123 idle} exec
```

Note The output of the **show channel** *num* command shows the completion status of a script, which could be a static call script or a call script on demand. If a script is removed or no script is configured, the script completion indication is not shown.

Path Confirmation

Path confirmation is the term Callgen uses to indicate any form of communication passed inband (in the voice or bearer channel) to verify end-to-end voice communication of telephony calls. Path confirmation also applies to H.323 calls by using the DTMF-relay technique. In that case, along the voice path, DTMF tones are inband signals when passing a telephony interface and outband H.245 messages when passing a network interface.

Note Path confirmation cannot be used when originating or terminating calls over a VoFR interface.

Currently, path confirmation only supports ping-type calls with an H.323 originate or terminate endpoint.

path-confirmation type ping [{**string** *dtmf_string*} | **called-number**]

string *dtmf_string*—Numeric string to ping back and forth. For example:

```
path-confirmation type ping string 1234
```

called-number—Pings the channel's called number back and forth.

If no arguments are specified, a system-selected DTMF tone is pinged back and forth. The system-selected DTMF sequence is 01B.

Ping-type path confirmation uses the passing and detecting of numeric DTMF tones between the originate and terminate channels to verify the channel. Here is how it works:

1. An originate channel with path confirmation configured performs a call setup to a terminate channel that also has an identical path confirmation configuration.
2. After answering the call, the terminate channel waits a configurable delay (**cut-through-time**), and then sends out a sequence of numeric DTMF tones. The default value for **cut-through-time** is 800 milliseconds.
3. After the correct reception of this tone sequence, the originate channel pings the same sequence back to the terminate channel.
4. After receiving the correct tone sequence back, the terminate channel again pings the sequence back to the originate channel.
5. Repeats steps 3 and 4 for the duration of the call.

Path confirmation for a call is successful if all the DTMF tones are received in the correct order. If the configured duration of a call is 0, at least one path-confirmation sequence is attempted from both endpoints after the call is established.

The options for the **path-confirmation** parameter are described below. These parameters only take effect when **path-confirmation type ping** or **path-confirmation type 3-tone-slope** is specified.

Note To return a path-confirmation parameter to its default value, prefix the configuration statement with **no**. For example: **no path-confirmation type**.

Specifying 3-Tone Slope

path-confirmation type 3-tone-slope [*freq-1 freq-2 freq-3*] [**duration** *duration* [**milliseconds**] [**full-duplex**] [**dsp-det**]]

Instead of dialing DTMF tones, 3-tone-slope path confirmation sends three continuity tones back and forth. You can configure any three tones in an ascending order between 300 Hz to 3400 Hz of at least 100 Hz apart. The default tones are 404 Hz, 1004 Hz, and 2804 Hz. You can also specify the duration of each tone (the default is 400 milliseconds).

By default, the path confirmation works at half-duplex fashion, repeatedly sending and receiving tones alternatively. To repeatedly send and receive tones simultaneously, specify **full-duplex**. This helps to verify whether echo or crosstalk degrades the voice path during a conversation. The parameters **digit-on-time** and **digit-off-time** have no effect on this type of path confirmation.

If the **dsp-det** option is specified, Callgen utilizes the DSP to detect the incoming tones. If this option is not specified (default), Callgen utilizes the CPU resources to detect the incoming tones.

Note The **dsp-det** feature is not supported on platforms using NextPort DSP, such as as5400, as5400HPX, and as5850.

An example of two-stage dialing using 3-tone slope path-confirmation is:

Originate:

```
called-number 408
script {ps 2 sd 5258414}
path-confirmation type 3-tone-slope
```

Terminate:

```
called-number 5258414
path-confirmation type 3-tone-slope
```

path-confirmation type 3-tone-slop [**play** *audio-file-tag*]

Supports Skinny images in Callgen release 3.2 and release 4.0T or newer. You can configure this path confirmation for the VoIP (H.323 and SIP) or EFXO (emulated IP phone) interfaces. It also works for telephony interfaces, but it’s not recommended. Use all the default parameter settings if you use the provided audio file *3tone_s.au* (available at /auto/autons/vts/audio/). This feature demands CPU resources to play out the tone file and detect the incoming tones.

Note If you configure 3-tone path confirmation in a topology with one side as telephony and the other side as VOIP/Skinny IP phone, you need to make sure that the tone file used on VOIP/Skinny IP phone channels is matched with the tones configured on the telephony channels.

Specifying Continuity Test (COT) Path Confirmation

path-confirmation type cot {**single** | **continuous**}

COT path confirmation only works for telephony interfaces. If single mode is specified, COT is only performed once. If continuous mode is specified, the test is run repeatedly until the call duration expires.

Configuring **path-confirmation digit-on-time/digit-off-time** has no effect on COT path confirmation.

Unlike other path confirmation types, for COT path confirmation, it is recommended that you specify a cut-through time as accurate as possible to guarantee the end-to-end synchronization of the originate and terminate channels. Other path confirmation types are asynchronous, so the requirement of cut-through time is loose: larger than the actual cut-through time and less than timeout.

The following is an example for PRI back-to-back connection:

```
channel 2 type voice mode originate
  duration 20 seconds
  inter-call-delay 2 seconds
  called-number 8983212
  path-confirmation type cot single
  path-confirmation time-out 5 seconds
  interface port:1:D

channel 2 type voice mode terminate
  called-number 8983212
  path-confirmation type cot single
  path-confirmation cut-through-time 0 seconds
  path-confirmation time-out 5 seconds
  interface port:1:D
```

Specifying Error Tolerance

path-confirmation error-tolerance *percentage*

Specifies an acceptable level of errors. As long as the average error percentage remains within this tolerance, the call stays up and is considered confirmed upon completion. If at any time the average error percentage exceeds the error tolerance, the call is immediately disconnected.

To tolerate all errors, specify **100**. The default is 0. For example, the following specifies an average error percentage of 20 percent:

```
path-confirmation error-tolerance 20
```

Specifying Error Threshold

path-confirmation error-threshold {**end-of-call** | *n_digits*}

Specifies how many digits to receive before adhering to the error tolerance specified with the **error-tolerance** command. This prevents calls that might encounter errors at the beginning of the call from being prematurely disconnected.

The following indicates that 15 digits should be received before Callgen starts disposing of calls based on the error tolerance. For instance, suppose error tolerance is set at 20 percent, and the first two of three digits received during path confirmation are in error (such as wrong digit or timeout). This is an average error percentage of 66 percent, which is greater than our tolerance. However, because an error threshold of 15 is specified, the call is not disconnected.

```
path-confirmation error-threshold 15
```

The following indicates that Callgen calculates the average error percentage at the end of the call. It is not disposed in the middle of its duration, although the instant error percentage might be greater than the specified error tolerance.

```
path-confirmation error-threshold end-of-call
```

Specifying Time Terminate Channel Waits to Send Path Confirmation

path-confirmation cut-through-time *n* [**milliseconds** | **seconds**]

Specifies how long a terminate channel waits before sending path confirmation. It is required because, in many cases, a terminate channel answers a call and starts sending DTMF tones before the originate channel's voice band is available, causing the initial DTMF tones to be missed.

For **ping** path confirmation, this parameter specifies the minimum delay the channel waits before sending the first digit in the configured DTMF sequence. For **script** path confirmation, it is the minimum delay the channel waits before executing the script. The default is 800 milliseconds. Configure this parameter based on the network topology.

Examples:

```
path-confirmation cut-through-time 1200 milliseconds
path-confirmation cut-through-time 3 seconds
```

Specifying Length of Time to Wait for a Tone

path-confirmation time-out *n* [{[**seconds**]| **milliseconds**}] [**subseq** *m* [{[**seconds**]| **milliseconds**}]]

Value *n* specifies how long a channel waits for the first tone in a DTMF sequence in a ping path-confirmation case or monotone sequence in a 3-tone (half-duplex) path-confirmation case. The 3-tone path-confirmation includes 3-tone, 3-tone play, and 3-tone dsp-det. The default is 15 seconds.

Value *m* specifies how long a channel waits for the subsequence tone in a DTMF sequence in a ping path-confirmation case or monotone sequence in a 3-tone (half-duplex) path-confirmation case. The 3-tone path-confirmation includes 3-tone, 3-tone play, and 3-tone dsp-det. The default is 6 times the tone on duration.

For example:

```
path-confirmation time-out 5 seconds subseq 3 seconds
```

Note For version 4.1T or earlier, this specifies how long a channel waits for each tone in a DTMF sequence in a ping path-confirmation case or monotone sequence in a 3-tone path-confirmation case.

Note For “path-confirmation type cot {single | continuous},” “path-confirmation type 3-tone full-duplex,” and “path-confirmation type resilient,” only the first value of “path-confirmation time-out” is applied, that is, **subseq** timeout is not being used.

Specifying Delay Between Tones

path-confirmation digit-off-time *n* [**milliseconds**]

Specifies the delay between sending each DTMF tone in a sequence. The default is 150 milliseconds. For example:

```
path-confirmation digit-off-time 300 milliseconds
```

Specifying Tone Duration

path-confirmation digit-on-time *n* [**milliseconds**]

Specifies the duration of each DTMF tone in a sequence. The default is 50 milliseconds. For example:

```
path-confirmation digit-on-time 150 milliseconds
```

Specifying Tone Tolerance

path-confirmation tone-tolerance *n* [**hertz**]

Specifies the tolerance, plus or minus, of a receiving tone to be acknowledged as an expected tone when performing 3-tone-slope path confirmation, without the **dsp-det** option. The default is 100 Hz. For instance, if 2804 Hz is expected, any receiving tone ranging from 2705 to 2903 Hz is regarded as a correct reception. For example:

```
path-confirmation tone-tolerance 200 hertz
```

Setting the Delay After a Tone

path-confirmation post-sending-delay *n* [**milliseconds**]

Sets the guard time to the next script command after sending a string of DTMF tones. Different interfaces require different delays after sending DTMF tones to ensure the last digit was sent and to avoid receiving echo in the tear-down phase. The default is 600 milliseconds. For example:

```
path-confirmation post-sending-delay 300 milliseconds
```

Testing Transmission Quality of UUTs

path-confirmation type resilient *freq* **vaf** *VAF%* **tone-on-time** *duration* **max-hit** *hit-duration*

Tests the transmission quality of units under test (UUT). A channel with this configuration sends a relatively long duration of a tone (4 to 10 seconds) and simultaneously expects to receive a different tone (>200 Hz difference) from the other end. The **vaf** (voice activity frequency) parameter specifies the duty-cycle of the tone (percentage of tone-on in a period). It is useful for generating different voice traffic loads on UUTs if **vad** (default) has been configured in the corresponding VoIP dial peers. The **max-hit** parameter defines the maximum tolerable duration of a hit, which is an interrupted silence among voice activity, and is called clip if happening at the beginning or ending edge. The path confirmation fails if a hit or a clip has been detected with a duration greater than **max-hit** (milliseconds).

Resilient path confirmation requires the two sides having tone frequencies >200 Hz apart and exactly the same **vaf** and **tone-on-time**. A bonus statistic of resilient path confirmation is that counts of hits and clips during a call are reported in channel statistics when using the **show channel** command (see Displaying Channel Information (page 4-12)). It should always be less than **max-hit**.You can specify the threshold-hit duration by **voice-quality hit-threshold** *n* **milliseconds**. The default is 200 milliseconds.

Things to Note When Using Path Confirmation

For the following information, consider **ping** path confirmation implemented and behaving as its script equivalent.

- A call is disconnected in the middle of a script only if the remote end disconnects or the script times out waiting for a tone. The configured call duration is ignored, unless the script ends with the **ls** instruction (loops until duration expires).
- When sending DTMF tones, the **digit-off-time** is also applied before the first digit is sent. Thus, if a script includes **sd 1234**, the actual cut-through delay is configured **cut-through-delay** + configured **digit-off-time**.
- When originating or terminating calls over a VoIP interface with path confirmation, refer to the following configuration:

Originate Router (10.0.1.5):

Dial-peer configuration:

```
dial-peer voice 1001 voip
 destination-pattern 525....
 dtmf-relay h245-alphanumeric h245-signal
 session target ipv4:10.0.3.6
```

Callgen configuration:

```
channel 1 type voice mode originate
  duration 3 minutes
  inter-call-delay 3 seconds
  calling-number 5270001
  called-number 5250001
  path-confirmation type ping
  interface voip:1001
```

Terminate Router (10.0.3.6):

Dial-peer configuration:

```
dial-peer voice 2001 voip
 destination-pattern 527....
 dtmf-relay h245-alphanumeric h245-signal
 session target ipv4:10.0.1.5
```

Callgen configuration:

```
channel 1 type voice mode terminate
  called-number 5250001
  path-confirmation type ping
  path-confirmation cut-through-time 2 seconds
  interface voip:2001
```

FXO Loop-start Emulation

emulate fxo-loop-start {**on** | **off**}

Terminates calls on CAS interfaces, where fxo-loop-start emulation is desired. This is a special requirement for when the router's CAS interface is connected to a channel bank, and the Callgen terminate router needs to accept the incoming setup indication similar to an FXO interface. The **on** option turns on fxo-loop-start emulation. The default is off. This command is configurable on terminate channels only.

Q.SIG Messaging

Callgen provides ISDN Q.SIG messaging to tunnel Q.SIG messages between endpoints (originate and terminate channels). The Q.SIG messages support is based on Symphony's raw message API to pass the messages along with Q.931 signaling messages. Currently, we only support the Q.SIG messaging in CALL SETUP. The Q.SIG message is manually entered in the originate channel from the CLI with the format of two hexdecimal digits separated by blank spaces. Three call types are available: basic, signal-only, and dummy-facility. The terminate channel receives the Q.SIG messages and stores in-channel statistics information.

The syntax is:

[**no**] **q931 setup** *hexdecimal-string*
[**no**] **qsig-calltype** {**basic-call** | **sigonly** | **dummy-facility**}
[**no**] **qsig-message** {**on** | **off**}

For example, to make a signal-only call with called number 5678, you first need to configure the ISDN switch type to **primary-qsig** and the appropriate pri-group for both the originate and terminate Callgen router.

Originate channel configuration:

```
channel 1 type voice mode originate
...
q931 setup "A1 04 02 A8 80 18 01 AC 6C 06 21 83 70 05 A1 35 36 37 38"
qsig-calltype sigonly
qsig-message on
```

Terminate channel configuration:

```
channel 2 type voice mode terminate
...
qsig-calltype sigonly
qsig-message on
```

To display the raw message received in the terminate channel:

```
(callgen) show chan 2
Channel 2 Call Statistics
   ...
   last abnormal disconnect cause: 0
   RawMsg: 04 02 A8 80 18 01 AC 6C 02 21 83 70 05 A1 35 36 37 38
```

To make a basic call with called number 5678:

Originate channel configuration:

```
channel 1 type voice mode originate
...
q931 setup "A1 04 03 80 90 A3 18 03 A9 83 81 70 05 A1 35 36 37 38"
qsig-calltype basic-call
qsig-message  on
```

Terminate channel configuration:

```
channel 2 type voice mode terminate
...
qsig-calltype basic-call
qsig-message  on
```

To display the raw message received in the terminate channel:

```
(callgen) show chan 2
Channel 2 Call Statistics
   ...
   ...
   last abnormal disconnect cause: 0
   RawMsg: A1 04 03 80 90 A3 18 03 A9 83 81 70 05 A1 35 36 37 38
```

ISDN Overlap Dialing

There are several ISDN switch types that support overlap-receiving on IOS routers. In this mode, the interface waits for additional call-control information, such as partial DNIS digits. To test such a feature, Callgen enables overlap-dialing to send a complete called number, digit by digit, via a series of INFORMATION messages. The syntax is:

[**no**] **isdn-overlap-dialing** {**off** | **on** [**no-complete**] [**init-digits** *n*]}

If you use the **no-complete** option, “no complete” is enclosed in the INFORMATION message that carries the last digit to indicate DNIS completion. Usually the receiving gateway or PBX relies on the ISDN T302 timer to determine the end of DNIS collection. The default is to enclose COMPLETE along with the last digit.

If **isdn-overlap-dialing** is configured in a channel with an ISDN interface, the Q.931 SETUP message includes no digits or the first few digits of the configured called number. The remaining digits are then sent in the following INFORMATION messages, one digit at a time. To control the timing of sending overlap-dialing digits, use the commands **path-confirmation post-dialing** *pdd*, **path-confirmation digit-on** *don*, and **path-confirmation digit-off** *doff*.

Note Although these commands use the prefix "path-confirmation," they have nothing to do with path confirmation; it is just to borrow existing commands for convenience.

The first overlap-dialed digits (the second digit in the called number) is sent *pdd+don+doff* milliseconds later than the SETUP message (which carries the first digit). Later, the overlap-dialed digits are sent in order with an interval of *don+doff* milliseconds.

You can use **init-digits** *n* to specify the number of the called-number digits, from 0 to 32, to send in the SETUP message. The default is 0.

Making Calls Using SIP

The Cisco Session Initiation Protocol (SIP) functionality, introduced in Cisco IOS release 12.1(1)T and enhanced in Cisco IOS release 12.1(3)T, enables Cisco access platforms to signal the setup of voice and multimedia calls over IP networks. The supported platforms are Cisco AS5300 access servers, and Cisco 2600 and 3600 series routers.

To make a SIP call using Callgen, configure the dial peers to initiate the SIP session instead of the H323 session. For example:

```
dial-peer voice number voip
session protocol sipv2
```

The call received by the next gateway router can be transferred to either a POTS or VoIP dial peer. To configure a SIP call transfer for a POTS dial peer, enter the following:

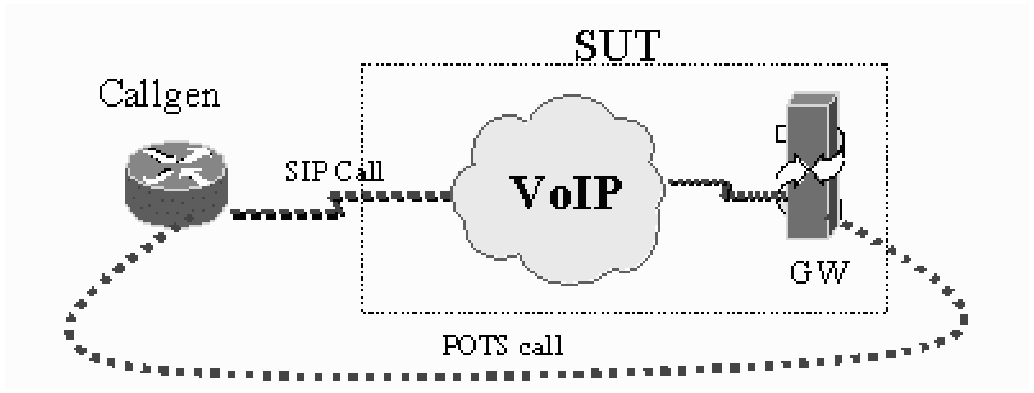
```
dial-peer voice number pots
application session
destination-pattern pattern
port slot/port
```

To configure a SIP call transfer for a VoIP dial peer, enter the following:

```
dial-peer voice number voip
application session
destination-pattern pattern
session target ipv4:x.x.x.x
```

For detailed information, see "Enhancements to the Session Initiation Protocol for VoIP on Cisco Access Platforms" at http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t3/dtsipgv2.htm

In the following example, Callgen is making a SIP call over the IP network. The terminating gateway receives the call and transfers it to a POTS dial peer back to the Callgen router.



Callgen configuration:

```
channel 1 type voice mode originate
duration 1 min
inter-call-delay 3 sec
called-number 5170000
interface voip:500

channel 2 type voice mode treminate
called-number 5170000
interface port:1/1/0
```

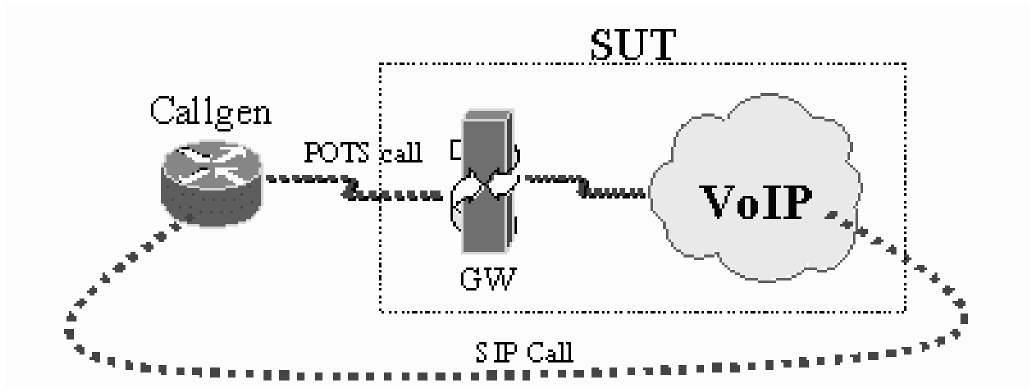
Callgen IOS configuration:

```
dial-peer voice 500 voip
destination-pattern 517....
session protocol sipv2
session target ipv4:10.0.2.46
codec g711ulaw
no vad
```

Gateway IOS configuration:

```
dial-peer voice 600 pots
application session
destination-pattern 517....
port 2/0/0
```

In the following, the Callgen router is making a POTS call to a gateway, which makes a SIP call over the IP network. The call is eventually terminated in the Callgen router.



Callgen configuration:

```
channel 3 type voice mode originate
  duration 1 minutes
  inter-call-delay 3 seconds
  calling-number 5551212
  called-number 5170000
  interface port:1/0/0

channel 4 type voice mode terminate
  called-number 5170000
  interface voip:600
```

Callgen IOS configuration:

```
dial-peer voice 10005 pots
 application CALLGEN_VOICE_CTM
 group 1
 direct-inward-dial
 port 1/0/0
!
dial-peer voice 600 voip
 application callgen_voice_ctm
 incoming called-number 5170000
 session protocol sipv2
 codec g711ulaw
 no vad
```

Gateway IOS configuration:

```
dial-peer voice 500 voip
 destination-pattern 517....
 session protocol sipv2
 session target ipv4:10.0.2.39
 codec g711ulaw
 no vad
```

Measuring End-to-End Jitter

voice-quality type end-to-end-jitter [**simplex** {**send-only** | **receive-only**}] [**period** *n* [**milliseconds**]]

It is not unusual for consecutive voice packets to suffer from jitter effect after they have been delivered across networks. Though the receiving end may remove jitter by various buffering schemes, a typical network congestion could easily lead to audible jitter effect.

Callgen measures jitter by sending test tones periodically on one side and taking the time difference of every two consecutive arriving tones on the other side. To get the jitter measurement, we subtract the time difference from the testing period. Note that we are measuring voice jitter, not packet jitter, so there might not be voice jitter but packet jitter exists. However, this is an expected behavior as long as jitter buffering works fine.

Ideally, test tones arrive precisely in the period that the sending side sends (a jitter of zero). If a test tone arrives early, the time difference to the previous tone is shorter than the testing period, so jitter is negative, which is called lead jitter. If a test tone arrives late, jitter is positive, which is called lag jitter.

Two modes do the measurement of end-to-end jitter: duplex mode and simplex mode. In duplex mode, both the originate and terminate channels send and receive tones and do the measurement. In simplex mode, one channel is configured as **send-only** and the other as **receive-only**. The send-only channel sends the tones and does not do measurements. The receive-only channel receives the tones and does the measurements.

In this example configuration of jitter measurement, the originate channel and terminate channel do not need to be on the same Callgen router when running a jitter measurement:

```
channel 1 type voice mode originate
  duration 3 minutes
  called-number 5250001
  voice-quality type end-to-end-jitter
  interface port:0:0

channel 2 type voice mode terminate
  called-number 5250001
  voice-quality type end-to-end-jitter
  interface port:1:0
```

If you use the **show channel** *num* command to view the test results, you would see something similar to the following:

```
current end-to-end lead jitter: 1%, min: 10ms, max: 20ms, avg: 12ms
current end-to-end lag jitter: 1%, min: 4ms, max: 20ms, avg: 10ms
```

The percentage, 1% in this example, indicates how frequently jitter was detected. Min, max, and avg indicate the range of the 1% measurements.

The following output of the **show chan voice-quality** command shows the average jitter, which is calculated by multiplying the above percentage with the amount specified by "avg." The plus or minus (-/+) indicates whether the jitter is lead or lag.

```
ch-1-vo-o, state: HOLD, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.1/+0.1,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: 0, avg-rtt: 0, echo: na/0/0
  avg-pesq: 0.000
```

Currently, jitter measurement works better with digital interfaces, but suffers interference from echos with 2-wire analog (FXS/FXO) interfaces, so we recommend using the following configuration to train the echo canceller first:

script {**sd 1234567890 ls 20**}

Measuring End-to-End Delay

voice-quality type end-to-end-delay to-channel *n* [**simplex** {**send-only** | **receive-only**}] [**correction** *duration* [**milliseconds**]]

To measure the latency of voice from one end to the other end, the originate and terminate channels must be on the same Callgen router. The **to-channel** parameter specifies the partner channel to do the measurement in the opposite direction. The partner channel of an originate channel must be the unique terminate channel answering its calls. On the other hand, the partner channel of a terminate channel must be the unique originate channel calling it. It is crucial to give a correct partner channel to make a legal measurement.

Note You need to be careful if you are used to assigning the whole trunk T1/E1 interface with a single called number. End-to-end delay measurement requires each channel has a unique called number on a Callgen router.

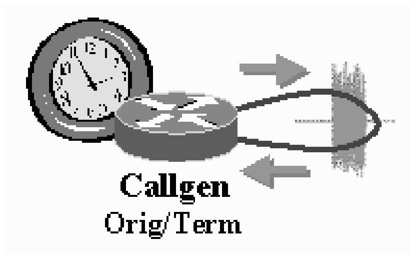
Two modes do the measurement of end-to-end jitter: duplex mode and simplex mode. In duplex mode, both the originate and terminate channels send and receive tones and do the measurement. In simplex mode, one channel is configured as **send-only** and the other as **receive-only**. The send-only channel sends the tones and does not do measurements. The receive-only channel receives the tones and does the measurements.

Since the measurement includes the processing delay on the local router, usually 5 to 15 milliseconds, you might want to subtract it from the measurements by specifying the **correction** parameter. To obtain an accurate correction duration, you could connect the two voice ports back-to-back and do end-to-end delay measurement with a correction of 0 (the default). The reported average delay after a one-hour test should be a good estimation of the processing delay. If you are using the network-side image for PRI calls, you have to boot the router with a regular image and configure one voice port as the network side to enable back-to-back connection to do an estimation of processing delay. This is done by going into the serial:23 (T1) or serial:15 (E1) and using the following command sequence: **isdn net, shut, no shut**.

To recover the misalignment of IOS clock and DSP clock caused by different timing granularities, Callgen employs a phase-locking algorithm to ensure the measurements to a +(-1) millisecond accuracy. However, if the latency happens to fluctuate intensely, for example UUTs with larger jitter or using low-bit-rate codec, Callgen is not be able to lock the phase and a successful measurement cannot be made. In that case, you have to explicitly disable the phase-locking by configuring **voice-quality phase-locking off** in both channels doing end-to-end delay measurement. This results in a looser range of accuracy. Nevertheless, since the errors are symmetric and equiprobable to the actual value at positive and negative directions, the average value after a long run should be a good enough measurement.

The measurement result is included in channel statistics (using the **show channel** *num* command). The maximum, minimum, and average delay is reported together with the number of measurement, which somewhat indicates how much you could trust the average value.

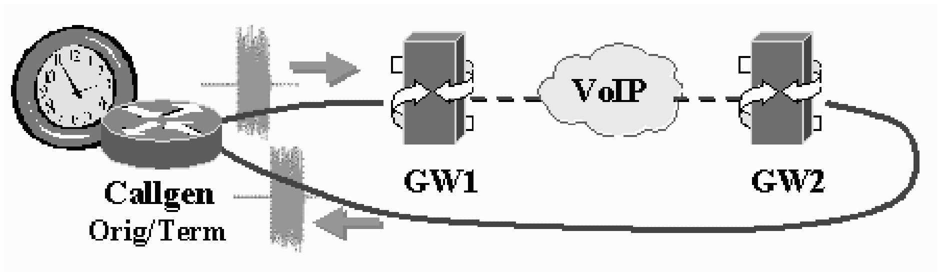
The following is an example setup and CAS configuration to measure processing delay (T1/0 connects to T1/1 back-to-back).



```
channel 1 type voice mode originate
  duration 1 hour
  called-number 5250001
  voice-quality type end-to-end-delay to-channel 2
  interface port:0:0

channel 2 type voice mode terminate
  called-number 5250001
  voice-quality type end-to-end-delay to-channel 1
  interface port:1:0
```

Say after one hour, the reported average delay is 11 milliseconds in channel 1, and 9 milliseconds in channel 2. Now you can connect T1/0 to the system under test (SUT) and T1/1 back from SUT and use the following setup and configuration to measure the end-to-end delay.



```
channel 1 type voice mode originate
  duration 1 hour
  called-number 5250001
  voice-quality type end-to-end-delay to-channel 2 correction 11
  interface port:0:0

channel 2 type voice mode terminate
  called-number 5250001
  voice-quality type end-to-end-delay to-channel 1 correction 9
  interface port:1:0
```

Then after another hour, say the reported average delay is 75 milliseconds in channel 1 and 71 milliseconds in channel 2, we could say in the past hour the end-to-end delay from originate to terminate is 71 milliseconds, and from terminate to originate is 75 milliseconds in average.

Measuring Round-Trip Time

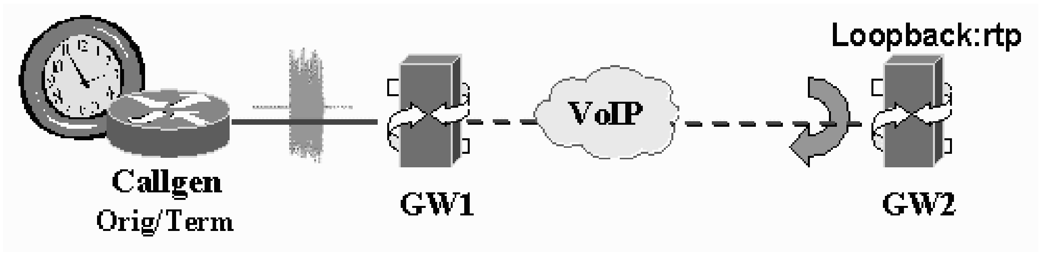
voice-quality type round-trip-time

Measuring round-trip time (RTT) is a bonus feature associated with end-to-end delay measurement. This is because the end-to-end delay measurement runs in a full-duplex fashion, so Callgen will not have a problem to listen to its looped-back test tone even when the tone is still under sending.

To make a loopback end to terminate a call and echo voice back to the originate router, you could configure a dial peer on the remote end similar to the following example:

```
dial-peer voice 8100 voip
 destination-pattern 525....
 session target loopback:rtp
```

With this dial peer, all calls to 525.... are answered by the remote router and all following voice gets echoed (Note that you need to load the router with an image including fix to CSCdp02879. Or, simply use a Callgen image after release 2.1.) A typical setup is illustrated below.



The following is an example of a typical Callgen configuration on the originate side (you do not need terminate channels to terminate calls in a loopback setup).

```
channel 1 type voice mode originate
  duration 30 minutes
  inter-call-delay 5 seconds
  calling-number 5270001
  called-number 5250001
  voice-quality type round-trip-time
  interface port:0:D
```

Note You do not need to configure Callgen to terminate VoIP calls if you already have an RTP loopback. However, configuring a Callgen terminate channel is probably easier than configuring an RTP loopback if you do have a Callgen image there. In fact, if a Callgen image is running on the terminate router, RTP loopback has no effect (it is ignored by Callgen).

There are some advantages of measuring RTT instead of end-to-end delay. First, a loopback saves the hassles of configuring different called numbers and specifying partner channels, which are required for the end-to-end delay measurement. Second, you do not need to change physical wiring to connect a call back to the originate router; hence, it saves one interface port. However, the price is that RTT does not directly account for an end-to-end quality. Also, QoS of voice traffic might not guarantee the latency of return trip so that RTT becomes less informative in helping to perceive an end-to-end quality of the system under test.

To view the current statistic of measurement, use the **show channel** *num* command to list the minimum, maximum, and average of RTT measured so far on that channel.

Measuring Echo

voice-quality type echo [**period** *n* [**milliseconds**]]

In most cases, echo is caused by an impedance mismatch in the hybrid when doing a 4-wire to 2-wire conversion. It could happen at a near-end or a far-end PBX or Telco switch. Usually, the near-end echo is tolerable and, in fact, is sometimes preferable since it makes the voice richer, but far-end echo could be annoying to a conversation. Two key metrics that echo testing checks are delay, which is the round-trip time of the spoken voice, and energy loss, which is the attenuation of the voice leaking back.

Callgen does echo measurement by having one side be silent after a call connection while the other side keeps sending random tones. The incoming voice activity is monitored concurrently. Any incoming voice spurt is regarded as an echo of the most recent test tone sent, since the remote end remains silent. Echo delay is calculated as the time difference between sending the test tone and when the echo is received. Energy loss is calculated by measuring the attenuation of the peak instantaneous energy of the sent and received tones.

The following is an example of measuring echos perceived by the originate side. To measure echos perceived by the terminate side, move **voice-quality type echo** from channel 1 to channel 2. The originate channel and terminate channel do not need to be on the same Callgen router when running a echo measurement.

```
channel 1 type voice mode originate
  duration 3 minutes
  called-number 5250001
  voice-quality type echo
  interface port:0:0

channel 2 type voice mode terminate
  called-number 5250001
  interface port:1:0
```

To avoid interference by ring-back tones or incoming DTMF tones for analog interfaces, include the following with the **voice-quality** statement:

script {**ps 6**}

If you use the **show channel** *num* command to view the test results, you would see something similar to the following. The threshold of the short and audible echo is 50 milliseconds.

```
short echo: min: 3ms/-22dB, max: 22ms/-36dB, avg: 8ms/-26dB (46 tones)
audible echo: min: 158ms/-36dB, max: 364ms/-24dB, avg: 201ms/-28dB (9 tones)
echo-cancellation convergence: 1840ms  ACOM: 39dB
```

In the output, ms is the delay, dB is the energy loss, and tones is how many successful measurements have been collected so far. The convergence time is the aggregate duration of voice activity required to train an echo-canceller to attenuate echo down to a sufficient low volume. ACOM (combined attenuation) is the total round-trip loss of signal energy. The shorter the former and the larger the latter indicates a better echo-cancellation algorithm.

The output of the **show chan voice-quality** command gives a more concise presentation. For example, you might get one of the following displays:

```
ch-1-vo-o, state: INACTIVE, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: 0, avg-rtt: 0, echo: na/0/0
  avg-pesq: 0.000

ch-1-vo-o, state: HOLD, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: 0, avg-rtt: 0, echo: nc/201/28
  avg-pesq: 0.000

ch-1-vo-o, state: HOLD, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: 0, avg-rtt: 0, echo: cv/1840/39
  avg-pesq: 0.000
```

"na/0/0" indicates that the measurement is not available. "nc/201/28" stands for not converged, but the current average audible delay is 201 ms, and the average energy loss is 28 dB. "cv/1840/39" indicates that echo cancellation has converged, with a convergence time of 1840 ms and an ACOM of 39 dB.

You also might want to measure the echo return loss (ERL) and echo return loss enhancement (ERLE) of a system. ERL is the average energy loss with echo-canceller disabled. ERLE is the difference of ACOM with echo-canceller enabled to the average energy loss without echo-cancellation.

Logging Out-of-Bound Latency Measurements

While performing an end-to-end-delay or round-trip-time test, you might want to log abnormal measurements that fall outside a predefined range. To specify the range, use the following configuration (time unit is milliseconds):

voice-quality logging latency longer-than *ms1* [**shorter-than** *ms2*]

Logging Voice Quality Measurement Values Externally

You can log all recorded voice quality measurements, like end-to-end-delay and jitter, in an external file, using the following command. Before performing external logging, touch the specific file in the TFTP server. The file permissions must be at least -rwxrw-rw-.

channel *startChannel#* [- *endChannel#*] **perform external-logging**

The following is an example of the type of information that would be logged.

```
global type voice
path 1 tftp://10.0.2.10/pnedunur/ch#1_e2e

channel 1 type voice mode originate
called-number 123456
duration 60 s
voice-quality type end-to-end-delay to-channel 2
voice-quality logging external path 1
interface port:2/0:0

channel 2 type voice mode terminate
called-number 123456
voice-quality type end-to-end-delay to-channel 1
interface port:2/1:0
```

Clearing Voice Quality Statistics

You can clear a channel's voice quality statistics either after every call is made or by explicitly clearing the channel counters (**until-global-clear** option). By default, voice quality statistics are not cleared "until-global-clear."

chan *n* **clear-vq-stat** {**per-call-basis** | **until-global-clear**}

PSQM

Callgen currently implements path confirmation by dialing DTMF tones back and forth between originate and terminate units. One of the drawbacks to this approach is that DTMF tones do not generate the same traffic pattern as voice. In addition, the success and failure of passing DTMF tones over a voice path does not necessarily match the receiving quality if voice was sent instead.

Therefore, to monitor the voice quality of connections, Callgen uses PSQM, an implementation of ITU-T standard P.861 that specifies the methodology to obtain subjective speech scores by analyzing the recorded and reference voice clips. PSQM has been implemented and incorporated in the Cisco Voice Quality Test (VQT) system. To this, Callgen has added extensions to develop a remote PSQM computation server. Callgen itself provides the functions of audio file playback and recording on routers. For more information about PSQM, see http://wwwin-eng.cisco.com/Eng/NUBU/WWW/Voice/TISU-Test/Tools/VQT_overview.pdf.

Measuring Speech Quality Using PSQM

You cannot use the current implementation of voice quality measurement with path confirmation, because they both need to pass audio or tones in-band. Also, you have to set up a PSQM server before starting a channel with a PSQM configuration (see Setting Up a PSQM Server, page 6-27).

The way PSQM works is similar to that of path confirmation:

1. An originate channel with voice quality configured performs a call setup to a terminate channel that also has an identical voice-quality configuration.
2. After answering the call, the terminate channel tests the connection of the terminate router and the PSQM server and then waits for the originate router to do the same test. Once both sides pass the connection test, they are ready to perform voice-quality measurement.
3. The originate channel starts recording voice incoming from the channel, and the terminate channel starts playing a reference audio file to the channel.

4 After recording the whole audio content, the originate channel writes the recorded content to the PSQM server for score calculation and then starts to play the reference audio file, and the terminate channel starts recording the incoming voice. That is, they switch the roles of recorder and player.

5 After recording the whole audio content, the terminate channel writes the recorded content to the PSQM server for score calculation. At this point, both sides wait a configurable delay (**psqm-inter-delay)** to allow the PSQM server to finish the calculation.

6 Repeats steps 3 to 6 for the duration of the call.

If the configured duration of a call is 0, each side functions as a recorder and a player at least once to finish an iteration.

To configure a channel for performing PSQM, include the following channel configuration:

voice-quality type psqm audio-file *file-tag* [{**sync** {**3-tone** | **dtmf** | **tone** *frequency duration*} | **no-sync**}] [**align-check**]

voice-quality psqm-server *IP_addr* **path** *path-tag protocol*

where:

file-tag specifies the audio file to be played.

sync specifies the type of tones to synchronize both ends in testing PSQM server reachability before running PSQM (default is 3-tone).

no-sync skips the PSQM server reachability test. If you are sure that the router has IP connectivity with the PSQM server, specify **no-sync** to speed up obtaining the first score. However, you need to ping the PSQM server at least once after booting a Callgen router to get the IP route cached.

align-check sends extra alignment tone so that the PSQM server can check whether the recording and reference audio files are well aligned. The PSQM server must be started with the alignment checking option.

IP_addr is the IP address of the PSQM server.

path-tag is the path to accommodate the recorded files on the PSQM server.

protocol can be either **tcp** or **udp** depending on the kind of socket needed to connect to the **psqm-server**. The remote PSQM server must also be using the same protocol.

If you want Callgen to connect to the PSQM server at a port other than the default (UDP:7861, TCP:7776), use the following command to specify the port on the **psqm-server** where the server will be running:

voice-quality psqm-port *port*

Note Make sure that the protocol and port numbers are the same on the server and the Callgen configuration. Otherwise, the server will not get requests for score calculation.

For example, in the following, the source of the audio file is “flash:reference.au” and the recorded files are written to */tftpboot/spchang/PSQM_FILE* on workstation 10.0.2.10:

```
global type voice
  path 1  tftp://10.0.2.10/spchang/PSQM_FILE
  audio-file 1  flash:reference.au
```

For some Linux operating systems, the */tftpboot* prefix might be required. The default audio file *reference.au* is included in the PSQM accessory file *PSQM-SPARC-xx.tar* or *PSQM-Linux-xx.tar.* You should copy it to the Flash memory of both the originate and terminate routers.

The way Callgen detects the end of an audio file during recording is to perform voice activity detection (VAD). Once a configurable duration of silence (**max-audio-silence**) is detected, the channel stops recording immediately. Therefore, it should be set a little longer than the maximum duration of silence in the reference audio file. Otherwise, a channel stops recording before its counterpart finishes playing. However, it should not be set too long just for the sake of a safe VAD, because then the recorded files will all have a long silent trailing and, hence, reduce the efficiency of the PSQM server. The default value of **max-audio-silence** is 1.6 seconds.

Note In Callgen 3.3, Callgen receives a comfort noise packet after a very short period of silence, which is about 200 milliseconds. Currently, the recording is stopped if the comfort noise packet is received before **max-audio-silence** is recorded.

Since the PSQM calculation is very resource-demanding for a PSQM server, it is suggested to not start too many concurrent channels performing PSQM measurement. Three channels for a Sun SPARC 10 and less than 10 channels for an Ultra SPARC 5 are recommended. Alternatively, you can start more channels by increasing the **psqm-inter-delay** to make the recording of each channel occur less frequently.

Here is an example of a PSQM configuration:

```
global type voice
  path 1  tftp://10.0.2.10/spchang/PSQM_FILE
  audio-file 1  flash:reference.au

channel 1 - 3 type voice mode originate
  voice-quality type psqm audio-file 1
  voice-quality psqm-server 10.0.2.10 path 1
  voice-quality max-audio-silence 500 milliseconds
  voice-quality psqm-inter-delay 8 seconds
  called-number 4085258414
  inter-call-delay 10 seconds
  start-to-start-delay 10 seconds
  interface port:0:D

channel 4 - 6 type voice mode terminate
  voice-quality type psqm audio-file 1
  voice-quality psqm-server 10.0.2.10 path 1
  voice-quality max-audio-silence 500 milliseconds
  voice-quality psqm-inter-delay 8 seconds
  called-number 4085258414
  interface port:1:D
```

The default number of channels doing PSQM is 60. If more channels with PSQM will be running concurrently (for example, 96), use the following configuration:

```
global type voice
   sys-var psqm-file-pool 96
```

Note that in this case, at least 96 temporary files have to be pre-touched using the **touch_it** script described in the next section.

To view the PSQM scores, use a command similar to the following when all channels stop running:

```
show chan psqm-scores server-path tftp://10.0.2.10/spchang/PSQM_FILE/trace.txt
```

where **trace.txt** would be **trace.html** if you specified **html** when starting the PSQM server program.

The above command shows something like the following:

```
PSQM Score Statistics

Channel  Average  Count  (0 - 1)  (1 - 2)  (2 - 3)  (3 - 4)  (4 - 5)  ( > 5 )
  1       0.200    8        8        0        0        0        0        0
  2       0.200    7        7        0        0        0        0        0
  3       0.200    7        7        0        0        0        0        0
  4       0.200    8        8        0        0        0        0        0
  5       0.200    7        7        0        0        0        0        0
  6       0.200    7        7        0        0        0        0        0
```

Setting Up a PSQM Server

Signal processing calculations are very CPU demanding, which is why the Callgen routers write the audio files recorded on the PSQM server (via TFTP). The server then processes the data. Most of the following steps only need to be done once.

To run the audio measurements feature, you must copy the binary of the server in the directory of your TFTP server. For optimum results, it is strongly recommended that you have your TFTP server set up locally to your testbed, and that you have a powerful server (avoid Sparc 5). If the TFTP/PSQM server is too busy, you can experience inaccurate poor scores.

If you already have a TFTP server running, you can skip the following.

There are two kinds of TFTP servers: *in.tftpd* and *tftpd*. The main difference is that with *in.tftpd* you must create the files before, and with *tftpd* the files are automatically created when needed. To know which one is currently running, use **more /etc/inetd.conf**:

```
(...)
# Tftp service is provided primarily for booting. Most sites run this
# only on machines acting as "boot servers."
#
tftp   dgram   udp     wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
(...)
```

Installing in.tftpd

Step 1 Copy the binary *in.tftpd* to */usr/sbin* if it does not exist.

Step 2 Edit the */etc/inetd.conf* file and insert the following line (each block is separated by a tabulation):

```
tftp dgram udp wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
```

Step 3 Do a **ps -A | grep inetd** to get the pid of the inet daemon. Do **kill -1** *pid* to tell the daemon to reread its configuration file.

Installing tftpd

Step 1 Copy the binary *tftpd* to */usr/sbin* if it does not exist. The binary is in */autons/vts/bin/image*.

Step 2 Create the file *tftpd.conf* in */etc* with the following content:

```
defaultDirectory        /tftpboot
debugLevel      7
inputWait       300
accessList      1       readwrite       0.0.0.0 0xffffffff
defaultAccessList 1
```

Step 3 Edit the */etc/inetd.conf* file and insert the following line (each block is separated by a tabulation):

```
tftp dgram udp wait root /usr/sbin/tftpd tftpd -s /tftpboot
```

Step 4 Do a **ps -A | grep inetd** to get the pid of the inet daemon. Do **kill -1** *pid* to tell the daemon to reread its configuration file.

Setting Up Your Environment

Step 1 Create the directory PSQM_FILE in */tftpboot/<username>*.

Step 2 Download the PSQM accessory file *PSQM-SPARC-xx.tar* or *PSQM-Linux-xx.tar* to */tftpboot/<username>/PSQM_FILE/*. Use the command **tar xvf PSQM-xxxxx-xx.tar** to unpack all accessory files.

Step 3 Pre-touch the recorded files. The following script creates the files needed by the routers to check if the TFTP server is reachable as well as to touch all possible filenames of recorded files. The username is the name on the PSQM server.

touch_it {**in.tftpd** | **tftpd**} *username router_name* [*number_of_files*]

For example, in the following the username is “spchang” on the PSQM server, and a Callgen router named “cg-5300-1” will be used for PSQM measurement:

```
touch_it in.tftpd spchang cg-5300-1 96
```

If you have two Callgen routers, you must do this for each *router_name*. The default *number_of_files* touched by the script is 60. If your test has to run more than 60 concurrent channels with PSQM, use the *number_of_files* parameter.

Step 4 Make sure that all the appropriate rights are turned on (use **chmod 666 ***).

Running the Server

You can run the PSQM server with the following options:

-m—Specifies whether the PSQM scores are calculated in real time, or whether the server saves the PSQM requests offline and then runs the server in batch mode to handle all the commands. The options are **realtime**, **offline**, and **batch**.

Offline mode queues PSQM score calculation requests from remote routers in a file named *jobs*. This offline feature is beneficial for slow servers, because the scores are not calculated immediately but calculated on request by running the server in batch mode.

Note The *jobs* file is not created unless you run the **touch_it** utility provided. It also creates a *temp*/ folder to store the audio files for which the scores need to be calculated.

-x—Specifies the name of the tftpd program used on the TFTP server. The options are **in.tftpd** (the default) or **tftpd**.

-p—Specifies the transfer protocol of the message socket with Callgen. The options are **tcp** or **udp** (the default).

-l—Specifies the port on which the PSQM server runs, ranging between 1024 and 10,000. This command allows you to run multiple PSQM servers on the same machine and process requests from different Callgen routers.

-r—Specifies the reference audio file that the PSQM function uses for its calculations. The file is in 16-bit linear format. The default is *reference.bin*.

-a—Specifies strict file-size checking. By default, this is turned off.

-b—Specifies a bad score threshold. If the calculated PSQM scores are greater than the value specified, they are recorded in */tftpboot/<username>/PSQM_FILE/BAD_AUDIO/* for later review.

-o—Specifies the file format for the PSQM scores. Generates either *trace.txt* or *trace.html*, depending on the option selected: **text** (the default) or **html**.

-w—Specifies what type of audio wave form to generate. If you specify 1, only the recorded audio wave form is plotted. If you specify 2, it plots the recorded and referenced audio wave forms together on one plot. You must specify the **-o html** option, since the plot is generated on an html file.

If you do not specify any options when running the server (that is, you start the server using the command **server_psqm**), the default parameters are used, which is equivalent to the following:

server_psqm -m realtime -x in.tftpd -p udp -r reference.bin -b 3.0 -o text -l 7861

If the server does not run, most likely it is a problem with the ucb library. Add the following line to your *.cshrc* file:

setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH":"/usr/ucblib

If you type **server_psqm**, the following should appear, indicating that the server is now ready:

```
PSQM Server Program Version 3.2
          Copyright (c) 2000-2001 by Cisco Systems, Inc.

          To display usage: server_psqm -h

          Following shows current setting:
          tftpd=          in.tftpd
          protocol/port=  UDP/7861
          reference=      reference.bin
          threshold=      3.000
          html trace=     No
          waveform plot=  No
          alignment=      No

          Wait for requests from Callgen...
```

Reviewing Results

Once everything is running, you can view the result on the console or in the file *trace.txt*. For example:

```
PSQM: 0.202  out:0.00%  (xcorr:85% MSR:13.4)     (cg-5300-1 ch#0001)  00:03:40.767
PSQM: 0.218  out:0.00%  (xcorr:90% MSR:12.5)     (cg-5300-1 ch#0002)  00:04:10.747
PSQM: 3.299  out:30.05% (xcorr:51% MSR:4.1)      (cg-5300-2 ch#0001)  00:04:14.139
PSQM: 0.198  out:0.00%  (xcorr:89% MSR:19.0)     (cg-5300-2 ch#0002)  00:04:44.015
PSQM: 0.200  out:0.00%  (xcorr:88% MSR:18.8)     (cg-5300-1 ch#0001)  00:04:47.435
```

The output contains the following information:

- The first number (for example, 0.202) is the PSQM score. This score is highly correlated with a corresponding MOS score: if you have a good PSQM score, you would have a good MOS score in most of the cases. Be aware that a PSQM score is to a certain extent the inverse of a MOS score: 0 is perfect.

  If you have a score under 2, the quality is good. If you have a score above 5, it basically means that you have bad quality. Between 2 and 5, the quality might or might not be acceptable, so it should be checked further.
- The second number (for example, 0.00%) is the percentage of outliers. The PSQM algorithm is performed for every frame of 256 samples, and it then issues a score. The final PSQM score is the average of all the scores. An outlier is a score that is not close enough to the average.
- The numbers following xcorr and MSR (Max-to-Second-Ratio) are explained in the section Cross-Correlation and MSR Values (page 6-30), since some background on pseudorandom sequence and the cross-correlation (xcorr) of random sequence is needed to understand these numbers.
- The second to last column (for example, cg-5300-1) means that the file was recorded on the router Term5300 on channel 1.
- The last argument (for example, 00:03:40.767) is the time when it was recorded.

Cross-Correlation and MSR Values

Because the PSQM server program needs to align recording and reference waveforms, Callgen plays a short noise-like sequence ahead of every audio file to indicate the beginning. The recorded audio has this sequence close to the beginning (the static you heard at the beginning of each recorded file). Based on that, the PSQM server program can estimate the beginning of real voice content to a 125uS (1/8 ms) accuracy by searching for the peak of the xcorr value.

If there is an obvious peak of xcorr detected, you would get a high xcorr and MSR value (the ideal case is xcorr=100%, MSR=infinity). That means the two waveforms are well aligned and, hence, you can have more confidence in the PSQM score. If the values are low, the alignment probably failed and the PSQM score is questionable.

For T1/E1, you should have xcorr>30% and MSR>3.0 (POTS has a smaller MSR); otherwise, you might want to ignore that specific score.

For example, if you have the following:

```
PSQM= 0.280  Outliers=0.00%     (Xcorr=90% MSR:15.1)    (5300-21 Ch#001)
PSQM= 2.139  Outliers=14.17%    (Xcorr=59% MSR:1.1)     (5300-24 Ch#002)
PSQM= 2.169  Outliers=16.17%    (Xcorr=64% MSR:1.3)     (5300-24 Ch#002)
PSQM= 0.281  Outliers=0.00%     (Xcorr=90% MSR:15.1)    (5300-21 Ch#001)
```

If the maximum length sequence (MLS) detection cannot find an obvious peak, the PSQM score is problematic, since you cannot trust the offset. The MLS is sent before Callgen plays the reference file. For the above values, MSR=1.1 and 1.3, these two cases happened because the F packet was lost during MLS:

```
original MLS: ABCDEFGHIJK
received MLS: ABCDEGHIJK
```

There will be two peaks in this case: one thought it begins with A (it does!); the other thought it begins with a packet before A (wrong!).

```
original:                  ABCDEFGHIJK
peak1 thought it received:  ABCDEGHIJK.
                           +++++------

peak2 thought it received:  .ABCDEGHIJK
                           ------+++++
```

If you get a score worse than the threshold you set (default 3.0), the server must have saved the corresponding audio file, so you can check the quality by listening to it. In the example above, the corresponding file would be:

BAD_AUDIO/cg-5300-2_0001_00:04:14.139.au

The audio format is .au 8kHZ, 8-bit, mu-law. Mu-law is the native format for Sun and can be played by many audio player programs.

Creating an Audio Test File

Sometimes you might want to create your own audio test file to test a different language or a specific type of voice (male, female, or child). Another reason might be if you have unexpected poor results when linking two Callgen routers together.

In the first case, you must replace the sound file (for example, *reference.au*) in your router. This is a classic mu-law audio file with header (default Sun audio format). Pay attention not to take a file that is too long (6 to 8 seconds is the maximum), because the memory available on a router is limited and the longer it is, the longer it takes to process on the server, possibly resulting in bad scores.

To create an audio test file, you must create the corresponding reference file in the TFTP/PSQM server location. You cannot use the existing file, because the DSP playing/recording operations on the routers are not transparent.

Step 1 Run PSQM on the Callgen routers. You will notice that most of the scores are bad (because we are still using the old reference file on purpose) but identical (same PSQM score and same percentage of outliers). Take one of the *.au* files from PSQM/BAD_AUDIO. We will use *hello.au* for this example.

Step 2 Find the beginning of the audio data using the command **xcorr hello.au**. This program also converts the recorded file into 16 bits linear.

Step 3 Delete the number of samples given by **xcorr** (“offset”) at the beginning of the audio file, using an audio edit tool like coolEdit. Save the clipped audio as *hello2.au*.

Step 4 Use the command **addAUinfo hello2.au hello3.au** to fix the header bug introduced by coolEdit. It does not negatively affect anything if the header is already correct.

Step 5 Use the command **au2bin hello3.au** to get a new reference *.bin* file (*hello3.bin*) to use on the PSQM server. You must also copy *hello3.au* to Callgen routers as the new audio file to be played.

Viewing Voice Quality Statistics

To briefly list voice-quality statistics of all voice channels measured so far, use the command **show channel voice-quality**. For example:

```
ch-1-vo-o, state: INACTIVE, CSR:98.252%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 1%, hit: 0, clip: 0, avg-e2e: 0, avg-rtt: 0, echo: na/0/0
  avg-pesq:0.000
```

```
ch-2-vo-o, state: INACTIVE, CSR:99.250%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 7, clip: 1, avg-e2e: 0, avg-rtt: 0, echo: na/0/0
  avg-pesq:0.000

ch-3-vo-o, state: INACTIVE, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e:88, avg-rtt: 0, echo: na/0/0
  avg-pesq:0.000

ch-4-vo-o, state: INACTIVE, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.4/+0.4,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: 0, avg-rtt: 0, echo: na/0/0
  avg-pesq:0.000
```

where:

CSR is the current call success ratio (confirms/attempts)

avg-psqm is the average PSQM score

avg-jtr shows the average lead (-) and lag (+) jitters

tone-err is the path-confirmation tone error percentage

clip/hit is the clip and hit counts

avg-e2e is the average end-to-end delay

avg-rtt is the average round-trip time

echo shows current echo measurement status

avg-pesq is the average PESQ score

Note In Callgen, you can only configure a channel to do one type of voice-quality measurement at a time. So most numbers, except the configured measurement listed for a channel, are 0, which actually means "not available."

To update PSQM scores, include a PSQM work directory. For example:

show channel voice-quality psqm-path tftp://10.0.2.10/spchang/PSQM_FILE/trace.txt

Without the path, the listed PSQM scores are either 0 or the previous update.

You can use channel filtering commands, such as **in-class** and **in-hold-state**. For example:

show channel in-class orig_pri_psqm voice-quality

PESQ

Perceptual Evaluation of Speech Quality (PESQ) is an enhanced measurement for voice quality in telecommunications. PESQ was specifically developed for end-to-end voice quality testing under real network conditions, such as VoIP, POTS, ISDN, and GSM. PESQ was officially approved as the new ITU-T recommendation P.862, and it is meant to be the successor of ITU-T P.861/PSQM. KPN Research of the Netherlands and British Telecommunications (BT) developed PESQ by combining two advanced speech quality measures, PSQM+ and PAMS (see www.pesq.org).

Measuring Speech Quality Using PESQ

You cannot use the current implementation of voice quality measurement with path confirmation, because they both need to pass audio or tones in-band. Also, you have to set up a PESQ server before starting a channel with a PESQ configuration. Setting up a PESQ server is the same as setting a PSQM server, except you are running a different server program (see Setting Up a PSQM Server (page 6-27)). Also, you should only configure either PESQ or PSQM but not both in a channel.

The way PESQ works is similar to path confirmation:

1. An originate channel with voice quality configured performs a call setup to a terminate channel that also has an identical voice-quality configuration.
2. After answering the call, the terminate channel tests the connection of the terminate router and the PESQ server and then waits for the originate router to do the same test. Once both sides pass the connection test, they are ready to perform voice-quality measurement.
3. The originate channel starts recording voice incoming from the channel, and the terminate channel starts playing a reference audio file to the channel.
4. After recording the whole audio content, the originate channel writes the recorded content to the PESQ server for score calculation and then starts to play the reference audio file. The terminate channel starts recording the incoming voice. That is, they switch the roles of recorder and player.
5. After recording the whole audio content, the terminate channel writes the recorded content to the PESQ server for score calculation. At this point, both sides wait a configurable delay (**psqm-inter-delay**) to allow the PESQ server to finish the calculation. (Just borrow this command from PSQM for convenience.)
6. Repeats steps 3 to 5 for the duration of the call.

If the configured duration of a call is 0, each side functions as a recorder and a player at least once to finish an iteration.

To configure a channel for performing PESQ, include the following channel configuration:

voice-quality type pesq audio-file *file-tag* [{**sync** {**3-tone** | **dtmf** | **tone** *frequency duration*} | **no-sync**}]

voice-quality pesq-server *IP_addr* **path** *path-tag protocol*

where:

file-tag specifies the audio file to play.

sync specifies the type of tones to synchronize both ends in testing PSQM server-reachability before running PESQ (default is 3-tone).

no-sync skips the PESQ server-reachability test. If you are sure that the router has IP connectivity with the PESQ server, specify **no-sync** to speed up obtaining the first score. However, you need to ping the PESQ server at least once after booting a Callgen router to get the IP route cached.

IP_addr is the IP address of the PESQ server.

path-tag is the path to accommodate the recorded files on the PESQ server.

protocol can be either **tcp** or **udp**, depending on the kind of socket needed to connect to the **pesq-server**. The remote PESQ server must also be using the same protocol.

If you want Callgen to connect to the PESQ server at a port other than the default (UDP:7862, TCP:7377), use the following command on the **pesq-server** where the server is running:

voice-quality pesq-port *port*

Note Make sure that the protocol and port numbers are the same on the server and the Callgen configuration. Otherwise, the server will not get requests for score calculation.

In release 4.0T, the audio recording part of the PESQ is redesigned. In this new design, a fixed length of audio is recorded and the parameter **max-audio-silence** is not used. Callgen does not exactly know when to start recording, so the parameter **pesq-pre-record-delay** is used to control when to start recording. Ideally, you should start recording when the playing happens. If not, try to record slightly after the playing starts. The reference file contains more than 1 second of silence before the voice starts, so you can miss some initial silence. A default value based on a good value for 5300 ISDN back-to-back topology is defined for backward compatibility with version 3.3T. The default is 1100 milliseconds.

voice-quality pesq-pre-record-delay *time* **milliseconds**
voice-quality pesq-post-play-delay *time* **milliseconds**

Pesq-post-play-delay delays the originating channel from disconnecting the call. Sometimes the originating channel might disconnect a call before the terminating channel finishes recording. A default value based on a good value for 5300 ISDN back-to-back topology is defined for backward compatibility with version 3.3T. The default is 1000 milliseconds.

In the following, the source of the audio file is flash:pesqReference.wav, and the recorded files are written to */tftpboot/spchang/PSQM_FILE* on workstation 10.0.2.10:

```
global type voice
  path 1  tftp://10.0.2.10/spchang/PSQM_FILE
  audio-file 1  flash:pesqReference.wav
```

The default audio file *pesqReference.wav* is included in the PSQM/PESQ accessory file *PSQM-PESQ-SPARC-xx.ta*r. You should copy it to the Flash memory of both the originate and terminate routers.

Note There is no PESQ server available for Linux machines, and there are no future plans for having one.

The way Callgen detects the end of an audio file during recording is to perform voice activity detection (VAD). Once a configurable duration of silence (**max-audio-silence**) is detected, the channel stops recording immediately. Therefore, it should be set a little longer than the maximum duration of silence in the reference audio file. Otherwise, a channel stops recording before its counterpart finishes playing. However, it should not be set too long just for the sake of a safe VAD, because then the recorded files will all have a long silent trailing and, hence, reduce the efficiency of the PESQ server. The default for **max-audio-silence** is 1.6 seconds.

Note In Callgen 3.3, Callgen receives a comfort noise packet after a very short period of silence, which is about 200 milliseconds. Currently, the recording is stopped if the comfort noise packet is received before **max-audio-silence** is recorded.

Since the PESQ calculation is very resource-demanding for a PESQ server, it is suggested to not start too many concurrent channels performing PESQ measurement. Three channels for a Sun SPARC 10 and less than 10 channels for an Ultra SPARC 5 are recommended. Alternatively, you can start more channels by increasing the **psqm-inter-delay** so that the recording of each channel occurs less frequently.

Here is an example of a PESQ configuration:

```
global type voice
  path 1  tftp://10.0.2.10/spchang/PSQM_FILE
  audio-file 1  flash:pesqReference.wav

channel 1 - 3 type voice mode originate
  voice-quality type pesq audio-file 1
  voice-quality pesq-server 10.0.2.10 path 1
  voice-quality psqm-inter-delay 8 seconds
  called-number 4085258414
  inter-call-delay 10 seconds
  start-to-start-delay 10 seconds
  interface port:0:D

channel 4 - 6 type voice mode terminate
  voice-quality type pesq audio-file 1
  voice-quality pesq-server 10.0.2.10 path
  voice-quality psqm-inter-delay 8 seconds
  called-number 4085258414
  interface port:1:D
```

The default number of channels doing PESQ is 60. If more channels with PESQ will be running concurrently (for example, 96), use the following configuration:

```
global type voice
   sys-var psqm-file-pool 96
```

Note that in this case, at least 96 temporary files have to be pre-touched using the **touch_it** script (Setting Up a PESQ Server (page 6-36)). This command is also borrowed from PSQM for the PESQ feature.

To view PESQ scores, use a command similar to the following when all channels stop running:

```
show chan pesq-scores server-path
tftp://10.0.2.10/spchang/PSQM_FILE/_pesq_itu_results.txt
```

The above command shows something like the following:

```
PESQ Score Statistics

Channel  Average  Count  ( < 0)  (0 - 1)  (1 - 2)  (2 - 3)  (3 - 4)  (4 - 5)
  1       4.232    2        0        0        0        0        0        2
```

Note We recommend that you configure the PESQ channel with no duration so that a call will be disconnected after each side does one playing and recording. The longer the call duration, the higher chance that the originating channel and the terminating channel will be out of sync timing wise.

Note PRI calls have more consistent PESQ score results than CAS and analog calls.

Note If PRI interfaces are used, make sure that the clock source of one controller is configured “internal” and the other side of the clock source is configured “line.” Use **show controller** to make sure that there is not a clocking or framing problem before starting the PESQ measurement in Callgen.

Setting Up a PESQ Server

Setting up a PESQ server uses the same process as setting up a PSQM server. See Setting Up a PSQM Server (page 6-27) for details.

Step 1 Create the directory PSQM_FILE in */tftpboot/<username>*.

Step 2 Download the PSQM/PESQ accessory file *PSQM-PESQ-SPARC-xx.tar* to */tftpboot/<username>/PSQM_FILE/*. Use the command **tar xvf PSQM-PESQ-xxxxx-xx.tar** to unpack all accessory files.

Step 3 Pre-touch the recorded files. The following script creates the files needed by the routers to check whether the TFTP server is reachable as well as touches all possible filenames of recorded files. The username is the name on the PESQ server.

touch_it {**in.tftpd** | **tftpd**} *username router_name* [*number_of_files*]

In the following, the username is spchang on the PESQ server, and a Callgen router named cg-5300-1 is specified for PESQ measurement:

```
touch_it in.tftpd spchang cg-5300-1 96
```

If you have two Callgen routers, you must do this for each router name. The default number of files touched by the script is 60. If your test has to run more than 60 concurrent channels with PESQ, use the *number_of_files* parameter.

Step 4 Make sure that all the appropriate rights are turned on (use **chmod 666 ***).

Running the Server

You can run the PESQ server with the following options:

-p—Specifies the transfer protocol of the message socket with Callgen. The options are **tcp** or **udp** (the default).

-l—Specifies the port on which the PESQ server runs, ranging between 1024 and 10,000. This command allows you to run multiple PESQ servers on the same machine and process requests from different Callgen routers.

-r—Specifies the reference audio file that the PESQ function uses for its calculations. The file is in 16-bit linear format. The default is *pesqReference.bin*.

-s—Specifies the sample rate. Either 8000Hz (the default) or 16000Hz. However, Callgen only supports a 8000Hz sample rate

-b—Specifies the bad score threshold. The default is 3.0. If the scores are below this threshold, the degraded audio files are saved into the BAD_AUDIO directory. You can relate a PESQ score entry with the saved degraded file via the timestamp.

You must specify the reference file when running the server:

```
server_psqm -r pesqReference.bin
```

If the server does not run, it is most likely a problem with the ucb library. Add the following line to your *.cshrc* file:

setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH":"/usr/ucblib

If you type `pesqServer - r pesqReference.bin`, the following appears, indicating that the server is ready:

```
PESQ Server Program Version 1.0
Copyright (c) 2002 by Cisco Systems, Inc.

To display usage: pesqServer -h

Following shows current setting:
protocol/port=  UDP/7862
reference=      pesqReference.bin

Wait for requests from Callgen...
```

Reviewing Results

After everything is running, you can view the results in the file *_pesq_itu_results.txt* or on the console. For example:

```
ROUTER#CHANNEL   TIMESTAMP       REFERENCE       DEGRADED        PESQMOS
SUBJMOS  COND    SAMPLE_FREQ     CRUDE_DELAY
(cg-5300-7 ch#0001)        00:42:18.315         or109.wav       cg-5300-7#59.wav
         SQValue=2.699   0.000           0       8000           -1.0840
(cg-5300-8 ch#0001)       00:40:07.307         or109.wav       cg-5300-8#59.wav
         SQValue=4.216   0.000           0       8000           1.2960
```

The output contains the following information:

- The fourth column (with the **SQValue=**prefix) contains the PESQ score. This score is highly correlated with a corresponding MOS score. If you have a good PESQ score, you have a good MOS score in most cases. The perfect PESQ score is 4.5.
- The first column (for example, cg-5300-7 ch#0001) shows that the file was recorded on the router cg-5300-7 on channel 1.

Creating an Audio Test File

Sometimes you might want to create your own audio test file to test a different language or a specific type of voice (male, female, or child). Another reason might be if you have unexpected poor results when linking two Callgen routers together.

In the first case, you must replace the sound file (for example, pesq*Reference.wav*) in your router. This is a classic 8-bit mu-law audio file with header (.wav format). Make sure you do not take a file that is too long (6 to 8 seconds is the maximum), because the memory available on a router is limited and the longer it is, the longer it takes to process on the server, possibly resulting in bad scores.

To create an audio test file, you must create the corresponding reference file in the TFTP/PESQ server location. (This file is a 16-bit PCM .wav file.) You cannot use the existing file, because the DSP playing/recording operations on the routers are not transparent.

Viewing Voice Quality Statistics

To briefly list voice-quality statistics of all voice channels measured so far, use the **show channel voice-quality** command. For example:

```
ch-1-vo-o, state: INACTIVE, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: N/A, avg-rtt: 0, echo: na/0/0
  avg-pesq: 4.232
```

```
ch-3-vo-o, state: INACTIVE, CSR: 0.000%, avg-psqm: 0.000, avg-jtr: -0.0/+0.0,
  tone-err: 0%, hit: 0, clip: 0, avg-e2e: N/A, avg-rtt: 0, echo: na/0/0
  avg-pesq: 0.000
```

where:

CSR is the current call success ratio (confirms/attempts)

avg-psqm is the average PSQM score

avg-jtr shows the average lead (-) and lag (+) jitters

tone-err is the path-confirmation tone error percentage

clip/hit is the clip and hit counts

avg-e2e is the average end-to-end delay

avg-rtt is the average round-trip time

echo shows the current echo measurement status

avg-pesq is the average PESQ score

Note In Callgen, you can only configure a channel to do one type of voice-quality measurement at a time. So most numbers, except the configured measurement listed for a channel, are 0, which actually means "not available."

To update PESQ scores, include a PESQ work directory. For example:

show channel voice-quality pesq-path tftp://10.0.2.10/spchang/PSQM_FILE/_pesq_itu_results.txt

Without the path, the listed PESQ scores are either 0 or the previous update.

E911 Calls on an Analog Interface

Callgen tests E911 calls by terminating 911 calls on an FXS-DID card. If the FXS-DID port is configured with CAMA callback (either long or short), the FXS-DID can collect ANI digits after the 911 digits. The collected ANI digits are sent to Callgen for further verification.

The FXS-DID voice port should be configured as follows

```
voice-port 6/1/1
signal did cama short
timing digit 75
timing inter-digit 65
```

If the voice port is not configured as a CAMA signal, it acts like a regular FXS-DID port. The following formats of an ANI sent from the CAMA (FXO-M1) interface card can be verified by terminating the Callgen router:

- KP-2-ST—None of the ANI digits are dialed out except digit 2.
- KP-0-NPA-NXX-XXXX—All 10 digits of the ANI are dialed out preceded by a 0.
- KP-0-NXX-XXXX-ST—Seven digits of the ANI are dialed out preceded by a number between 0 and 3.

Sample Configuration of a CAMA Port

The CAMA port should be configured to specify the type of ANI that needs to be dialed out. For example:

```
voice-port 6/1/1
signal cama KP-0-NXX-XXXX-ST
```

Sample Configuration on a Terminating Callgen Router

For signaling type KP-2-ST, **verify-ani** must be configured under the channel configuration.

For signaling type KP-0-NPA-NXX-NXXX, the channel should contain all 10 digits of the ANI **verify-ani** CLI. For example:

```
Channel 1 type voice mode terminate
   Called-number 911
   Verify-ani 4088538540
   path-confirmation type ping string 123
   interface port:1/0/0
```

For signaling type KP-0-NXX-XXXX-ST, since we do not know the first incoming digit before collecting ANI digits, we use a wildcard for the ANI number that needs to be verified. For example:

```
Channel 1 type voice mode terminate
   Called-number 911
   Verify-ani ,8538540
   path-confirmation type ping string 123
   interface port:1/0/0
```

You can only configure **verify-ani** for the terminate channels. If **verify-ani** is configured, the following counters are shown as part of **show channel** statistics:

```
ani-match: 1
  ani-mismatch: 0
  callback: 0
  call confirms: 1
  callback confirm: 0
```

Only path-confirmation type ping works with E911 calls. If **verify-ani** is configured along with path-confirmation, the call confirm counter is incremented but not the confirms counter. Based on the ANI information received, either **ani-match** or **ani-mismatch** is incremented, which can be seen using **show channel** *channel_number*. If the CAMA signal is not configured, the ANI digits are not collected. If path-confirmation is configured and successfully executed, the regular confirms counter is incremented like any other call.

Skinny IP Phone Simulation

Callgen has been successfully used to test telephony interfaces as well as H.323 and SIP VoIP features of Cisco products. Callgen can directly leverage most signaling protocols supported on IOS voice-enabled routers, such as analog, CAS, R2, BRI/PRI, VoFR, and VoAAL5.

However, a major missing link is the Cisco-proprietary Skinny protocol, which is used between the Cisco Call Manager and Cisco IP phones. Since the Skinny protocol is not intended to be used between a gateway and a Call Manager (MGCP is used on VG200, Cat6K, and 2600/3600 instead), it has never been implemented on IOS routers.

Vespa is the first project to build Skinny into IOS to enable Call Manager fallback caused by WAN outage. It also provides a soft-version, keyswitch system of IP phones. Because the IP phone simulation feature depends on Vespa’s code, it was developed on a separate branch based on the

Vespa project branch. The images containing the IP phone simulation feature are released as special images that are Callgen 3.0-based.

With the Callgen 4.0T release, the IP phone simulation feature is merged back to the main Callgen branch.

Simulating Two IP Phones

The following shows how to simulate two IP phones.

Step 1 Configure the command **ephone-emulation** in IOS configuration mode, which starts the Skinny client process to take care of Skinny station messages to Call Managers.

Step 2 In Callgen mode, enter the following configuration:

```
global type voice
  sys-var max-ip-phones 2

channel 1 type voice mode originate
  duration 3 minutes
  inter-call-delay 3 seconds
  calling-number 1001
  called-number 3001
  interface port:51/0/1
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.2.18
  ip-phone mac-address 00B0.6409.BDE5
  ip-phone call-manager-ip 10.0.2.4 port 2000
```

The **max-ip-phones** command creates soft interface ports numbered from 51/0/1 to 51/0/*n*, representing the jacks available for the number of simulated IP phones. **max-ip-phones** must always be configured before configuring the interfaces of the channels. Some IP phone-specific configurations (shown with **ip-phone** prefix) require that the interface port be specified first.

In this example, the IP address and MAC address of the Ethernet interface installed on the router are given in the channel configuration. That is not a requirement, since it would limit the total number of simulated IP phones to the number Ethernet interfaces a router could have. (To simulate multiple IP phones using a single physical interface, see Simulating Multiple IP Phones (page 6-42).)

Step 3 Enter the **show channel** command. It should show that the channel state is UNREG, which means the channel has not yet registered with a Call Manager.

Step 4 Enter **channel 1 cm-register** to change the channel state to INACTIVE.

After this point, channel 1 functions as if it were an analog phone with most Callgen functionalities retained. However, real DSP functions, such as tone generation, are not possible, since no DSP is involved or required. For example, in the above configuration, the device configuration with MAC address 00B0.6409.BDE5 and directory number 1001 is on Call Manager 10.0.2.4, and a real IP phone is registered to the same Call Manager with directory number 3001. The Callgen command **start 1 total-call 1** will ring the real phone. By doing so, we are actually calling a real IP phone from a simulated IP phone, so both the real IP phone and the Call Manager are being tested.

To test a Call Manager solely, calls must be terminated by a simulated IP phone. The following configuration is of a terminate channel functioning as an answering-side IP phone:

```
channel 2 type voice mode terminate
  called-number 3001
  interface port:51/0/2
```

```
    ip-phone model telecaster
    ip-phone codec g711ulaw
    ip-phone source-ip-address 10.0.2.19
    ip-phone mac-address 00B0.6409.CCC7
    ip-phone call-manager-ip 10.0.2.4 port 2000
```

Note that the source IP address and MAC address are different than those of channel 1. Assuming that a device configuration with MAC 00B0.6409.CCC7 and directory number 3001 has been given on Call Manager 10.0.2.4, the command **channel 2 cm-register** registers the simulated IP phone to the Call Manager and changes the Callgen channel state from UNREG to INACTIVE. Then, issuing the command **start 2** changes the state of channel 2 to IDLE, making it ready to answer incoming calls. If you enter the command **start 1**, you should see channel 1 making calls and channel 2 receiving calls over and over. The **stop** command stops the current call immediately. The command **channel 1 - 2 cm-unregister** unregisters both channels from the Call Manager.

You could also verify the signaling path along the whole call duration by configuring **path-confirmation type ping** and then checking the **confirms** statistic after a call stops. Since simulated IP phones have no RTP traffic, there are only keepalive messages after call setup. To enable RTP packet generation by playing out an audio file repeatedly, use the following command and add the following script in the channel configuration. The audio file *reference.au* must be copied over to the router’s Flash memory first.

```
global type voice
  audio-file 1 flash:reference.au

script { pl 1 ls }
```

You can also configure **path-confirmation type 3-tone play** *audio-file-tag* to verify the bearer path when CODEC type **g711ulaw** is configured. See Path Confirmation (page 6-8) for more details.

The CLI for configuring CODEC types is:

ip-phone codec {**g711ulaw** | **g7231 bitrate** {**5.3kbps** | **6.4kbps**} | **g729r8**}

Note You can only configure up to two phones (one originate and one terminate channel) per IP phone port. The two phones with the same IP phone port must have the same IP phone-related configuration (parameters with the **ip-phone** prefix). You can configure two phones sharing the same port when either phone is unregistered.

Note Callgen allows you to configure different CODEC types for signaling purposes. However, Callgen is only capable of generating g711ulaw RTP packets.

Note With CallManager 3.1, the integer value of a phone type telecaster was changed from 6 to 7. For Callgen image to work with both versions of CallManager, we have added a new phone model option. If you test the older version of CallManager, configure **ip-phone model telecaster** for a 7960 phone. If you test the newer version of CallManager, configure **ip-phone model telecastr-mgr** for a 7960 phone. You might need to change your existing scripts if you’re testing different versions of CallManager.

Note Make sure that **ip routing** is configured on the Callgen router. At one point, this IOS configuration was configured under the Ethernet interface, but it can also be configured under global configuration mode in some versions of IOS. By default, **ip routing** is enabled.

Simulating Multiple IP Phones

To simulate multiple IP phones, first configure **ephone-emulation** in IOS configuration mode. Then, configure a loopback interface with multiple IP addresses for all the source IP addresses to be used for simulated IP phones. For example:

```
interface Loopback10
 ip address 10.0.10.1 255.255.255.255
 ip address 10.0.10.2 255.255.255.255 secondary
.............
 ip address 10.0.10.99 255.255.255.255 secondary
 ip address 10.0.10.100 255.255.255.255 secondary
```

The following example configures 100 IP phones. The simulated IP phones are in the 10.0.10.0 subnet, and the Call Manager is in the 10.0.2.0 subnet, so the appropriate **ip route** configuration has to be done on the default gateway of the Call Manager to make 10.0.10.0 reachable.

```
global type voice
  sys-var max-ip-phones 100

channel 1 type voice mode originate
  calling-number 8001
  called-number 9001
  interface port:51/0/1
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.10.1
  ip-phone mac-address 05C0.CA11.0001
  ip-phone call-manager-ip 10.0.2.4 port 2000

channel 2 type voice mode originate
  calling-number 8002
  called-number 9002
  interface port:51/0/2
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.10.2
  ip-phone mac-address 05C0.CA11.0002
  ip-phone call-manager-ip 10.0.2.4 port 2000
..........

channel 50 type voice mode originate
  calling-number 8050
  called-number 9050
  interface port:51/0/50
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.10.50
  ip-phone mac-address 05C0.CA11.0032
  ip-phone call-manager-ip 10.0.2.4 port 2000

channel 51 type voice mode terminate
  called-number 9001
  interface port:51/0/51
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.10.51
  ip-phone mac-address 05C0.CA11.0033
  ip-phone call-manager-ip 10.0.2.4 port 2000
..........

channel 100 type voice mode terminate
  called-number 9050
  interface port:51/0/100
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.10.100
  ip-phone mac-address 05C0.CA11.0064
  ip-phone call-manager-ip 10.0.2.4 port 2000
```

In this case, MAC addresses are virtual rather than a physical ID. They are defined just to be used as the name of the IP phones. The actual MAC used to send out an Ethernet packet containing a Skinny message depends on the interface selected for the transmission. Before making a call, configure phone devices on Call Manager 10.0.2.4 with the following pairs (MAC address, directory number):

```
(05C0.CA11.0001, 8001)
(05C0.CA11.0002, 8002)
..........

(05C0.CA11.0032, 8050)
(05C0.CA11.0033, 9001)
..........

(05C0.CA11.0064, 9050)
```

With this configuration, enter **channel 1 - 100 cm-register** and check whether all the channels go into INACTIVE state successfully. If yes, the command **start total-calls 50** will simulate 50 IP phones calling another 50 IP phones simultaneously.

Configuring IP Phone Failover with Multiple Call Managers

To test the Call Manager failover feature, you can configure an IP phone with multiple Call Managers. When the primary Call Manager fails, the IP phone tries to register to the back-up Call Managers. You can configure a maximum of two back-up Call Managers per IP phone.

```
global type voice
  sys-var max-ip-phones 2

channel 1 type voice mode originate
  calling-number 8050
  called-number 9050
  interface port:51/0/1
  ip-phone model telecaster
  ip-phone codec g711ulaw
  ip-phone source-ip-address 10.0.10.100
  ip-phone mac-address 05C0.CA11.0064
  ip-phone call-manager-ip 10.0.2.4 port 2000
  ip-phone call-manager-ip 10.0.2.5 port 2000 backup1
  ip-phone call-manager-ip 10.0.2.5 port 2000 backup2
```

Configuring an IP Keyswitch

The following IP keyswitch configuration replaces the Call Manager used in the example in the section Simulating Two IP Phones (page 6-40).

```
keyswitch
 ip source-address 10.0.2.4 port 2000
```

```
!
ephone-dn  1
 number 1001
!
ephone-dn  2
 number 3001
!
ephone  1
 mac-address 00B0.6409.BDE5
 button  1:1
!
ephone  2
 mac-address 00B0.6409.CCC7
 button  1:2
```

Note The IOS feature IP Keyswitch has been renamed to Telephony Service. To get into telephony service configuration mode, type **telephony service** at the IOS global configuration mode. For more details, refer to the following URL:
http://www.cisco.com/univercd/cc/td/doc/product/access/ip_ph/ip_ks/ipkey2.htm

Configuring Cisco Call Manager

Note To register Callgen-simulated IP phones with the Cisco Call Manager (CCM), the **auto-registration** option of CCM needs to be turned off, so you need to manually configure the phone devices in CCM. The MAC address and DN of configured devices in CCM must match the MAC address and the DN (called-number) configured in Callgen.

Note If you need to configure many phones in CCM, you can use a utility called Bulk Administration Tool (BAT). You can search on the Cisco CCO page for more information, using the keyword "BAT."

SIP IP Phone Emulation

Since Callgen v4.1T, Callgen supports SIP IP phone emulation based on the IOS SRST code. Callgen can only register a called number. The major difference between SIP IP phone emulation and a SIP gateway call is that the SIP IP phone emulation requires a registration to a proxy server or a registrar server. Registration is not required for SIP gateway calls. If your SIP testing does not require registering a phone, you should configure the SIP gateway calls. For more details on generating SIP gateway calls, see Making Calls Using SIP (page 6-16).

Configuring SIP Phones

sip-endpt emulation

Enables the channel to do SIP phone registration.

sip-endpt server-ip *ip-address* [**port** *n*]

Configures the SIP proxy server (or IOS SIP SRST) IP and port information. If port is not specified, the default is 5060.

sip-endpt source-ip-address *ip-address*

Configures the source IP address of the emulated IP phone. Make sure that the Callgen router and the SIP server know how to reach each other. Otherwise, add any IP routing entries as necessary.

sip-endpt protocol {**tcp** | **udp**}

Configures the protocol used to set up a SIP call. If unconfigured, the default protocol is UDP.

```
Sample Callgen router configuration:
```

Note SIP phone registration is done without username/password authentication.

```
!IOS configuration
dial-peer voice 1 voip
 destination-pattern 6333
 session protocol sipv2
 session target ipv4:128.21.1.98

!Callgen configuration
channel 1 type voice mode originate
  calling-number 6333
  called-number 6222
  interface voip:1
  sip-endpt emulation
  sip-endpt server-ip  128.21.1.98
  sip-endpt source-ip-address 128.21.1.43
```

Configuring Cisco SIP SRST Registrar

Sample IOS configuration:

```
!
voice service voip
 sip
  registrar server expires max 600 min 60
!
voice register pool  1
 id network 128.21.1.0 mask 255.255.255.0
!
```

For details, see the IOS documentation on the CCO page:

http://www.cisco.com/en/US/partner/products/sw/iosswrel/ps5012/products_feature_guide09186a0080181116.html#24877

Note Image c3745-ipvoice-mz.123-7.9.T has been used in our testbed. c3660-js-mz does not work.

Registering and Unregistering SIP Phones

channel *n1* [- *n2*] **sip-register**

Registers a channel or a group of channels to the SIP proxy server/registrar server. If the registration is successful, the Callgen channel state changes from UNREG to INACTIVE.

channel *n1* [- *n2*] **sip-unregister**

Unregisters a channel or a group of channels from the SIP proxy server/registrar server. If the unregistration is successful, the Callgen channel state changes from INACTIVE to UNREG. A channel should be stopped first before attempting to unregister the channel from the server.

Example of SIP to SIP Call with 3-tone Path Confirmation

IOS configuration on the Callgen router

```
dial-peer voice 1 voip
 session protocol sipv2
 session target ipv4:128.21.1.60
 incoming called-number 1345
 dtmf-relay sip-notify
 codec g711ulaw
!
dial-peer voice 2 voip
 application callgen_voice_ctm
 session protocol sipv2
 session target ipv4:128.21.1.60
 incoming called-number 2678
 dtmf-relay sip-notify
 codec g711ulaw
!
```

IOS configuration of the SIP SRST Registrar router

```
voice service voip
 sip
  registrar server expires max 600 min 60
!
!
voice class codec 1
 codec preference 1 g711ulaw
!
voice register pool  1
 id ip 128.21.1.10 mask 0.0.0.0
 max registrations 76
 voice-class codec 1
!
voice register pool  2
 id ip 128.21.1.51 mask 0.0.0.0
 voice-class codec 1

interface FastEthernet0/0
 ip address 128.21.1.60 255.255.255.0
 no ip route-cache
 speed auto
 half-duplex
```

Callgen configuration

```
g3660-1(cgch_3_vo_o)#sh co

global type voice
  audio-file 1 flash:3tone_s.au


channel 1 type voice mode originate
  duration 30 seconds
  calling-number 1345
  called-number 2678
  path-confirmation type 3-tone-slope play 1
  interface voip:1
  sip-endpt emulation
  sip-endpt server-ip  128.21.1.60
  sip-endpt source-ip-address 128.21.1.51

channel 2 type voice mode terminate
  called-number 1345
  path-confirmation type 3-tone-slope play 1
  interface voip:1
  sip-endpt emulation
  sip-endpt server-ip  128.21.1.60
  sip-endpt source-ip-address 128.21.1.51

channel 3 type voice mode originate
  duration 20 sec
  calling-number 2678
  called-number 1345
  path-confirmation type 3-tone-slope play 1
  interface voip:2
  sip-endpt emulation
  sip-endpt server-ip  128.21.1.60
  sip-endpt source-ip-address 128.21.1.51

channel 4 type voice mode terminate
  called-number 2678
  path-confirmation type 3-tone-slope play 1
  interface voip:2
  sip-endpt emulation
  sip-endpt server-ip  128.21.1.60
  sip-endpt source-ip-address 128.21.1.51
```

Note If you want to do path-confirmatation type ping, you need to configure “dtmf-relay sip-notify” on the VoIP dial peers of both originate and terminate channels.

Notes For Generating SIP IP Phone Emulation Load Test

Since the VoIP interface does not use DSP resource, it depends on CPU resource to do both playing out a tone file and detecting the incoming tone. Different platforms have different limits on the max number of channels/calls that can be bearer at the same time. This limitation is from the IVR Playout limitation defined in IOS code. The following lists the theoretical IVR limits on the corresponding Callgen platforms. The same thing applies to any other kind of VoIP interface.

```
c3640  -- 30  (sys/machine/cisco_c3600.h)
c3660  -- 180 (sys/machine/cisco_c3600.h)
c3745  -- 120 (sys/src-7k-c3745/platform_c3745.c)
c7200  -- 600 (sys/machine/cisco_c7100.h)
              (Note that c7200 has two different versions, the later version has much
               more powerful CPU)
c5400  -- 480 (sys/src-7k-as5400/platform_as5400.c)
              (Note, there are also two version of 5400, the as5400, as5400HPX)
```

Note The built-in registration expiration time for the SIP IP phone emulation feature is 3600 seconds. Path-confirmation might fail if a channel has to re-register in the middle of an active call.

Interface Specifications

For voice channels, the interface is required (see Associating a Channel with an Interface (page 4-3)). Use the **show interface** command to see a list of valid choices and desired formats. The Callgen voice CTM is based on Symphony and is capable of placing calls on POTS, PRI, VoIP, and VoFR interfaces.

Note For VoXX calls, a dial peer must be specified through the interface command. The dial peer must have already been created as part of the router's running configuration.

The valid formats for interface specifications for the call types are:

Call Type	Format	Examples
POTS	**port:***slot*/*sub-unit*/*port*	interface port:0/0/0, interface port:0/1/0
PRI/CAS	**port:***slot*:*signalingchannel* where: slot = T1 controller slot # signaling channel = D for PRI, ds0-group number for CAS	interface port:0:1, interface port:1:D
POTS/PRI/CAS	**pots** This command lets a terminating channel answer a call from any interface.	interface pots
VoIP	**voip:***dial-peer tag*	voip:100
VoFR	**vofr:***dial-peer tag*	vofr:99

Call Disconnect Codes

The following are the cause codes you are most likely to see when you use the **show channel** *num* command. The output provides information on why the last call was disconnected. Any disconnect cause other than **NORM (16)** or **CALL_CLEARED (86)** that occurs while the channel is in the HOLD state is considered abnormal.

Result Code	Identifier	Explanation
1	UANUM	Unassigned number. No terminate channels have been configured for the called number.
16	NORM	Normal call completion.
17	BUSY	The terminate channel assigned to handle the call is busy handling another call.
21	REJECT	Call rejected.
47	NO_RESOURCE	Terminate channel is misconfigured or has not been started.
63	NOSV	No service available. VoIP sockets are unable to handle call load.
86	CALL_CLEARED	Call aborted by user.

Explanations for all call disconnect codes (as in 12.3T) can be found at
http://www.cisco.com/en/US/products/sw/iosswrel/ps5187/products_command_reference_chapter
09186a008017cfa4.html

CHAPTER 7

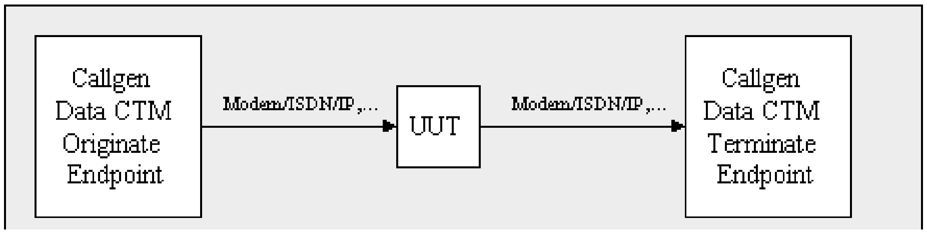
Data Call Type Module

The data CTM works in combination with IOS's Dial on Demand Routing (DDR) to provide modem, ISDN, VPDN Dialout, ProxyPPP, or other PPP/IP-based session control.

Typical Test Scenarios

Figure 7-1 depicts typical test scenarios involving the data CTM. In these scenarios, the Unit Under Test (UUT) could be anything from a single Network Access Server (NAS) to a complete VPDN/L2TP solution comprised of many components.

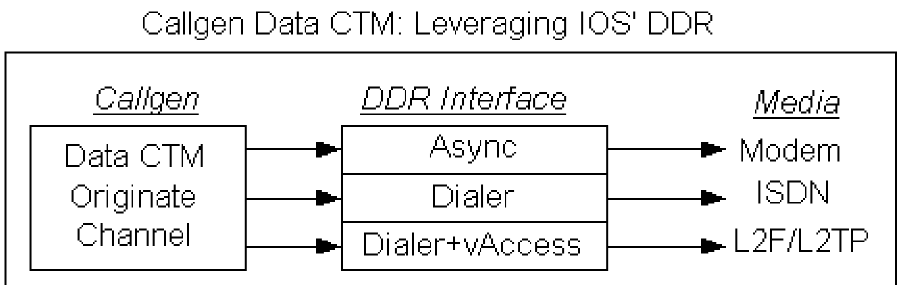
Figure 7-1 Typical Data CTM Test Scenarios


Callgen Data CTM Originate Endpoint
Modem/ISDN/IP,...
UUT
Modem/ISDN/IP,...
Callgen Data CTM Terminate Endpoint

Using the Data CTM

The data CTM requires separate originate and terminate Callgen routers. Like other Callgen modules, the terminate side must be started before the originate side. To initiate a call, the originate channel sends a UDP packet to the terminate channel via a specific dialer interface. That packet stimulates a DDR session, establishing IP connectivity between the Callgen originate and terminate channels. Figure 7-2 depicts how the originate endpoint can be used to drive various DDR interfaces.

Figure 7-2 Leveraging DDR interfaces



To keep the initial packet from getting discarded while the link is active, the dialer interface must be configured with a hold-queue size of 1.

To get more direct control over the call and the ability to get the link to drop when we want, the dialer interface is given a very brief idle timeout. We also specify a fast enable-timeout to re-enable the interface quickly.

Data CTM Configuration Parameters

Specifying the Source Address

source address *address*

Specifies the source address for the channel. On the originate channel, this is the address of the DDR interface. On the terminate end, loopback interfaces with secondary addresses work well enough.

Example:

```
clash-5300(callgen)# channel 101 source address 171.18.30.44
```

Specifying a Source Port

source port *number*

Specifies a different UDP port than the default (12321) for this channel. Note that the originating channel requires a destination port with the same port number. You should not need to change the source or destination port numbers unless some new functionality comes out that uses this port number.

The data CTM gobbles all packets sent to the specified UDP port and ignores all others. Only one source port number should be used on a Callgen router. To simplify the configuration of this parameter, use a Callgen class for assigning the port number to channels.

Example:

```
clash-5300(callgen)# channel 101 source port 12345
```

Specifying a Destination Address

destination address *address*

Specifies the target address for the originate channel. This command applies to originate channels only. The terminate channel should have a source address with the same ip-address.

Example:

```
clash-5300(callgen)# channel 101 destination address 171.18.30.44
```

Specifying a Destination Port

destination port *number*

Specifies a different UDP port than the default (12321) for the terminating channel. The terminating channel requires a source port with the same port number. You do not need to change the source or destination port numbers unless some new functionality uses this port number.

The data CTM gobbles all packets sent to the specified UDP port and ignores all others.

Example:

```
clash-5300(callgen)# channel 101 destination port 12345
```

Specifying the DDR Interface

dialer-interface *name*

Identifies the DDR interface on which to send packets out. This parameter is optional and is for originate channels only. If not specified, be sure to install a static route that directs the destination packet out the proper interface.

Example:

```
clash-5300(callgen)# channel 101 dialer-interface Async101
```

Defining the Packet Rate

frame rate *n*

Defines the rate at which data packets are generated by the channel. To generate no packets, use a rate of 0. The default rate for an originate channel is 20 per minute (1 packet every 3 seconds), and for a terminate channel it is 0.

Example:

```
clash-5300(callgen)# channel 101 frame rate 10 per second
```

Defining Packet Frame Size

frame size *bytes*

Defines the IP packet frame size (in bytes) for data packets. The size does not include the media encapsulation, which would be an additional 14 bytes per packet for Ethernet. The default frame size is 100 bytes.

Example:

```
clash-5300(callgen)# channel 101 frame size 50
```

Defining the Data Pattern

frame pattern

Defines the data pattern to use with data packets sent over the channel. The default pattern is zeros.

Example:

```
clash-5300(callgen)# channel 101 frame pattern random
```

Specifying Setup Timeout

setup-timeout *seconds*

Defines the maximum amount of time the channel waits for a call setup to complete. This parameter applies to originate channels only. The specified value should be longer than the maximum amount of time it will take for the DDR interface to establish the PPP session. The default is 45 seconds. Experimentation shows that modem calls typically complete in less than 22 seconds, and ISDN calls take less than 2 seconds.

Example:

```
clash-5300(callgen)# channel 101 setup-timeout 60
```

Specifying Teardown Timeout

teardown-timeout *seconds*

When an active call is terminated, either implicitly when the duration has expired or explicitly with the **stop** command, the Callgen channel sends a disconnect packets to the other endpoint. If the originating endpoint does not receive a disconnect ack packet from the terminating endpoint, the originating endpoint resends the disconnect packet every second. If the originating endpoint does not receive the disconnect ack packet before the terminate timeout expires, a failure is logged. The default is 5 seconds.

Example:

```
clash-5300(callgen)# channel 101 teardown-timeout 15
```

Specifying Intercall Delay

inter-call-delay *seconds* seconds

The inter-call-delay value should be greater than the sum of **dialer idle-timeout** and **dialer enable timeout**, which are configured under async interfaces. It is good practice to configure an appropriate inter-call-delay so that Callgen does not rush to make a modem call before the IOS dialer is ready for another call, in which case, setup fail might occur.

See Specifying the Time Between Calls (page 4-5) for more detaisl on inter-call-delay.

Example:

```
clash-5300(callgen)# channel 101 inter-call-delay 60 seconds
```

Specifying Automatic Increment of Source/Destination IP Addresses

You can individually assign each data channel a source and destination address. You can also automatically assign the addresses to the channels using the following commands. These commands can be used only on a class basis.

source-ip-base-mask *address mask*

Specifies the source base IP address for the data channels and the subnet mask. The address and mask must be in inet dot notation.

source-increment-step *step*

Specifies the increment step of the source IP addresses for the data channels. This provides more control over assigning the subsequent IP addresses. Step varies from 1 to 255. The default is 1.

dest-ip-base-mask *address mask*

Specifies the destination base IP address for the channels and the subnet mask. The address and mask must be in inet dot notation. This command applies to originate channels only.

dest-increment-step *step*

Specifies the increment step of the destination IP addresses for the data channels. This provides more control over assigning the subsequent IP addresses. Step varies from 1 to 255. The default is 1. This command applies to originate channels only.

For example:

```
cg-5300-3(callgen)#class oAsync type data mode originate
cg-5300-3(cgcc_oAsync_da_o)#source-ip-base-mask 111.1.0.1 255.255.255.0
cg-5300-3(cgcc_oAsync_da_o)#dest-ip-base-mask 113.1.0.1 255.255.255.0
cg-5300-3(cgcc_oAsync_da_o)#source-increment-step 1
cg-5300-3(cgcc_oAsync_da_o)#dest-increment-step 1
cg-5300-3(cgcc_oAsync_da_o)#show conf

class oAsync type data mode originate
  source-ip-base-mask 111.1.0.1 255.255.255.0
  source-increment-step 1
  dest-ip-base-mask 113.1.0.1 255.255.255.0
  dest-increment-step 1


cg-5300-3(callgen)#chan 1 - 5 class oAsync
cg-5300-3(cgch_1-5_da_o)#show conf

class oAsync type data mode originate
  source-ip-base-mask 111.1.0.1 255.255.255.0
  source-increment-step 1
  dest-ip-base-mask 113.1.0.1 255.255.255.0
  dest-increment-step 1

channel 1 class oAsync
  source address 111.1.0.1
  destination address 113.1.0.1

channel 2 class oAsync
  source address 111.1.0.2
  destination address 113.1.0.2

channel 3 class oAsync
  source address 111.1.0.3
  destination address 113.1.0.3

channel 4 class oAsync
  source address 111.1.0.4
  destination address 113.1.0.4

channel 5 class oAsync
  source address 111.1.0.5
  destination address 113.1.0.5
```

Configuring Path Confirmation for Modem Calls

Similar to voice calls, path confirmation can be performed for data calls. Proprietary UDP traffic is sent across the connected originate and terminate channels from time to time, until the call duration expires.

The path confirmation CLI for modem channels is very similar to voice channels:

path-confirmation type ping

This command is applicable for both originate and terminate channels. Once the path confirmation is configured, confirmation statistics are shown as part of the **show channel** statistics.

When path-confirmation type ping is defined on the originating channel, the channel's frame rate, frame size, and frame pattern configuration apply toward the ping packets. Setting the frame rate to non-zero, pauses the originating channel for the user-specified duration in between each packet. If the frame rate is set to zero, the channel sends the next ping packet as soon as it receives an acknowledgement for the last ping.

On the terminating channel, the channel's frame size and frame pattern configuration applies toward the ping-ack packets. Its frame rate configuration is ignored, since it does not apply toward path confirmation traffic.

Performing FTP Over Async/Dialer Cards

Until now there was no simple solution to generate modem/ISDN calls and perform FTP functionality over async/dialer interfaces. This solution is based on LNE FTPSE. In addition to the regular configuration to generate modem calls, you must configure certain FTP commands.

global type data [**mode** {**originate** | **terminate**}] **ftp** {**client** | **server**}

This is a global configuration mode. It is advised that you configure the FTP server on the same router as the terminate channel because it is necessary for the server to be started before the FTP client starts the process, since the terminate channel is the first to know that end-to-end connectivity is established.

The following commands are channel configurations:

ftp username *username*

Specifies the username required to authenticate the FTP session.

ftp password *password*

Specifies the password required to authenticate the FTP session.

ftp command {**get** | **put**}

Specifies the job type initiated by the client side.

ftp filename *filename*

Specifies the filename that is either sent to or from the FTP server.

ftp filesize *file_size*

Specifies the size of the file sent and stored on the FTP server. Required only when the command is **put**.

ftp xfer-size *xfer_size*

Required to be configured on the server side when the client is going to initiate the **get** command.

Generating TCP Traffic

By default, the data CTM generates UDP traffic to the terminate channel. To generate TCP traffic, you must enable it with the command **tcp-gen enable** command.

All TCP-related traffic options are assigned using the **tcp-gen** parameter. The following options are available.

tcp-gen frame size *#bytes*

Specifies the TCP message frame size in bytes to be sent from the endpoint (either originate channel or terminate channel). Valid size ranges are from 1 to 65535. The default is 100 bytes.

tcp-gen frame count *#*

Specifies the total count of TCP messages to send before the TCP connection can tear down. Valid number ranges are from 1 to 4294967295. The default is 0, which specifies that TCP messages are sent infinitely until the TCP connection tear down. For originate channels only.

tcp-gen frame rate *#frames* [**per** {**second** | **minute** | **hour**}]

Specifies the number of TCP messages to send within the specified time. The maximum rate is 65535 per second. The default is 10 frames per second. For originate channels only.

tcp-gen keepalive {**on** | **off**}

Enables and disables TCP keepalive mode. The default is off.

tcp-gen echo {**on** | **off**}

Enables and disables the terminate channel to echo the TCP message back to the originate channel, or enables and disables the originate channel to expect the terminate channel to echo messages to it. The default is on.

tcp-gen nagle {**on** | **off**}

Enables and disables the Nagle algorithm in the TCP connection. The default is on.

tcp-gen tos *hex_number*

Specifies the Type Of Service field in the TCP frame header. The default is on.

tcp-gen verify {**on** | **off**}

Enables and disables the verification of received TCP messages. Note that both originate and terminate channel always check for a sequence number and a magic number in the message header, even if **verify** is off. The default is on.

tcp-gen window *#bytes*

Specifies the endpoint's (originate or terminate channel) TCP window size in bytes. Valid size ranges are from 1 to 65535. The default is 4128 bytes.

tcp-gen connect port *dest_port*

Specifies the destination port number to which to connect. The destination IP address information is obtained from the channel's source address. You must configure **source address** before configuring this parameter; otherwise, an error message results. For originate channels only.

tcp-gen connect rate *#connect* [**per** {**second** | **minute** | **hour**}]

Specifies the number of TCP connections to make in the specified time. The maximum rate is 65535 connection per second, but actually you can only make two TCP connections per second because of the nature of TCP state sequence. For originate channels only.

tcp-gen connect count *#connect*

Specifies the number of TCP connections to make before stopping TCP traffic generation. Valid number ranges are from 1 to 65535. The default is 0, which specifies that there is only one TCP connection for the life of originate channel. For originate channels only.

tcp-gen connect start-time-delay *delay*

Specifies the delay before **tcp-gen** starts making a TCP connection to its destination channel. Valid range is from 1 to 65535. The default is 0, no delay. The delay time is measured from the time when the originate data channel receives the SETUP_ACK to the time when the data channel issues the TCP connect request. For originate channels only.

tcp-gen listen port *port_number*

Specifies the listener's port number information. For terminate channels only.

Modem Configuration Example

The following is an example of an originate Callgen configuration for a modem:

```
config t
username Async1 password cisco
interface Serial2:15
 no ip address
 encapsulation ppp
 dialer-group 1
 ! for 12.0 (or later) Network config only:
 isdn net
 !
 isdn incoming-voice modem
 no cdp enable
 no ip mroute-cache
 no logging event link-status
 no fair-queue
interface Async1
 ip address 111.1.0.1 255.255.255.0
 encapsulation ppp
 dialer in-band
 dialer enable-timeout 6
 dialer idle-timeout 2
 dialer string 5551111
 dialer hold-queue 1
 dialer-group 1
 ppp chap hostname async1
 async mode dedicated
 no peer default ip address
 no fair-queue
 no cdp enable
 no logging event link-status
 ppp authentication chap
end
callgen
 class oAsync type data
 duration 15
 inter-call-delay 10
 frame rate 90 per min
 frame size 256
channel 101 class oAsync
 source address 111.1.0.1
 destination address 113.1.0.1
 dialer-interface Async 1
```

The following is an example of a terminate Callgen configuration for a modem:

```
config t
username async1 password cisco
interface Group-Async1
 ip unnumbered Fastethernet 0
 encapsulation ppp
 async mode dedicated
 no logging event link-status
 no peer default ip address
 no fair-queue
 no cdp enable
 ppp authentication chap
 group-range 1 60
!
```

```
interface Loopback 3
 no logging event link-status
 ip address 113.1.0.1 255.255.255.0
end
callgen
 class tAsync type data mode terminate
 channel 101 class tAsync
  source address 113.1.0.1
```

ISDN Configuration Example

The following is an example of an originate Callgen configuration for ISDN:

```
config t
username isdn1 password cisco
interface Serial0:15
 no ip address
 encapsulation ppp
 isdn net
 isdn incoming-voice modem
 dialer pool-member 1
 max-link 1
 no cdp enable
 no ip mroute-cache
 no logging event link-status
 no fair-queue
interface Dialer1
 ip address 110.1.0.1 255.255.255.0
 encapsulation ppp
 dialer remote-name d1
 dialer idle-timeout 2
 dialer enable-timeout 2
 dialer string 5551101 class ISP1
 dialer pool 1
 dialer hold-queue 1
 dialer-group 1
 ppp chap hostname isdn1
 no fair-queue
 no cdp enable
 no logging event link-status
 ppp authentication chap
dialer-list 1
protocol ip permit
end
callgen
 class oIsdn type data
  duration 15
  inter-call-delay 3
  frame rate 10 per sec
  setup-timeout 5
  teardown-timeout 5
  frame size 256
 channel 1 class oIsdn
  source address 110.1.0.1
  destination address 112.1.0.1
  dialer-interface Dialer1
```

To prevent IOS from trying to bundle all the links together on the terminate side of the PPP connection, it is important that each dialer authenticate using a unique name. Use the command **ppp chap hostname** for this purpose.

The following is an example of a terminate Callgen configuration for ISDN:

```
config t
username isdn1 password cisco
interface Serial0:15
 ip unnumbered Fastethernet 0
 encapsulation ppp
 isdn incoming-voice modem
 dialer-group 1
 ppp authentication chap
 no peer default ip address
 no fair-queue
 no ip mroute-cache
 no cdp enable
 no logging event link-status
interface Loopback 1
 no logging event link-status
 ip address 112.1.0.1 255.255.255.0
end
callgen
 class tIsdn type data mode terminate
 channel 1 class tIsdn
 source address 112.1.0.1
```

Creating a Data CTM IOS Configuration

The following Callgen commands configure IOS for ISDN data and modem calls. This becomes the default IOS configuration. Depending on the testbed, you might need to change the IOS configuration.

Specifying the Host

[**no**] **host-access** *hostname*

Specifies the host that brings up the PPP session. This command configures IOS as follows: "username *hostname* password 0 cisco". This is required for the PPP CHAP authentication, which tries to match passwords on both hosts that participate in the PPP session. Using this command, you configure the host of the Callgen originate side and the host that participates in the PPP session. You can only configure two hostnames with the **host-access** command.

This is a global-mode command applicable to both originate and terminate sides.

In this example, the cg-5300-3 is trying to bring up a PPP session with cg-5300-4:

```
cg-5300-3(callgen)#global type data mode originate
cg-5300-3(cgset_da_o)#host-access cg-5300-3
cg-5300-3(cgset_da_o)#host-access cg-5300-4
```

If cg-5300-4 is the terminate side, the same commands are used in global mode, as follows:

```
cg-5300-4(callgen)#global type data mode terminate
cg-5300-4(cgset_da_o)#host-access cg-5300-3
cg-5300-4(cgset_da_o)#host-access cg-5300-4
```

Enabling Chat Scripts

[**no**] **auto-config-dialer enable**

Enables chat scripts called dial, map-class, and dialer-list. The dialer-list defines the interested traffic that will trigger the DDR call.

This is a global-mode, originate-only command.

For example:

```
cg-5300-3(callgen)#global type data mode originate
cg-5300-3(cgset_da_o)#auto-config-dialer enable
```

Specifying the Interface

[**no**] **config-dialer** {**Async** | **Dialer**} *interface_number* **Serial** *serial_number* **:** *timeslot_number*

Configures async or dialer interfaces as well as the Serial D channel interface to be used to bring up the PPP session. For ISDN data calls, specify **Dialer** followed by the dialer interface number. For modem calls, specify **Async** followed by the async interface number. *serial_number* is the serial interface number, and *timeslot_number* is the D channel time slot. For example, the time slot number is 15 for E1 and 23 for T1.

This is a channel-mode, originate-only command.

If you specify **Dialer**, this command also configures "username Isdn# password 0 cisco", where # represents the dialer interface number. If you specify **Async**, it configures "username Async# password 0 cisco".

Make sure that the channel is configured with a source address before issuing this command, because the same IP address is used to configure the dialer or async interface. The data CTM requires that the source IP address of a channel be the same as the IP address of the dialer or async interface.

In the following example, the user intends to use E1 0 and Dialer3, so interface Dialer3 and interface Serial0:15 are configured.

```
cg-5300-3(callgen)#chan 1 type data mode originate
cg-5300-3(cgch_1_da_o)#source address 112.2.2.2
cg-5300-3(cgch_1_da_o)#config-dialer Dialer 3 Serial 0 : 15
```

Specifying the Serial D Channel Interface

[**no**] **config-dialer-interface Serial** *serial_int_number* **:** *timeslot_number*

Configures the Serial D channel interface on the terminate side. This command configures "username Isdn# password 0 cisco". By default, it assumes 60 dialer interfaces.

When the **config-dialer-interface** command is used, **ip unnumbered FastEthernet0** is configured on the terminate side. Make sure that FastEthernet0 is configured with an IP address. If not, change it to the corresponding Ethernet interface. For example, **ip unnumbered Ethernet0**.

This is a global-mode, terminate-only command.

For example:

```
cg-5300-4(callgen)#global type data mode terminate
cg-5300-4(cgset_da_t)#config-dialer-interface Serial 0 : 15
```

Specifying Group Async Interface

[**no**] **config-Group-Async** *group_async_number* **range** *range_low range_high*

Configures the Group-Async interface on the terminate side. *range_low* and *range_high* are the Async interfaces that belong to the same group specified by *group_async_number*. The range should include the Async interfaces that have been configured on the originate Callgen. If any Async interfaces within that range are already a member of another Group-Async interface, an error message is displayed.

This is a global-mode, terminate-only command.

For example:

```
cg-5300-4(callgen)#global ty data mode terminate
cg-5300-4(cgset_da_t)#config-Group-Async 3 range 1 30
Building configuration...

cg-5300-4(cgset_da_t)
#
```

IOS Configuration Issues

Loopback Interface on the Terminate Channel

The command **source address** *IP_address* in a terminate channel creates a loopback interface in Terminate IOS, as follows:

```
cg-5300-3(callgen)#chan 1 type data mode terminate
cg-5300-3(cgch_1_da_t)#source address 113.3.3.3

interface Loopback10
 ip address 113.3.3.3 255.255.0.0 secondary
 ip address 100.100.100.100 255.255.255.255
 no logging event link-status
```

By default, it uses loopback interface 10 and assigns 100.100.100.100 to its primary IP address. Then it puts the channel's source address as a secondary IP address. If you have a couple of terminate channels, all the source IP addresses are added as secondary in loopback 10. This interface is removed only when the **clear configuration** command is used. If the channel is removed, the corresponding IP address is removed from the loopback interface.

ISDN Status

The ISDN status needs to be "network" on one side, and "user" on the other side. To check this, use the **show isdn status** command in IOS exec mode. In addition, the Layer 2 state needs to be in MULTIPLE_FRAME_ESTABLISHED.

To make one side "network" and to establish ISDN connectivity, use the following commands in the corresponding Serial D interface:

isdn net
shut
no shut

Making an ISDN Data Call

This example assumes that Callgen Originate and Callgen Terminate are connected back to back.

Originate side:

Step 1 In global mode, use the **host-access** command.

Step 2 In global mode, use the **auto-config-dialer enable** command.

Step 3 To configure channels, use the **config-dialer Dialer** command in channel mode. Make sure that the source IP address is configured for the channel before issuing this command.

Terminate side:

Step 1 In global mode, use the **host-access** command.

Step 2 In global mode, use the **config-dialer-interface** command.

Step 3 To configure channels, use the **config-dialer Dialer** command in channel mode. Make sure that the source IP address is configured for the channel before issuing this command.

Making a Modem Call

This example assumes that Callgen Originate and Callgen Terminate are connected back to back.

Originate side:

Step 1 In global mode, use the **host-access** command.

Step 2 In global mode, use the **auto-config-dialer enable** command.

Step 3 To configure channels, use the **config-dialer Async** command in channel mode. Make sure that the source IP address is configured for the channel before issuing this command.

Terminate side:

Step 1 In global mode, use the **host-access** command.

Step 2 In global mode, use the **config-Group-Async** command.

Step 3 To configure channels, use the **config-dialer Async** command in channel mode. Make sure that the source IP address is configured for the channel before issuing this command.

Data CTM Channel Status

The callgen **show channel** *num* command displays the following statistics and configuration information for the specified data channel:

```
5300-1(callgen)#show chan 1

Channel 1 Call Statistics
  elasped channel run time: 00w0d11:22:26.992 and counting
  channel state: IDLE
  setup attempts: 2125
  accepts: 2125
  confirms: 0
  setup-fails: 0
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  passed-calls: 2125
  failed-calls: 0
  setup rate: 3 calls per minute
  accept rate: 3 calls per minute
  setup time: min: 972ms, max: 1664ms, avg: 1245ms
  hold time: min: 15000ms, max: 15012ms, avg: 15000ms
  disconnect time: min: 12ms, max: 52ms, avg: 22ms
  idle time: min: 3000ms, max: 3068ms, avg: 3000ms
  packets sent: 53125
  bytes sent: 12665000
  missing receive data packets: 0 of 0
```

```
      channel state: Inactive
      Dialer successes: 2125, failures: 0, last: none
      tcp active time: 0ms
      tcp packets sent: 0
      tcp packets recevied: 0
      tcp bytes sent: 0
      tcp bytes received: 0
      tcp transmit rate: 0 bytes/sec
      tcp receiving rate: 0 bytes/sec
      tcp packets verify errors: 0
      tcp echo: on
      tcp state:
      tcp last error:
      tcp connect attempted: 0
      tcp connect errors: 0
      tcp connect reset: 0
      tcp connect timeout: 0
    Channel 1 Current Settings
      channel is of class oIsdn
      channel type is data
      channel mode is originate
      minimum call-to-call-delay is 0 seconds
      call duration is 15 seconds
      minimum inter-call-delay is 3 seconds
      start-to-start-delay is 0 seconds
      start-time-delay is 0 seconds
      dialer-interface Dialer1
      setup-timeout 10 seconds
      teardown-timeout 10 seconds
      frame size 256
      frame rate 90 per minute
      frame pattern zeros
      source address 110.1.0.1
      source port 12321
      destination address 112.1.0.1
      destination port 12321
```

“Packets” and “bytes sent” indicate how many packets were sent by this channel to the other channel. “Missing receive data packets” indicates how many of the data packets received by the interface were dropped vs. the total number of packets received by the interface.

For an explanation of some of the common statistics counters, see Displaying Channel Information (page 4-12).

Troubleshooting the Data CTM and Related DDR Configurations

The data CTM and DDR use a lot of IOS functionality. Aside from the basic router configuration and troubleshooting skills, the following DDR and related references can be of assistance:

- http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/dial_c/dcprt4/index.htm
- http://wwwin-eng.cisco.com/Eng/IOS/Comm_Server/SW_Unit_Functional_Specs/vpdndialout.fs.fm@1
- http://wwwin-eng.cisco.com/~ketaylor/voip/Callgen/dataCTM/SampleConfigs/

To help avoid problems, try starting with configurations that are known to be good. If you need to start from scratch, get one originate/terminate pair working before creating any remaining channels.

If the **show** command indicates failures, use the **show log** command to find out which channels failed:

```
5300-1(callgen)#show

Aggregate Call Statistics
  Elasped time of session: 00w0d00:02:29.292 and counting
  Originate Statistics
    active channels: 60 of 120
    setup attempts: 150
    accepts: 112
    confirms: 0
    setup-fails: 1
    aborts: 0
    abnormal disconnects: 0
    confirmed errors: 0
    other errors: 0
    ...

5300-1(callgen)#show log
CH155_12:39:50.751 (setup-error, Timeout waiting for setup): from state Setup, setup
30000 ms, duration 00w0d00:00:00.000, teardown:0 ms
```

DDR call setup failures can occur for numerous reasons. With modem calls, for example, the dialer could fail to find the required resources, the CHAT script could fail, or the PPP negotiation could fail. The failure indicated above was detected by the dialer:

```
5300-1(callgen)#show chan 155

Channel 155 Call Statistics
  elasped channel run time: 00w0d00:02:41.444 and counting
  channel state: IDLE
  setup attempts: 2
  accepts: 1
  confirms: 0
  setup-fails: 1
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  passed-calls: 1
  failed-calls: 1
  setup rate: 1 calls per minute
  accept rate: 22 calls per hour
  setup time: min: 21760ms, max: 21760ms, avg: 21760ms
  hold time: min: 15000ms, max: 15000ms, avg: 15000ms
  disconnect time: min: 152ms, max: 152ms, avg: 152ms
  idle time: min: 30000ms, max: 54000ms, avg: 42000ms
  packets sent: 26
  bytes sent: 5996
  missing receive data packets: 0 of 0
  channel state: Inactive
  Dialer successes: 1, failures: 1, last: success
  ...
```

This dialer statistic is based on the same information reported by the **show dialer int** command:

```
5300-1#show dialer int Async55

Async55 - dialer type = IN-BAND ASYNC NO-PARITY
Idle timer (2 secs), Fast idle timer (20 secs)
Wait for carrier (30 secs), Re-enable (6 secs)
Dialer state is data link layer up
Dial reason: ip (s=111.55.0.1, d=113.55.0.1)
Time until disconnect 1 secs
Current call connected 00:00:27
Connected to 5511155 (5300-2)
```

```
Dial String      Successes   Failures    Last called   Last status
5511155                  3          1    00:00:27       successful   Default
```

In the 12.0(3.0.3)T release, there are still many types of dial failures that are not captured by this statistic. For example, none of the CHAT or PPP-related failures that were generated during the data CTM verification process were detected by the dialer. To track the status of this issue, see CSCdm03256.

One particular failure was very easy to reproduce under bursty call conditions with modem calls. This failure manifested as an LPC timeout. To keep track of the status of this issue, refer to CSCdm08354.

Whether debugging hard or intermittent call setup failures, the only sure-fire technique at this point is to turn on debugs. For intermittent failures where you are generating lots of calls, be sure to configure a large enough logging buffer and disable console debug messages.

To watch for intermittent PPP failures, use the **start** command to run calls in the background. Regardless of the call type, turn on **debug ppp negotiation** and **debug dial**. For intermittent CHAT failures (modem only), use **debug chat**. While waiting for failures, you can use the Callgen **show** command periodically to poll the call statistics.

```
Router# config t
Router(config)# logging buffer 1048576
Router(config)# no logging console
Router(config)# end
Router# debug dial
Router# debug ppp negotiation
Router# debug chat
```

With hard failures, only attempt one call at a time with the **start** *ChannelNumber* **total-calls 1** command. Also, turn on **debug ppp authentication** to verify that part of the process is functional.

Fax Call Type Module

The fax CTM supports fax on the Voice Feature Card (Libretto project) and the T.37 Store and Forward Fax (Allegro) systems, and inherits their underlying functionality and capabilities.

For Allegro, the fax CTM supports origination and termination of calls on digital T1/E1 as well as analog FXS, FXO, and E&M interfaces. Two types of modem modules are supported: MICA and Microcom. MICA modems support fax transmissions only, while Microcom modems support both. The fax CTM requires at least one Ethernet interface.

For Libretto, fax origination and termination is available on both T1/E1 CAS and ISDN PRI. Libretto requires vcware 8.02, which is located in */tftpboot/libretto-test/Vcware/vcw-vfc-mz.c549.hc.8.02.bin*. It does not work with earlier vcware versions.

Some configuration parameters are common to both fax interface types; others are specific to either Allegro or Libretto.

Specifying the Fax Interface Type

You can run either Allegro or Libretto at a given time on a given router. To specify the fax interface type, use the following command at the IOS level:

fax interface-type fax-mail

For the AS5300 platform, use the following to set the interface type:

fax interface-type {**vfc** | **modem**}

To run the fax CTM in Libretto mode, specify **vfc**. To run the fax CTM in Allegro mode, specify **modem**.

Note To switch between Allegro and Libretto, you must first reload the router by issuing **write erase** and then **write mem** before running the **fax interface-type** command.

Common Fax CTM Configuration Parameters

The following parameters are common to both the Allegro and Libretto interface types. Some of the commands, as noted, are class-level commands (see Creating Classes (page 4-11)).

Specifying the Calling and Called Numbers

To establish a call, Callgen must know what number to dial as well as what number to associate with a channel. This is accomplished with the following two parameters:

called-number—For originate channels, this specifies the number called. For terminate channels, this specifies the number associated with the specified channel. This is a required parameter for both fax originate and terminate channels.

Example:

```
clash-5300(callgen)# channel 1 called-number 19192221212
```

calling-number—For originate channels only. Specifies the number you are calling from. Optional.

Example:

```
clash-5300(callgen)# channel 1 calling-number 2484646
```

Specifying Payload Transmission

To specify a fax payload to transmit, first use the following command to define the location and associate an index to it.

global type fax mode orig index *value* **path** *value*

For example, the following specifies a TIFF image file located in Flash to use for fax transmission. It is identified by the index value of 1. This is a global-level fax CTM command, in the sense that it is not associated with fax CTM channel or class commands. You can define different fax payloads, such as a single-page fax TIFF image, multipage fax TIFF image, or a simple text file.

```
global type fax mode orig index 1 path flash:page2.tif
```

After defining the global fax commands, you can assign a specific payload from the pool of fax payload definitions configured to an originate channel or class using the following command:

payload index *value*

For example, the following chooses the payload “flash:page2.tif” for all fax calls made through channel 1:

```
chan 1 payload index 1
```

Note You can download the fax payload files *page2.txt* and *page2.tif* from */autons/vts/sample/fax*.

Specifying Payload Reception

By default, incoming fax payload is not saved. To save it, first use the following command to define the location and associate an index to it. Callgen dynamically generates the filename consisting of the channel number, the num-th call on the channel, and a suffix of **.tif**

global type fax mode term index *value* **path** *value*

For example:

```
global type fax mode term index 1 path flash:
```

Then in the channel definition, use the following command to specify the index:

payload save index *number*

For example:

```
chan 1 payload save index 1
```

If you specify the payload save index 1 that is pointing to flash:, and you made five fax calls on channel number 1, you will see the following filenames listed in flash:

```
callgen-5300# show flash:
System flash directory:
File  Length   Name/status
1     30861    chan1_1.tif
2     30861    chan1_2.tif
3     30861    chan1_3.tif
4     30861    chan1_4.tif
5     30861    chan1_5.tif
```

Note Whenever channel 1 receives a fax payload, it is saved as "chan1_#.tif" in flash:. The # indicates the num-th of the call in the same channel. It should match the number of the **accepts** counter in the **show channel** command. This is to prevent the tif image from being overwritten by later calls on the same channel.

Limitation of IOS: The PCMCIA Flash used in the routers does not process simultaneous write requests to the Flash by queuing the requests. So you can save as many tif files as fax calls generated into the Flash from a single channel, but it would not be able to save all the tif files from multiple simultaneous calls in multiple channels into the Flash. This is only a limitation with the Flash. You can save multiple files from multiple simultaneous calls in multiple channels to TFTP.

Verifying Page Count

pl-verification type page-count *num-pages*

Checks whether the specified number of pages has been transmitted or received. It can be used with both originate and terminate channels.

Example:

```
pl-verification type page-count 2
```

Verifying Size

pl-verification type page-size *num-bytes*

Checks whether the received fax payload is the specified number of bytes. The value specified should be the size of the reference payload being used. Applies to fax terminate channels only.

Example:

```
pl-verification type page-size 23526
```

Specifying Size Tolerance

pl-verification size-tolerance *num-bytes*

This command works in conjunction with the **pl-verification type page-size** command. It allows a tolerance (in number of bytes) to be specified with reference to the actual payload size, within which the page size can be considered a success. Applies to fax terminate channels only.

Example:

```
pl-verification size-tolerance 50
```

Note To ensure that consistent fax images are received by the terminating Callgen fax channel, make sure that you are using either the *page2.tif* or *page2.txt* image file located in the directory */autons/vts/sample/fax/*. Also, if the T1 controller type is used, make sure that one end is configured as the clock source internal and the other end is configured as the clock source line.

Specifying the Starting Called Number by Class

start-called-number *number*

Generates called numbers for the channels of a class. Each channel gets a called number starting from the specified point, which is incremented by 1 or as specified with the **called-increment-step** command (see Specifying the Increment for Generating Called Numbers by Class (page 8-5)).

Note Libretto requires a larger number of dial peers: two dial peers are created per channel and for a Libretto fax class, and two class-level dial peers are created for all subsequent channels.

When using class-level dial peers, design your called numbers so that they do not overlap, because Callgen does not take care of overlapping called numbers.

Example:

```
class f type fax mode originate
  start-called-number 400

cg-5300-6(callgen)# chan 1 - 2 class f
channel 1 class f
  called-number 400

channel 2 class f
  called-number 401
```

Specifying the Starting Calling Number by Class

start-calling-number *number*

Generates calling numbers for the originate channels of a class. Each channel gets a calling number starting from the specified point, which is incremented by 1 or as specified with the **calling-increment-step** command (see Specifying the Increment for Generating Calling Numbers by Class (page 8-5)).

Example:

```
class f type fax mode originate
  start-calling-number 800

cg-5300-6(callgen)# chan 1 - 2 class f
channel 1 class f
  calling-number 800

channel 2 class f
  calling-number 801
```

Specifying the Increment for Generating Called Numbers by Class

called-increment-step *step*

Specifies the increment for generating the next called number for the channels in a class.

Example:

```
class f type fax mode originate
  start-called-number 400
  called-increment-step 4

cg-5300-6(callgen)# chan 1 - 2 class f
channel 1 class f
  called-number 400

channel 2 class f
  called-number 404
```

Specifying the Increment for Generating Calling Numbers by Class

calling-increment-step *step*

Specifies the increment for generating the next calling number for the channels in a class.

Example:

```
class f type fax mode originate
  start-calling-number 400
  calling-increment-step 4

cg-5300-6(callgen)# chan 1 - 2 class f
channel 1 class f
  calling-number 400

channel 2 class f
  calling-number 404
```

Using Libretto

This section lists the commands specific to Libretto as well as how to configure Libretto on Cisco IOS.

Callgen Configuration Commands

The following Callgen configuration commands are specific to Libretto.

Specifying the Interface Port

interface *port*

Originate channels require an interface configuration for sending faxes.

Example:

```
interface port:0:D
```

Terminate channels that use digital T1 interface do not require an interface port and receive faxes on any available interface. However, terminating channels that use an analog interface, such as FXO, FXS, or E&M, do require interface configuration:

Example for analog FXO, FXS, and E&M interfaces only:

```
interface port:0:D
```

Specifying the Payload Image Encoding Type

payload image encoding *encode_type*

Specifies the type of compression encoding scheme to use for Libretto fax transmission. The options are modified huffman (**mh**), modified read (**mr**), and modified modified read (**mmr**).

Example:

```
payload image encoding mmr
```

Specify Payload Image Resolution

payload image resolution *resolution_type*

Specifies the resolution for transmitting and receiving faxes. The options are **fine** and **standard**.

Example:

```
payload image resolution fine
```

Configuring Libretto on Cisco IOS

Step 1 On both the Callgen fax originate and terminate routers, enter:

```
mta receive aliases callgen_fax.com
```

Step 2 On both the Callgen fax originate and terminate routers, enter:

```
mta send server ip-addr
```

The IP address must be that of the local Ethernet interface.

Step 3 On the router running Libretto, enter:

```
fax interface-type fax-mail
```

For the AS5300 platform, enter:

```
fax interface-type vfc
```

Step 4 Some interactive voice response (IVR) TCL applications for Libretto uses for processing need to be specified in the IOS configuration. Define the following IVR scripts exactly as follows on both the originate and terminate routers:

```
call application voice cgen_libretto_onramp path
call application voice cgen_libretto_offramp path
```

path specifies the location of the on-ramp and off-ramp TCL. For example, *tftp://10.0.2.10/libretto-test/t37_onramp13.tcl*.

Note You can obtain the scripts from */tftpboot/libretto-test/Scripts/*. You can use *t37_onramp13.tcl* for **cgen_libretto_onramp**, and *t37_offramp.0.0.6.workaround.tcl* for **cgen_libretto_offramp**.

Libretto Debug Commands

Debugging ESMTP Messages

debug mta {**send** | **receive**}

Debugs the ESMTP messages exchanged during call setup.

Debugging T30 and T38 Protocol Messages

debug fmsp {**send** | **receive**} {**t30** | **t38**}

Debugs T30 and T38 protocol messages exchanged during fax transmission.

Debugging Off-ramp and On-ramp Messages

debug foip {**off-ramp** | **on-ramp**}

Debugs the off-ramp (fax origination) and on-ramp (fax termination) message exchanges.

Debugging Mail Service Interface Provider Messages

debug mspi {**send** | **receive**}

Debugs the Mail Service Interface Provider messages exchange.

Displaying Libretto Channel Status

The Callgen **show channel** command displays the following statistics and configuration information for the specified fax channel for Libretto:

```
cg-5300-6(callgen)#sh chan 1

Channel 1 Call Statistics
  elapsed channel run time: 00w0d00:00:30.512
  channel state: INACTIVE
  setup attempts: 5
  accepts: 5
  confirms: 0
  setup-fails: 0
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  passed-calls: 5
  failed-calls: 0
  setup rate: 472 calls per hour
  accept rate: 472 calls per hour
  setup time: min: 0ms, max: 0ms, avg: 0ms
  hold time: min: 30232ms, max: 30512ms, avg: 30344ms
  disconnect time: min: 0ms, max: 0ms, avg: 0ms
  idle time: min: 0ms, max: 0ms, avg: 0ms
  last disconnect cause: 1 - Normal disconnect
  Tx payload type text
  Tx Pages/Tx Bytes: 6/52917
  Rx Pages/Rx Bytes: 0/0
  pl-verify-pass: 0
  pl-verify-fail: 0
```

```
Channel 1 Current Settings
  channel type is fax
  channel mode is originate
  minimum call-to-call-delay is 0 seconds
  call duration is 0 seconds
  minimum inter-call-delay is 0 seconds
  start-to-start-delay is 0 seconds
  start-time-delay is 0 seconds
  reqd-call-to-call-delay is 0 seconds
  calling-number is not configured
  called-number is 456
  payload index 1
  pl-verification type not configured
  interface is port:0:D
  modem speed used: 14400
  fax call duration: 22 secs
```

Note For an explanation of some of the common statistics counters, see Displaying Channel Information (page 4-12).

Quality of Fax

Note This section only applies to Callgen 3.3T and earlier.

The quality of fax feature is only available for Libretto (fax on vfc) and not for Allegro (fax on modem). The main focus is on T30 signaling protocol abnormalities.

Note To use this feature, you must be familiar with T30 protocol messaging. For information about the T30 protocol, see http://wwwin-eng.cisco.com/Standards/WWW/ITU/pdf-files/t/t30.pdf. For Libretto's T30 implementation, refer to ENG-49930. For ITU's QoF documents refer to E.452.

In the T30 state machine implemented by Libretto, if there is a signaling problem, additional messages are exchanged to resolve the situation. Callgen tracks the DIS, DCS, FTT, RTP and RTN messages. The message exchanges indicate that the network is having a problem (refer to ENG-112383 for implementation and design details). For example, if an FTT is transmitted, it means that the terminate fax machine is having a problem training at the modem speed.

Similarly, Libretto has a powerful feature of lowering modem speed if the fax machines are having a problem training at the same speed. For example, if an FTT is received resulting in a subsequent modem speed reduction, Callgen keeps track of the reductions and at what speed the eventual transmission took place.

Callgen automatically displays this information in the **show channel** output for the particular channel if there was a problem in quality of fax. For example:

```
Channel 1001 Current Settings
  channel is of class LiOr
  channel type is fax
  channel mode is originate
  minimum call-to-call-delay is 0 seconds
  call duration is 0 seconds
  minimum inter-call-delay is 10 seconds
  start-to-start-delay is 60 seconds
```

```
 start-time-delay is 0 seconds
 reqd-call-to-call-delay is 0 seconds
 calling-number is not configured
 called-number is 12341001
 payload index 1
 pl-verification type page-count 2
 interface is port:3:D
 modem speed used: 14400
 fax call duration: 63 secs
 FTT count: 5
 DCS count: 2
 Modem Reduction count: 5
 Reduced Modem Speed: 2400
```

Configuring Allegro on Cisco IOS

Step 1 On both the Callgen fax originate and terminate routers, enter:

```
mta receive aliases callgen_fax.com
```

Step 2 On both the Callgen fax originate and terminate routers, enter:

mta send server *ip-addr*

The IP address must be that of the local Ethernet interface.

Step 3 On the router running Allegro, enter:

```
fax interface-type modem
```

Allegro ISDN, Modem, and Dialer Configurations

The following are excerpts of ISDN, modem and dialer configurations.

```
isdn switch-type primary-5ess
!
controller T1 0
 framing esf
 linecode b8zs
 pri-group timeslots 1-24
!
interface Serial0:23
 no ip address
 no ip directed-broadcast
 encapsulation ppp
 no ip route-cache
 dialer rotary-group 1
 dialer-group 1
 isdn switch-type primary-5ess
 isdn incoming-voice modem
 no fair-queue
!
interface Group-Async1
 ip unnumbered FastEthernet0
 no ip directed-broadcast
 encapsulation ppp
 no ip route-cache
 ip tcp header-compression
 no ip mroute-cache
 async default routing
 async dynamic address
 async mode interactive
 no peer default ip address
 no cdp enable
 group-range 1 48
 hold-queue 10 in
!
interface Dialer1
 ip unnumbered Loopback0
 no ip directed-broadcast
 encapsulation ppp
 no ip route-cache
 no ip mroute-cache
 dialer in-band
 dialer idle-timeout 300
 dialer-group 1
 peer default ip address pool def
 no fair-queue
 ppp authentication chap
 ppp multilink
!
line 1 48
 autoselect ppp
 modem InOut
 modem autoconfigure type microcom_hdms
 rotary 1
 transport input all
!
```

Displaying Allegro Channel Status

The Callgen **show channel** command displays the following statistics and configuration information for the specified fax channel for Allegro:

```
5300-1(callgen)#show chan 1
Channel 1 Call Statistics
  elapsed channel run time: 00w0d00:00:45.536
  channel state: INACTIVE
  setup attempts: 2
  accepts: 2
  confirms: 0
  setup-fails: 0
  aborts: 0
  abnormal disconnects: 0
  confirmed errors: 0
  other errors: 0
  passed-calls: 2
  failed-calls: 0
  setup rate: 158 calls per hour
  accept rate: 158 calls per hour
  setup time: min: 0ms, max: 0ms, avg: 0ms
  hold time: min: 45532ms, max: 46008ms, avg: 45770ms
  disconnect time: min: 0ms, max: 0ms, avg: 0ms
  idle time: min: 0ms, max: 0ms, avg: 0ms

  last disconnect cause: 1 - Normal disconnect
  Tx payload type tiff
  Tx Pages/Tx Bytes: 6/65558
  Rx Pages/Rx Bytes: 0/0

Channel 1 Current Settings
  channel type is fax
  channel mode is originate
  minimum call-to-call-delay is 0 seconds
  call duration is 0 seconds
  minimum inter-call-delay is 0 seconds
```

```
      start-to-start-delay is 0 seconds
      start-time-delay is 0 seconds

      calling-number is not configured
      called-number is 5718507
      payload index 1
```

Placing Fax Calls on Analog Interfaces

In addition to the digital interfaces, the Libretto project supports analog interfaces on 36XX platforms from 12.2(8)T label. Leveraging on this feature, the Callgen fax CTM supports the analog interfaces for FXS, FXO, and E&M ports to place fax calls. The Callgen configuration required to place the fax calls remains the same, and all the commands are also supported. However, to be able to terminate fax calls on an analog port, you must configure the interface with the corresponding port as part of the fax channel configuration. The interface configuration is not required when the fax call is terminate on the digital port.

Note Although the port is configured in the fax channel, the call is still processed and handled by the Libretto application, and the fax is received by the Callgen terminate channel through the store-and-forward mechanism.

A sample configuration looks something like this on the terminate side:

```
channel 1 type fax mode terminate
  called-number 1234
  payload save index 1
interface port:6/0/0
```

The dial-peers created by the fax CTM in IOS based on the above configuration is:

```
dial-peer voice 10000 pots
 application CALLGEN_FAX_CTM
 incoming called-number 1234
 group 1
 direct-inward-dial
 port 6/0/0
 prefix 1234
!
dial-peer voice 10001 mmoip
 application fax_on_vfc_onramp_app out-bound
 group 1
 destination-pattern 1234
 information-type fax
 session target mailto:$d$@callgen_fax.com
```

One of the key differences in the POTS dial-peer created for analog interfaces compared to digital interfaces is the configuration of the port and prefix. The port is configured to pick the corresponding POTS dial-peer for an incoming fax call based on the port information. Also note that CALLGEN_FAX_CTM is the application configured in the POTS dial-peer instead of cgen_libretto_onramp. CALLGEN_FAX_CTM adds the prefix to the called number and hands off the call to the Libretto on-ramp TCL script for setting up the ESTMP call leg.