Application Note

Building Conferencing Applications Using Intel® NetStructure™ Host Media Processing Software

Intel in Communications

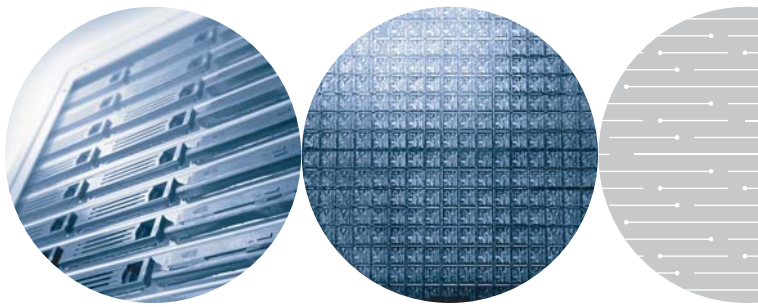

Table of Contents

Executive Summary

Intel® NetStructure™ Host Media Processing (HMP) software performs media processing tasks on general-purpose host processors in servers based on Intel® Architecture without the use of specialized digital signal processors (DSPs). This application note describes *HmpConfDemo*, a sample audio conferencing application that uses release 1.1 of HMP software for the Windows operating system to demonstrate its capabilities, including its use of H.323 and the Session Initiation Protocol (SIP) for call control.

What Is Host Media Processing?

Host media processing is a new way to deliver voice media processing in telephony applications by using the media processing capabilities of the host processors in off-the-shelf servers, eliminating the need for special telephony hardware based on DSPs.

Intel NetStructure Host Media Processing (HMP) software implements host media processing and is optimized for Intel® Pentium® processors. As a result, HMP software can supply media services for building flexible, scalable, cost-effective next-generation IP media servers. Developers can more rapidly create advanced voice processing applications while VARs and end users can significantly reduce the costs associated with installing, configuring, and maintaining these applications.

Release 1.1 of HMP software for the Windows operating system provides these main features

- **Built-in network interface card (NIC)** — Provides IP connectivity
- **Industry-standard H.323 and SIP protocol support** — Used for call control
- **H.450.2 supplementary services protocol support** — Supplies call transfer capability
- **IP multicast (transmit only)** — Facilitates implementation of features such as announcements and listen-only conferences with large numbers of participants

Other important features include:

- **Low-bit-rate coders** — Supports G.711, G.723, and G.729
- **Continuous Speech Processing** — Allows easy migration from PSTN-based speech solutions on telephony boards to an HMP environment
- **Scalability** — Provides up to 96 concurrent user sessions with a mix of voice, speech, T.38 fax, and conferencing media processing resources per system, with at least 50% of CPU and memory available to an application
- **Dual CPU configurations with hyperthreading** — Enables greater densities per inch of rack space.
- **Higher density** — Supplies up to 120 ports of interactive voice response (IVR) or conferencing per system.
- **T.38 fax** – Supports the most popular fax protocol for voice over IP (VoIP) networks

To ensure real-time media processing performance, HMP software is implemented as a Windows* operating system kernel-mode driver, which runs at real-time priority. The software is optimized for Intel Pentium III and Intel Pentium 4 processors.

HmpConfDemo Overview

In order to demonstrate the capabilities of HMP software release 1.1 for the Windows operating system, a simple audio conferencing application called *HmpConfDemo*, has been developed. *HmpConfDemo* shows how release 1.1 directly supports the H.323 and SIP signaling protocols for call control through the use of the Intel® Dialogic® Global Call API.

This document explains *HmpConfDemo* in detail, and describes ways to use the features of HMP software. The application is available for download at http://membersresource.intel.com/license/agreement.asp?url=downloads/appnotes/8896/HmpConfDemo.zip. The code can be used to easily create implementations that adapt existing communications applications for host media processing.

HmpConfDemo Features

HmpConfDemo is specifically designed to demonstrate the enhanced features of HMP software 1.1 listed in this section.

Call Control

- Global Call API
- SIP and H.323
- DTMF modes — In-band (H.323) and in-band and RFC 2833 (SIP)
- RAS (gatekeeper) support via Global Call API

Note that the number of simultaneous calls is limited by the number of licensed channels.

Media

- R4 Voice API
- Audio CODEC — G.711 µLaw with 10, 20, or 30 ms frame size
- Play and record capabilities
- Detection and generation of DTMF digits
- Bidirectional audio streaming using the Continuous Speech Processing API set

Conferencing

- Programming interface — Intel's DCB API
- Active talker detection
- DTMF detection
- DTMF clamping (optional)
- Echo cancellation (optional)
- Monitoring via receive-only audio recorder
- Setting or retrieving conference attributes while a conference is in progress

Application Interface and Configuration

HmpConfDemo provides a convenient console interface that supplies information about application performance and system statistics. Console command capabilities include:

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

At startup, *HmpConfDemo* reads the *conf_demo.cfg* configuration file to pre-set some common parameters. (Note that this file must be placed in the same directory as the application executable file.) All parameters are optional. If some parameters are omitted or the file cannot be found in the default location, the application will use a default parameter set, which is hard-coded in the program code.

The following section lists the parameters that can be set at startup.

Application-Level

- Number of simultaneous calls
- Path and filenames for audio files for welcome, invalid passcode, call later, and goodbye messages
- Path and filename for application log file
- Maximum size of log file
- Log file and console information verbosity

Board-Level

- Active talker feature (enable/disable)
- Active talker detection interval
- DTMF clamping (enable/disable)
- DTMF digit to enable/disable muting for participants

Single-Conference

- Passcode
- DTMF detection during conference (enable/disable)
- Echo cancellation (on/off)

Refer to the *Running HmpConfDemo* section later in this document for a list of the application parameters that can be updated during runtime and the *Appendix* for a list of default parameters.

Call Logging

All events, messages, and API calls are saved in a text file while the application is running. The maximum size of the log file and the logging level (verbosity) may be set via the application configuration file. If the data in the log file exceeds its specified maximum size, the logging data wraps to the beginning of the file and overwrites the data that was logged previously.

Limitations

- To simplify the source code for *HmpConfDemo*, gatekeeper support has not been omitted. However, sample code for gatekeeper integration is supplied as a starting point for programmers who want to add gatekeeper support.
- Conference-linked media resources will not detect DTMF tones if a conferee is in receive-only (mute) mode. A separate media resource is required to detect DTMF tones from conferees in receive-only mode.
- Volume control for individual participants is not supported in release 1.1 of HMP software.

HmpConfDemo Implementation

This section describes the system requirements and programming model for *HmpConfDemo*. It also includes details about its classes, objects, and call flow.

System Requirements

- **Hardware** — One standard NIC card
- **Software** — Intel NetStructure Host Media Processing Software Release 1.1 for Windows
- **Operating System** — Microsoft Windows* 2000

Programming Model

HmpConfDemo is written in asynchronous mode, using a single process/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-thread approach may be not optimal for high-density applications, this model was chosen to simplify programming and make the sample implementation easier to understand.

Detailed Description

Conferencing features in *HmpConfDemo* are accessed using the DCB API. The Global Call API is used to implement call control. Both APIs are designed for DM3 architecture. Intel's voice API is used for basic voice functionality.

Classes and Objects

HmpConfDemo defines and uses the C++ classes described in Table 1.

Class	Description
CAppLog	Provides the interface for application logging
CConsoleIo	Provides a set of console input/output operations. Contains two independently scrolled windows, a static menu bar, and user input space. Allows a user to dynamically build a menu, asynchronously wait for a menu key, or supply input. User interaction enables two types of callback functions, invoked by pressing a key. Function keys enable menu choices and the ENTER key enables 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	Implements the DCB (conferencing) API on a low level. The single *CConfManager* class contains a linked list of existing conferences represented by *CConference* objects. *CConfManager* dynamically creates *CConference* objects as needed.
CIpDev	Abstracts IP signaling, H.323, and SIP under the Global Call API and streaming interface. *CIpDev* objects provide basic voice functionality on the contained media resource.

Table 1. Classes in *HmpConfDemo*

Call Flow

At startup, the demo creates a single object of the *CConfManager, CConsoleIo,* 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 two waiting functions: *sr_waitevt(20)*, which waits for an event with a 20 msec timeout, and *waitForUserInput()*.

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

- If the event is generated by the IP call control library, *main()* will search the event source object in a global array of *CIpDev* objects called *ipDevArray*, using the *getEventObject()* function and event device handle as an argument of this function. *main()* will then call the *processEvent()* method of this object.
- If the event is detected on a DCB (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 conference associated with the event.

While the conference events detected on a DCB device are used mainly for informational purposes, the events from the IP front end are driving the application state machine.

Some IP events may be fully processed by the *CIpDev class*, while others require additional instruction from *main()*. A return code of 0 from the *processEvent()* method of the *CIpDev* class indicates that the *CIpDev* object has processed the event completely and no further processing is necessary. A non-zero return code indicates that the event object must be given further instructions by the main business logic module. This approach is illustrated below using a single incoming IP call scenario.

1. An IP channel device receives a *GCEV_OFFERED* event from the Global Call API library, indicating an incoming call request on the IP channel.
2. The *GCEV_OFFERED* event is processed completely by the *CIpDev* class state machine, defined in the *processEvent()* method, which sets IP capabilities for the call by using the *setCapabilities()* method and then calls the *dx_Listen()* and *dti_Listen()* methods to establish a full-duplex connection between the network device and corresponding voice resource. The *processEvent()* method then returns a zero value to *main()*.
3. Since *dti_Listen()* is called asynchronously, it will cause a *GCEV_LISTEN* event to be generated for the device. The event is passed to the *processEvent()* method of *CIpDev*, where the state machine decides if the call can be answered, and, if it can, calls the *answer() function*.
4. The *answer()* method generates a *GCEV_ANSWERED* event, notifying the application that the call was successfully accepted and is currently in *CONNECTED* state. At this point, the *CIpDev* object must communicate this state to *main()* and get a new instruction. The *processEvent()* method returns a *CALL_CONNECTED* value to *main()*.
5. *main()* calls the *collectPasscode()* method on the object, causing the object to play a greeting and then enter a “get DTMF” state. When the object collects all the digits in the passcode or a timer expires, it returns a *DIGITS_RECEIVED* value to *main()* and waits for further commands or events.
6. *main()* calls the *addToConference(evtDev)* method of the *CConfManager* class, passing the connected IP object as an argument. *CConfManager* checks the passcode collected by the IP device, and if it is valid, adds the device to a conference with this passcode, or creates a new one.
7. Upon receiving the *GCEV_DISCONNECTED* event, the *CIpDev* state machine calls *gc_DropCall()* and *gc_ReleaseCall()* on the event object, and returns the *USER_DROPPED* value to *main()*.

8. *main()* instructs the *ConfManager* to remove the channel from the appropriate conference, calling the *removeFromConference()* method of the *ConfManager* class object. The conference bridge number member data is reset to -1 for the disconnected device, and the timeslot is freed via *dti_Unlisten()*. The channel then enters the *NULL* state and is ready to accept a new call.

Running *HmpConfDemo*

This section contains information about preparing the configuration file, entering a conference, accessing administrative features, and muting a phone set during a conference.

Preparing the Configuration File

Before running *HmpConfDemo*, edit its configuration file and make any necessary changes. Then save the file to the same directory that contains the *HmpConfDemo* executable file *(HmpConfDemo.exe)*.

Figure 1 is a sample configuration file. Spacing and tabs have been altered for readability, and may not be shown with technical accuracy. Wording of notes may differ.

```
NumberOfChannels = 4                    ! Maximum number of IP channels in use
                                        ! Default is all available channels
MaxLogSize = 100000                     ! Maximum number of lines in a log file
                                        ! Default is 100,000 lines
PrintLevel = 1                          ! Printout verbosity
                                        ! 0=All, 1=App, 2=Events, 3=Warnings, 4=Error
LogFileName = HmpDemo.log               ! Default name = HmpDemo.log
WelcomeFileName = welcome.pcm
BadPasscodeFileName = invalid.pcm
CallLaterFileName = bye.pcm
GoodByeFileName = bye.pcm

! Setting PrintLevel = 0 may affect performance if demo is used under a
! heavy load

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

! Number of individual volume controls is not supported in HMP 1.1 beta

VolumeControl = Yes
VolumeUp = 3
VolumeDown = 9
VolumeReset = 0

! Conference Information Sections
! Header Format: [Conference xxx]  where xxx is a unique decimal number
! (bridge number)
! Passcode field is mandatory and must be a unique conference ID
! All other values default to "No"

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

[Conference 6]
Passcode = 22345
DetectDigits = No
EchoCanceler = No

[Conference 4]
Passcode =12347

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

Figure 1. Sample Configuration File

Entering a Conference

To enter a conference, use the following procedure:

1. Access the directory which contains *HmpConfDemo*, and double-click on the *HmpConfDemo.exe* file.
2. Make an IP call to the *HmpConfDemo* using an IP address of the host PC on which the demo is running.
3. Enter the passcode for the conference that you wish to enter by using the dial pad on your IP phone. Conference passcodes are set up in the configuration file. For example, several conferences are set up in the sample configuration file in Figure 1. Conference 12 requires passcode 12345, and Conference 6 requires passcode 22345.

When a valid passcode is entered, you will be placed into the appropriate conference.

Accessing Administrative Features

Function keys are used to access the features of *HmpConfDemo*. The function keys are set up as follows:

F2

Updates the settings in the configuration file while the conferencing application is running. Edit the configuration file, and then press the F2 key to submit it to the application. The following parameters may be updated while the application is running:

Application-Level

- Path and filenames for audio files for welcome, invalid passcode, call later, and goodbye messages
- Log file and console information verbosity

Board-Level

- DTMF digit to mute participants (enable/disable muting)

Single Conference

- DTMF detection during conference (enable/disable)
- Echo cancellation (on/off)

Changing passcode and bridge number information during a conference will not affect the conference in progress, but will be recognized by the application when a newcaller attempts to enter the conference with the updated passcode.

F3

Retrieve information about the active talker.

F4

Change log and print level using the following settings:

0 = All messages
1 = API calls
2 = Events
3 = Warnings
4 = Errors
5 or above = None

F5

Display application statistics. The following statistics will be displayed:

- Number of conferences currently in progress
- Number of participants in each conference
- Number of conference resources currently available

F6

Start monitoring. User will be prompted to enter the bridge number of a conference to be monitored. Conference bridge numbers are displayed in the upper console window or can be accessed via the F5 function key.

When a bridge number is entered, the application will add a media resource to the conference (if one is available) in receive-only mode and will start recording a file called *Conf_xxx.pcm*, where xxx is the conference bridge number.

F7

Stop recording and remove the monitor from a conference.

F8

Stop a single conference. User will be prompted for the bridge number of the conference to be stopped.

F9

Stop all conferences. Pressing this function key will end all conferences that are currently in progress and free all resources. Calls will not be dropped.

F10
Exit the application.

Muting a Phone Set

To enable or disable muting for a phone set during a conference, press "*6" on the dial pad. The default digits for this function may be changed by editing the configuration file.

For More Information

Information about obtaining a copy of Intel NetStructure Host Media Processing (HMP) software can be found at **http://www.intel.com/network/csp/products/hmp/8399web.htm**.

More detailed information about HMP software release 1.1 is available at **http://www.intel.com/network/|csp/products/8762web.htm**.

A reference design guide, which includes information about building interactive voice response (IVR) applications with HMP software, can be found at **http://resource.intel.com/telecom/support/appnotes/an03002/index.htm**.

Appendix A: List of Defaults

This appendix contains a list of hard-coded default values for omitted or misspelled parameters in the configuration file of *HmpConfDemo*. The same defaults are used if the application cannot find, read, or parse the configuration file.

Default passcodes and their corresponding bridge numbers are listed in Table 2.

Passcodes	Bridge Numbers
12345	1
23456	2
34567	3
45678	4
56789	5
67890	6
78901	7

Table 2. Passcodes and bridge numbers

Default parameter settings are listed below by type.

Application-Level

- Number of simultaneous calls = 32
- Path and filenames for audio files messages:
  - Welcome = *welcome.pcm* in current directory
  - Invalid Passcode = *invalid.pcm* in current directory
  - Call Later = *bye.pcm* in current directory
  - Goodbye = *bye.pcm* in current directory
- Path and filename for application log file = *HmpDemo.log* in current directory
- Maximum size of log file = 100,000 lines
- Log file and console information verbosity = 0

Board-Level

- Active talker feature = disable
- Active talker detection interval = 3 (in 100 ms per unit)
- DTMF clamping = disable
- DTMF digit to enable/disable muting for participants = *6

Single-Conference

- Passcode = see Table 2
- DTMF detection during conference = disable
- Echo cancellation = off

To learn more, visit our site on the World Wide Web at **http://www.intel.com**.

1515 Route Ten
Parsippany, NJ 07054
Phone: 1-973-993-3000

Printed in the USA Printed on recycled paper. 12/03 00-8896-001