Dialogic®

Dialogic® Audio Conferencing API

Demo Guide

April 2008

05-2290-005

Publication Date: April 2008

Document Number: 05-2290-005

Contents

Tables

Revision History

This revision history summarizes the changes made in each published version of this document.

Document No.	Publication Date	Description of Revisions
05-2290-005	April 2008	Made global changes to reflect Dialogic brand and changed title to “Dialogic® Audio Conferencing API Demo Guide.”
05-2290-004	June 2005	Section 2.1, “Hardware Requirements”, on page 13: In Requirements for Linux, changed Red Hat kernel version to 2.6.8.1. Section 4.1, “Starting the Demo”, on page 17: Changed Step 3 to read “Make an IP call into the system using two IP endpoints.
05-2290-003	April 2005	Section 1.1, “Overview”, on page 9: Changed location of demo file. Changed name of demo executable file. Added note about running demo using channel numbers higher than 120. Section 1.2, “Features”, on page 9: Removed “RAS (Gatekeeper) support via Global Call API” as a feature demonstrated by the demo. Section 4.1, “Starting the Demo”, on page 17: Changed name of demo executable file.
05-2290-002	September 2004	Section 1.1, “Overview”, on page 9: Revised overview to be operating system independent. Section 2.1, “Hardware Requirements”, on page 13: Added Linux information to Hardware Requirements section. Section 4.1, “Starting the Demo”, on page 17: Added Linux information in Step 1 of procedure. Section 4.2, “Using the Demo”, on page 17: Revised Table 1 to include Linux information. Section 4.3, “Stopping the Demo”, on page 19: Added Linux information.
05-2290-001	March 2004	Initial version of document.

About This Publication

The following topics provide information about this publication:

- Purpose
- Applicability
- Intended Audience
- How to Use This Publication
- Related Information

Purpose

This publication describes the Dialogic® Audio Conferencing API demonstration program for the Dialogic® Host Media Processing (HMP) Software and provides instructions for running the demo program.

Applicability

This document version (05-2290-005) is published for Dialogic® Host Media Processing Software Release 3.0WIN and for Dialogic® Host Media Processing Software Release 3.1LIN.

This document may also be applicable to other software releases (including service updates) on Linux and Windows® operating systems. Check the Release Guide for your software release to determine whether this document is supported.

Intended Audience

This publication is intended for:

- Distributors
- System Integrators
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)

How to Use This Publication

Refer to this publication after you have installed the Dialogic® HMP Software, which includes the Dialogic® Audio Conferencing API software.

This publication assumes that you understand computer telephony terms and concepts, and that you are familiar with the Windows® or Linux operating system and the C programming language.

The information in this guide is organized as follows:

- Chapter 1, "Demo Description" provides a brief overview of the Dialogic® Audio Conferencing API demo.
- Chapter 2, "System Requirements" discusses the requirements for running the demo.
- Chapter 3, "Preparing to Run the Demo" lists the tasks to perform before running the demo.
- Chapter 4, "Running the Demo" describes the steps required to run the demo, the demo options, the various modes of demo operation, and how to stop the demo.
- Chapter 5, "Demo Details" provides additional information about the demo, such as the files used by the demo.

Related Information

See the following for additional information:

- *http://www.dialogic.com/manuals/* (for Dialogic® product documentation)
- *http://www.dialogic.com/support/* (for Dialogic technical support)
- *http://www.dialogic.com/* (for Dialogic® product information)

Demo Description 1

This chapter includes the following information about the Dialogic® Audio Conferencing API Demo for Dialogic® Host Media Processing (HMP) Software:

1.1 Overview

The Dialogic Audio Conferencing API Demo is a simple audio conferencing application that is implemented using the Dialogic® HMP Software. The Dialogic Audio Conferencing API Demo directly supports H.323 and SIP call control signaling protocols through use of the Dialogic® Global Call API.

The Dialogic Audio Conferencing API Demo application is written in asynchronous mode, using a single process, single thread. Events are handled using polled mode, where the sr_waitevt() function is called and exits when an event is available. Even though a single-threaded approach may not be optimal for high-density applications, this model was chosen to simplify programming and to make the implementation more obvious.

Conferencing features are accessed using the Dialogic® Audio Conferencing API. The Dialogic® Global Call API is used for implementing call control and the Dialogic® Voice API is used for basic voice functionality.

For Windows®, the Dialogic Audio Conferencing API Demo is located in:

```
$(INTEL_DIALOGIC_DIR)\demos\Conferencing\ConferencingDemo\Release\
```

For Linux, the Dialogic Audio Conferencing API Demo is located in:

```
$(INTEL_DIALOGIC_DIR)/demos/Conferencing/ConferencingDemo/Release/
```

Note: Attempts to run the demo using channel numbers higher than 120 will cause the demo to fail.

1.2 Features

The Dialogic Audio Conferencing API Demo demonstrates the following features:

- Call Control:
  – Programming Interface: Dialogic® Global Call API
  – Supported Call Control Protocols: SIP and H.323

- – DTMF mode: In Band (H.323), InBand & RFC 2833 (SIP)
  - – Number of simultaneous calls: Limited by the license used
- Media:
  - – Programming Interface: Dialogic® R4 Voice API
  - – Audio Codec: G.711mu-Law 10, 20 or 30 ms frame size
  - – Play and Record capabilities
  - – Detection and generation of DTMF digits
- Conferencing:
  - – Programming Interface - Dialogic Audio Conferencing API
  - – Active Talker detection
  - – DTMF detection
  - – DTMF Clamping (optional)
  - – Echo Cancellation (optional)
  - – Monitoring (via receive-only audio recorder)
  - – Setting or retrieving conference attributes on a conference in progress

1.3 Application Interface and Configuration

The application provides a convenient console interface which provides a view into the application performance and system statistics. Console command capabilities include:

- Start and stop the monitor for a specific conference
- Identify the active talkers
- Update the conference configuration by reloading the configuration file, without restarting the application.
- Change verbosity of the information

At start-up, the application reads the *conf_demo.cfg* configuration file to pre-set some common parameters. This file must be located in the same directory as the application executable file.

All parameters are optional. If some parameters are missed, or the file cannot be found in the default location, the application will use a default parameter set, hard-coded in the program code.

The following parameters may be set at start-up:

- Application-Level parameters:
  - – The number of simultaneous calls
  - – The path and names for "Welcome", "Bad passcode", "Call Later" and "Goodbye" audio files
  - – The name and path for the application log file
  - – Maximum size of the log file
  - – The log file verbosity
- Board-Level parameters
  - – Active Talker feature (enable / disable)

  - Active Talker detection interval
  - DTMF Clamping (enable / disable)
  - DTMF digit to mute / un-mute participants
- Single Conference Parameters:
  - The passcode
  - DTMF detection while in a conference (enable/disable)
  - Echo Canceller (on/off)

1.4 Call Logging

All events, messages, and API calls are saved in a text file, *HmpDemo.log*, as the application is running. Maximum size of the log file and logging level may be set via the application configuration file. When the file size exceeds the maximum size, the logging wraps to the beginning of the file and the previously logged information is overwritten.

System Requirements 2

The requirements for running the Dialogic® Audio Conferencing API Demo for Dialogic® Host Media Processing (HMP) Software are described in this chapter.

2.1 Hardware Requirements

Requirements for Windows

The hardware requirements for running the Dialogic Audio Conferencing API Demo using the Windows® operating system include:

- Machine using the Windows® 2000, Windows® XP, or Windows® 2003 operating system
- A standard network interface card (NIC)
- A minimum of two IP telephones
- Local Area Network

Requirements for Linux

The hardware requirements for running the Dialogic Audio Conferencing API Demo using the Linux operating system include:

- Machine using the Red Hat Enterprise Linux Advanced Server 3.0, Update 1 operating system with kernel version 2.6.8.1
- A standard network interface card (NIC)
- A minimum of two IP telephones
- Local Area Network

2.2 Software Requirements

The software requirements for running the Dialogic Audio Conferencing API Demo include:

- Dialogic® Host Media Processing (HMP) Software
- A runtime Dialogic® HMP Software license that supports the conferencing feature

Preparing to Run the Demo 3

This chapter provides information about the preparations required before running the Dialogic® Audio Conferencing API Demo for Dialogic® Host Media Processing (HMP) Software and includes the following sections:

3.1 General

Before running the Dialogic Audio Conferencing API Demo, check that the system requirements identified in Chapter 2, “System Requirements” have been adhered to, and that the Dialogic® HMP Software system service has been started. Refer to the appropriate Installation Guide for information about starting the system service.

3.2 Connecting to External Equipment

You will need to connect both the PC running the Dialogic® HMP Software and the IP telephones to an Ethernet LAN.

3.3 Editing the Configuration File

The configuration file for the demo code, *conf_demo.cfg*, is located in the same directory as the demo executable file and contains the following syntax (spaces and tabs are ignored):

Note: Individual volume control is not supported.

```
*****************************************************************
*   Configuration file for HMP Conferencing Demo                *
*   Comment lines use #, !, or *.                               *
*****************************************************************

[Common]
NumberOfChannels = 4        ! Max number of IP channels to use. Default = all available channels
MaxLogSize = 100000         ! Max Number of lines in a log file. Default =10000
PrintLevel = 1              ! Printout verbosity. 0-All, 1-App,  2-Events, 3-Warnings,  4-Errors
LogFileName = HmpDemo.log   ! Default = HmpDemo.log
WelcomeFileName = welcome.pcm
BadPasscodeFileName = invalid.pcm
CallLaterFileName = bye.pcm
GoodByeFileName = bye.pcm

# NOTE: Setting PrintLevel = 0 may affect performance if demo is used under heavy load.
```

```
[Board Parameters]
 ActiveTalker = Yes
 ATInterval = 5             ! Active talker interval in 100 millisecond units
 DTMFClamping = Yes
 SIP = Yes
 MuteDigits = *6

# NOTE: Individual volume control is not supported under HMP 1.1 BETA.
 VolumeControl =Yes         ! NOTE: Individual volume control is not supported.
 VolumeUp = 3
 VolumeDown = 9
 VolumeReset = 0

# Conference Information Sections
# Header Format: [Conference xxx], where xxx = a unique decimal number (bridge number)
# Passcode field is mandatory, and must be unique across the file, as well as a conf ID.
# All other values are set to "No" by default.

[Conference 12]
 Passcode = 12345
 DetectDigits = No          ! Triggers DCBEV_DIGIT event notification
 EchoCanceler = No

[Conference 6]
 Passcode = 34567
 DetectDigits = No
 EchoCanceler = No

[Conference 4]
 Passcode = 12347

[Conference 7]
 Passcode = 569
 DetectDigits = Yes
 EchoCanceler = No
```

If you edit this file, save it in the same directory as the application's executable file.

Running the Demo 4

Information about running the Dialogic® Audio Conferencing API Demo for Dialogic® Host Media Processing (HMP) Software is provided in the following sections:

4.1 Starting the Demo

To run the Dialogic Audio Conferencing API Demo, follow this procedure:

1. Open a command prompt and go to the directory where the demo is located.
   - For Windows®, the directory is:
     `$(INTEL_DIALOGIC_DIR)\demos\Conferencing\ConferencingDemo\Release\`
   - For Linux, the directory is:
     `$(INTEL_DIALOGIC_DIR)/demos/Conferencing/ConferencingDemo/Release/`
2. At the command prompt:
   - For Windows®, execute the *ConferencingDemo.exe* file.
   - For Linux, execute the *./ConferencingDemo* command.
3. Make an IP call into the system using two IP endpoints.
4. When prompted, enter the passcode number (one of the digit strings from the demo configuration file described in Chapter 3, “Preparing to Run the Demo”, and if the number is valid, you will be placed into a conference.

   Note: You can place your call into the system via IP using the default Passcode 12345.
5. Use a function key to access the application's features. See Section 4.2, “Using the Demo”, on page 17 for a description of the function keys.

   Note: To mute or un-mute a phoneset while in a conference, press '*6'. The default digits may be changed via the demo configuration file as shown in Chapter 3, “Editing the Configuration File”.

4.2 Using the Demo

You can choose several function key features when running the Dialogic Audio Conferencing API Demo as shown in Table 1.

Table 1. Dialogic® Audio Conferencing API Demo Function Keys

Function Key	Description
F2	Updates configuration, passcode information, and conference features while the application is running. User needs to edit the configuration file, save the file and then press the F2 key. The following parameters may be updated at run time: • Application-Level Parameters: – The path and names for "Welcome", "Bad Passcode", "Call Later" and "Goodbye" audio files. – The log file and console information verbosity • Board-Level Parameters: – DTMF digit to mute/un-mute participants • Single Conference Parameters: – DTMF detection while in a conference (enable / disable) – Echo Canceller (On/Off) ***Note:*** Changing passcode/bridge number information will not affect any conference in progress, but will take effect when the updated passcode is dialed by a new caller.
F3	Retrieve Active talker information
F4	Depending on the operating system: • Windows®: Change log and print level (0 = All messages, 1 = API calls, 2 = Events, 3 = Warnings, 4 = Errors, 5 and higher = None) • Linux: Stop the Dialogic® HMP Software Conferencing Demo and exit the application
F5	Display application statistics. This option will show how many conferences are currently in progress, how many participants are in each conference, and how many conferencing resources are currently available.
F6	Start monitoring. User will be prompted to enter the bridge number of a conference to be monitored (the number may be found in the upper console window or via F5 option). After the number is entered, the application will add a media resource (if one is available) in receive-only mode to the conference, and will start recording a file with the name Conf_xxx.pcm, where xxx is a conference bridge number.
F7	Stops recording and removes the monitor from the conference.
F8	Terminates a single conference. User will be prompted to enter a bridge number to terminate.
F9	Terminates all conferences. This option will stop all conferences that are currently in progress and free all resources.

4.3 Stopping the Demo

To stop the Dialogic Audio Conferencing API Demo and exit the application:

- For Windows®, press the F10 key.
- For Linux, press the F4 key.

Demo Details 5

The following sections provide further details about the Dialogic® Audio Conferencing API Demo for Dialogic® Host Media Processing (HMP) Software.

5.1 Files Used by the Demo

Table 2 lists the files used by the Dialogic Audio Conferencing API Demo.

For Windows®, these files are located in:

```
$(INTEL_DIALOGIC_DIR)\demos\Conferencing\ConferencingDemo\Release\
```

For Linux, these files are located in:

```
$(INTEL_DIALOGIC_DIR)/demos/Conferencing/ConferencingDemo/Release/
```

Table 2. Files Used by the Dialogic® Audio Conferencing API Demo

File Name	Purpose
conf_demo.cfg	The configuration file is used to preconfigure specific common parameters.
HmpDemo.log	The log file is an ASCII text that logs the results of the Conferencing demo run.
welcome.pcm	“Welcome” audio file
bye.pcm	“Goodbye” audio file
invalid.pcm	“Bad Passcode” audio file
bye.pcm	“Call Later” audio file

5.2 Classes and Objects

The Dialogic Audio Conferencing API Demo application defines and uses the following C++ classes:

- **CAppLog** - provides the interface for application logging.

- **CConsoleIo** - (Windows® only) provides a set of console input/output operations. This class contains two independently scrolled windows, a static menu bar and user input space. It allows a user to dynamically build a menu and asynchronously wait for a menu key or user input. The user input interactions use two callback functions which are invoked by pressing a function key (menu choice) or the ENTER key (user input).
- **CConfManager** - controls creation, modification and deletion of conferences. Manages resource usage and provides a conferencing service interface to the main module through a single object.
- **CConference** - low-level implementation of the Dialogic® Audio Conferencing API. The single CConfManager class contains a linked list of existing conferences represented by CConference objects. CConference objects are dynamically created by CConfManager, as needed.
- **CIpDev** - abstracts the IP signaling, H.323, and SIP abstracted under the Dialogic® Global Call API and streaming interface. This object also provides basic voice functionality for the contained media resource.

5.3 The Call Flow

At start-up, the Dialogic Audio Conferencing API Demo creates a single object of CConfManager, CConsoleIo (Windows® only), and CAppLog classes. CConfManager initializes all conference resources available on the system and sets board-level parameters that define all conferencing features (such as DTMF clamping, volume control, etc.). The application then enters an endless loop, where it shares time between sr_waitevt(20), waiting for an event with a 20 msec timeout, and waitForUserInput() waiting functions.

When any event is detected by sr_waitevt(), main() will check the event family first to define whether this event was generated by the IP front end device or the conference device.

If the event is generated by the IP call control library, the main() will search the event source object in a global array of CIpDev objects (ipDevArray), using the event device handle as an attribute (getEventObject() function, defined in main()) and call processEvent() method of this object.

Otherwise, if the event is detected on a conference device, the event will be passed to the processEvent() method of the single CConfManager object, which, in turn, will search an STL list of existing conference objects in an attempt to identify the appropriate conference that is associated with the event.

While the conference events in this application are used mainly for information purposes, the events from the IP front end are driving the application state machine.

Some IP events may be fully processed by CIpDev class, while others require additional instruction from the main(). A return code of 0 from the processEvent() method of CIpDev class indicates that the CIpDev object has processed the event completely and further processing is unnecessary. A non-zero return value indicates that the event object must be given further instructions by the main business logic module.

The following illustrates this approach using a single incoming IP call scenario:

- An IP channel device receives a GCEV_OFFERED event from the Dialogic® Global Call API library, indicating there's an incoming call request on this channel. This event is completely processed by CIpDev class state machine, defined in processEvent() method, which sets IP capabilities for this call, calling setCapabilities() method, and establishes a full-duplex connection between the network device and corresponding voice resource, using dx_Listen() and dti_Listen() methods.
- At this point, the processEvent() returns zero to the main(). Since dti_Listen() is called asynchronously, it will cause a GCEV_LISTEN event to be generated for this device. This event is passed to the processEvent() method of CIpDev, where the state machine decides that the call can be answered at this point, and calls the answer() function. The answer() method, in turn, generates a GCEV_ANSWERED event, notifying the application that the call was successfully accepted and is currently in the CONNECTED state. At this point, the CIpDev object needs to communicate this state to main() and get a new instruction, and processEvent() returns CALL_CONNECTED value to the main().
- The main() calls collectPasscode() method on this object, causing the object to play a greeting and then enter the Get DTMF state. After the object collects all digits (or a timer expires), it returns DIGITS_RECEIVED value to main() and waits for further commands / events.
- Next, the main() calls addToConference(evtDev) method of CConfManager class, passing the connected IP object as an argument. The CConfManager checks the passcode collected by the IP device, and, if valid, adds the device to a conference with this passcode (if it already exists), or create a new one.
- Upon receiving GCEV_DISCONNECTED event, the CIpDev state machine calls gc_DropCall() and gc_ReleaseCall() on the event object, and returns USER_DROPPED value to the main(). Main then instructs the ConfManager to remove the channel from a conference calling removeFromConference() method of the Conference Manager class object. The conference bridge number member data are reset to -1 for the disconnected device and the timeslot is freed via dti_Unlisten(). At this point, the channel enters the NULL state and is ready to accept a new call.

Index