Multimedia API

Programming Guide

August 2006

05-2455-004

Publication Date: August 2006

Document Number: 05-2455-004

Intel
1515 Route 10
Parsippany, NJ 07054

For **Technical Support**, visit the Intel Telecom Support Resources website at:
http://developer.intel.com/design/telecom/support

For **Products and Services Information**, visit the Intel Telecom and Compute Products website at:
http://www.intel.com/design/network/products/telecom

For **Sales Offices** and other contact information, visit the Buy Telecom Products page at:
http://www.intel.com/buy/networking/telecom.htm

Contents

Figures

Revision History

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

Document No.	Publication Date	Description of Revisions
05-2455-004	August 2006	Product Description: Revised first paragraph of Intraframe (I-Frame) section. Revised bullets about support for video picture formats and the H.263 video codec in Multimedia Capabilities. Requesting I-Frame Using SIP INFO: Revised first paragraph. Variables for Compiling and Linking: Removed example. Glossary: Added definition of I-frame.
05-2455-003	May 2006	Global: Updated to remove Linux-specific wording. Changed title of document. Video Codec: Changed Note to make it applicable to the current HMP release. SDP Functionality: Made last paragraph applicable to the current HMP release.
05-2455-002	January 2006	Chapter changes: Moved the file conversion information that was in chapter 5 of the *Programming Guide* to a new document: the *Multimedia File Conversion Tools User Guide* to (05-2453-001), which has been updated and is available from the following web site: *http://resource.intel.com/telecom/support/hmpmedia/index.htm* Made minor change in chapter sequence. Related Information section: Added the *Multimedia File Conversion Tools User Guide*. Multimedia Capabilities section: Based on web site revisions, changed the URL for the web page used to obtain the multimedia file conversion tools.
05-2455-001	September 2005	Initial version of document, published for Intel NetStructure® Host Media Processing Software Release 1.5 for Linux operating systems.

About This Publication

The following topics provide information about this publication:

Purpose

This document provides information about video technology and multimedia library features, as well as guidelines for building applications using the Multimedia API. The Multimedia API provides the ability to record and play back digitized multimedia (audio and video) to support video services in application programs.

This publication is a companion guide to the *Multimedia API Library Reference*, which provides details on functions and parameters in the multimedia software.

Applicability

This document was originally published to accompany the Intel NetStructure® Host Media Processing (HMP) Software Release 1.5 for Linux* operating systems.

This document may also apply to later releases of the HMP software as well as other Intel® telecom software releases. Check the *Release Guide* for your software release to determine whether this document is supported.

Intended Audience

This information 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 HMP software which includes the multimedia software. This publication assumes that you are familiar with and have prior experience with the operating system and the C programming language.

The information in this publication is organized as follows:

- Chapter 1, "Product Description" – provides an overview of video technology and the multimedia features of the HMP software.
- Chapter 2, "Event Handling" – provides information about Multimedia API events.
- Chapter 3, "Error Handling" – provides information about Multimedia API errors.
- Chapter 4, "Application Development Guidelines" – provides some guidelines for developing multimedia applications.
- Chapter 5, "Building Applications" – provides general information about building applications using the Multimedia API library.

This document also includes a glossary of terms and an index.

Related Information

See the following for related information:

- For information on obtaining and using the multimedia file conversion tools, see the following web site:
  http://resource.intel.com/telecom/support/hmpmedia/index.htm
- For related product documentation on Intel® telecom products, see the on-line documentation bookshelf provided with the software release or at the following Telecom Support Resources web site:
  http://resource.intel.com/telecom/support/documentation/releases/index.htm.
- For details on product known problems and late-breaking updates or corrections to the release documentation, see the *Release Update* for the software release you are using. Be sure to check the *Release Update* for any documentation updates or corrections. The *Release Update* is available from the documentation bookshelf on the web at the Telecom Support Resources website.
- For information on the product release, features, and system requirements, see the *Release Guide*.
- For reference information for all functions, parameters, data structures, values, events, and error codes in the Multimedia API, see the *Multimedia API Library Reference*.
- For information about the multimedia demo, see the *Multimedia for Host Media Processing Demo Guide*.

Product Description 1

This chapter provides an overview of video technology and the multimedia features of the Intel NetStructure® Host Media Processing (HMP) software. The following topics are included:

1.1 Overview of Video Technology

This section covers the following concepts in video technology:

- Video Codec
- Audio Codec
- Video Standards
- Intraframe (I-Frame)
- IP Media Server

1.1.1 Video Codec

A video codec is a device or software module that encodes and compresses raw video data to a digital format or decodes and decompresses from a digital video data format to a raw video data format. A video codec may also transcode between two digital formats. For multimedia, video transcoding is provided, if necessary, between the digital video format of the Real-time Transport Protocol (RTP) stream and the digital video format of the multimedia file that is being played from or recorded to.

Note: For this HMP release, video transcoding is not provided. The digital video format of the RTP stream and the multimedia file being played from or recorded to will match.

1.1.2 Audio Codec

An audio codec is a device or software module that encodes and compresses analog audio data to a digital format or decodes and decompresses from a digital audio data format to analog. An audio codec may also transcode between two digital formats. For multimedia, audio transcoding is provided, if necessary, between the digital audio format of the RTP stream and the digital audio format of the multimedia file that is being played from or recorded to.

1.1.3 Video Standards

The most important video standards today are H.261, H.263, MPEG-2 and MPEG-4. These and other standards are discussed in the following sections:

- H. 261 (P*64)
- H.263
- MPEG-1
- MPEG-2
- MPEG-4
- H.264

Compared to video codecs for CD-ROM or TV broadcast, codecs designed for the Internet require greater scalability, lower computational complexity, greater resiliency to network losses, and lower encode/decode latency for video conferencing. In addition, the codecs must be tightly linked to network delivery software to achieve the highest possible frame rates and picture quality.

Not all video standards are ideal for Internet video. Only very recently have standards such as MPEG-4 part 10 and H.264 (which are the same codec) emerged which bridge the gap between high efficiency error-resistant codecs for telephony applications (H.26x) and codecs designed for entertainment content streaming (MPEG-x). In the coming years, more effort will probably be focused on algorithms specifically designed for real-time video applications in the Internet. Research is currently underway looking at new scalable and flexible codecs, video QoS, etc.

H. 261 (P*64)

H.261, also known as P*64 codec (where P is an integer number meant to represent multiples of 64 Kbps), is an old standard and was targeted at teleconferencing applications. H.261 is intended for carrying video over ISDN – in particular for face-to-face videophone applications and video conferencing. The actual encoding algorithm is similar to but incompatible with that of MPEG codecs. However, H.261 needs substantially less CPU power for real-time encoding than MPEG. The algorithm includes a mechanism that optimizes bandwidth usage by trading picture quality against motion, so that a quickly-changing picture will have a lower quality than a relatively static picture. Used in this way, H.261 is a constant-bit-rate encoding codec rather than a constant-quality, variable-bit-rate encoding codec.

H.263

H.263 is the ITU-T standard designed for low bit rate communication. However, this standard will probably be used for a wide range of applications – not just low bit rate applications. H.263 is expected to replace H.261 in most applications. This has already happened in many applications.

The coding algorithm of H.263 is similar to that used by H.261, but it has been enhanced to improve performance and error recovery. H.263 uses half-pixel precision, whereas H.261 uses full-pixel precision and a loop filter. Some parts of the hierarchical structure of the data stream are now optional, so the codec can be configured for a lower data rate or better error recovery.

There are now four optional negotiable options included to improve performance:

- Unrestricted motion vectors
- Syntax-based arithmetic coding
- Advance prediction
- Forward and backward frame prediction similar to MPEG called P-B frames.

H.263 supports five resolutions:

- Quarter Common Intermediate Format (QCIF) and Common Intermediate Format (CIF) – the resolutions supported by H.261
- Sub-QCIF (SQCIF) – half the resolution of QCIF
- 4CIF – 4 times the resolution of CIF
- 16CIF – 16 times the resolution of CIF

The support of 4CIF and 16CIF means the codec can compete with other, higher bit rate video coding standards such as the MPEG standards.

MPEG-1

MPEG-1, 2, and 4 are currently accepted standards for the bandwidth efficient transmission of video and audio. The MPEG-1 codec targets a bandwidth of 1 to 1.5 Mbps offering VHS quality video at CIF (352x288) resolution and 30 frames per second. MPEG-1 requires expensive hardware for real-time encoding. While decoding can be done in software, most implementations consume a large fraction of a high-end processor. MPEG-1 does not offer resolution scalability and the video quality is highly susceptible to packet losses due to the dependencies present in the P (predicted) and B (bi-directionally predicted) frames (for more information about P and B frames, refer to Section 1.1.4, “Intraframe (I-Frame)”, on page 13). The B-frames also introduce latency in the encode process, since encoding frame N needs access to frame N+k, making it less suitable for video conferencing.

MPEG-2

MPEG-2 extends MPEG-1 by including support for higher resolution video and increased audio capabilities. The targeted bit rate for MPEG-2 is 4 to15 Mbps, providing broadcast quality full-screen video. The MPEG-2 draft standard does provide for scalability. Three types of scalability have been defined:

- Signal-to-Noise Ratio (SNR)
- Spatial and temporal
- One extension that can be used to implement scalability: data partitioning.

Compared with MPEG-1, MPEG-2 requires even more expensive hardware to encode and decode. MPEG-2 is also prone to poor video quality in the presence of losses, for the same reasons as MPEG-1. Both MPEG-1 and MPEG-2 are well suited to the purposes for which they were developed. For example, MPEG-1 works very well for playback from CD-ROM, and MPEG-2 is great for high-quality archiving applications and for TV broadcast applications. In the case of satellite broadcasts, MPEG-2 allows more than five digital channels to be encoded using the same

bandwidth as used by a single analog channel today, without sacrificing video quality. Given this major advantage, the large encoding costs are really not a factor. However, for existing computer and Internet infrastructures, MPEG-based solutions are too expensive and require too much bandwidth because they were not designed with the Internet in mind.

MPEG-4

MPEG-4, the latest encoding standard from MPEG, was finalized in October 1998 and ratified as a standard in 1999. MPEG-4 arose from a need to have a scalable standard supporting a wide bandwidth range from streaming video at less than 64 Kbps, suitable for Internet applications, to approximately 4 Mbps for higher-bandwidth video needs. MPEG-4 also arose from a desire, as digital encoding matures, to advance beyond simple conversion and compression to object recognition and encoding, as well as provide synchronized text and metadata tracks, to create a digital file that carries a meaning greater than the sum of its individual parts.

MPEG-4 supports both progressive and interlaced video encoding. One key concept of MPEG-4 is that it is object-based, coding multiple video object planes into images of arbitrary shape. Successive video object planes (VOPs) belonging to the same object in the same scene are encoded as video objects. MPEG-4 supports both natural (“analog”) and synthetic (“computer-generated”) data coding. Some VRML (Virtual Reality Modeling Language) technology is also incorporated to encode dimensionality.

MPEG-4 compression provides temporal scalability utilizing object recognition, providing higher compression for background objects, such as trees and scenery, and lower compression for foreground objects, such as an actor or speaker – much as the human eye filters information by focusing on the most significant object in view, such as the other party in a conversation. Object encoding provides great potential for object or visual recognition indexing, based on discrete objects within a frame rather than requiring a separate text-based or storyboard indexing database. In addition, MPEG-4 provides a synchronized text tract for courseware development and a synchronized metadata track for indexing and access at the frame level.

H.264

The main objective behind the H.264 project was to develop a high-performance video coding standard by adopting a “back to basics” approach where simple and straightforward design using well-known building blocks is used. The emerging H.264 standard has a number of advantages that distinguish it from the other existing standards mentioned above, while at the same time sharing common features with other existing standards. The following are some of the key advantages of H.264:

- Compared to H.263v2 (H.263+) or MPEG-4 Simple Profile, H.264 permits a reduction in bit rate by up to 50% for a similar degree of encoder optimization at most bit rates.
- Higher quality video: H.264 offers consistently good video quality at high and low bit rates.
- Error resilience: H.264 provides the tools necessary to deal with packet loss in packet networks and bit errors in error-prone wireless networks.
- Network friendliness: Through the Network Adaptation Layer, H.264 bit streams can be easily transported over different networks.

In general, H.263 and MPEG-4 have become the defacto standards for video delivery over low bandwidths. But broadband standards such as MPEG-1 and MPEG-2, which are useful for many types of broadcast and CD-ROM applications, are unsuitable for the Internet, at least in the near future. In fact, with the adoption of broadband, we should expect to see significant growth in MPEG-1 and MPEG-2 applications. Although MPEG-2 has had scalability enhancements, these will not be exploitable until the availability of reasonably priced hardware encoders and decoders that support scalable MPEG-2.

1.1.4 Intraframe (I-Frame)

To create motion in a video, individual frames of pictures are grouped together and played back. An Intraframe (I-frame) is a single frame of digital content that the compressor examines independent of the frames that precede and follow it and stores all of the data needed to display that frame. Several video codec standards, such as MPEG-4 and H.263, use I-frames.

In a compressed video, I-frames are usually used along with P-frames and B-frames, which are sometimes referred to as Interframes or delta frames. P-frames (short for predictive frames) follow I-frames and contain only data that has changed from the I-frame that comes before it. B-frames (short for bi-directional predictive frames) contain only the data that have changed from the preceding frame or differ from the data in the next frame.

So only the I-frame contains all the data needed to display the frame whereas the P- and B-frames store the changes that occur between the I-frames. For this reason, video decoders typically require at least a single initial I-frame before they can begin to accurately display video content. The quality of the video increases with a higher amount of I-frames, but I-frames take up more storage space and transmission bandwidth because they contain the highest amount of information bits.

1.1.5 IP Media Server

An IP media server provides various video services to wire line and wireless client devices, through a media gateway, if connection to a different network is required. Typical media services involving video include video mail, video conferencing, and video streaming.

1.2 Multimedia Capabilities

This section lists the multimedia capabilities provided by the Multimedia API Library and other related API libraries:

- Multimedia API Library - records and plays the multimedia data using a multimedia device.
- Device Management API Library - used to connect the multimedia device with an IP media device.
- IP Media API Library - provides the IP multimedia session control.
- Global Call API Library - provides IP call control for multimedia using SIP and Session Description Protocol (SDP) and must be used in third party call control (3PCC) mode.

More information about these APIs can be found in the corresponding API Library Reference documentation provided with this release.

These capabilities support applications providing video services, such as video mail, video color ring, video caller ID, and video location-based services.

The following multimedia capabilities are provided:

- Multimedia record and playback with basic playback control and synchronized audio and video.
  - Record and playback of audio and video, video only, or audio only
  - Record to and playback from a file or set of files
  - Transmit notification tone at start of recording
- Record from RTP stream to multimedia file. Play from multimedia file into RTP stream while maintaining synchronization.
- Supports the following audio codecs for RTP:
  - G.711
  - G.723.1
  - G.729A (compatible with G.729 format)
  - G.729AB (compatible with G.729B format)
- Supports the following video picture formats at 30, 15, 10, or 6 frames per second:
  - Common Intermediate Format (CIF) picture size (352 pixels by 288 pixels)
  - Quarter Common Intermediate Format (QCIF) picture size (176 pixels by 144 pixels)
  - Sub-QCIF picture size (128 pixels by 96 pixels), used for mobile handsets
- Supports the H.263 video codec. It can operate with a bit rate up to 6 x (64,000) = 384,000 bits per second, with a picture decoding rate up to (30,000)/1001 pictures per second.
- Supports a proprietary video file format and the Linear PCM (128 kbps) audio file format.
- Multimedia File Conversion Tools: These utilities (e.g., mmconvert and hmp3gp) provide off-line conversion of multimedia files. For information on obtaining and using the multimedia file conversion tools, see the following web site (check this web site periodically for updates to the conversion tools and their capabilities, as well as to the Multimedia File Conversion Tools User Guide):
  http://resource.intel.com/telecom/support/hmpmedia/index.htm
- Play to and record from SIP devices, depending on the capability of the device (audio or audio/video). Play video only if no audio is required. Play audio only for non-video devices.
- Supports existing IP Media Library (IPML) audio alarms for the voice portion of the multimedia stream.

  Note: Quality of Service (QoS) alarms and events are not supported for video streams.
- Play Voice API audio files in a multimedia session. You can play Voice API audio files in a multimedia session as long as tight synchronization with video is not required (as when playing with a video menu or status display). In this case, the “ipm” device in a multimedia session will listen to the “dxxx” device to which the Voice API is playing an audio file. This overrides any audio stream (but not video) from the “mm” device in the multimedia session.

1.3 File Formats

The Multimedia API supports the following file formats:

- Voice API Audio Play and Record File Formats
  - G.711 μ-law and A-law (48 kbps and 64 kbps)
  - OKI ADPCM (24 kbps and 32 kbps)
  - G.726 (16 kbps and 32 kbps)
  - Linear PCM (88 kbps)
  - Linear PCM (128 kbps).
- Multimedia API Audio Play and Record File Format
  - Linear PCM (128 kbps): 16-bit, 8 kHz, mono, LSB-MSB ("little-endian").
- Multimedia API Video Play and Record File Formats
  - Intel proprietary format

Note: Except for the 128 kbps linear PCM file format, the Multimedia API does not directly support the Voice API audio files. However, they are indirectly supported and you can play these Voice API audio files in a multimedia session as long as tight synchronization with video is not required (as when playing with a video menu or status display). In this case, the "ipm" device in a multimedia session will listen to the "dxxx" device to which the Voice API is playing an audio file. This overrides any audio stream (but not video) from the "mm" device in the multimedia session.

1.4 SIP Call Control Using Global Call

This section discuss the use of Global Call APIs for SIP call control in multimedia applications.

Note: More information about the Global Call API can be found in the *Global Call API for HMP Library Reference* and *Global Call API for HMP Programming Guide*.

The Global Call API is a simple to use "C" interface that allows you to develop both TDM and IP call control applications using a variety of TDM protocols such as ISDN PRI and T1 CAS, and IP protocols like SIP and H.323. It provides an abstraction to the application from the lower level details of each individual protocol, thus enabling faster deployment of multimedia applications.

The Global Call call control library operates in two modes:

- First Party Call Control (1PCC) mode, which is the default
- Third Party Call Control (3PCC) mode

First Party Call Control (1PCC) mode is where the Global Call call control library manages IP signaling (SIP or H.323) as well as the RTP media control via the IP Media Library (IPML). IPML is a complementary API that is used to manipulate RTP media streams (start, stop, or modify them). The call control library synchronizes the call control and media control state machines, thus abstracting the application from the details of creating and parsing Session Description Protocol

(SDP) bodies that are embedded in SIP messages and invoking the IPML functions to start, stop, or modify RTP streams.

Note: More information about the IP Media Library API can be found in the *IP Media Library API for HMP Library Reference* and *IP Media Library API for HMP Programming Guide*.

In the Third Party Call Control (3PCC) mode, the Global Call call control library provides maximum flexibility to the application, allowing it to manipulate both SDP bodies and invoke IPML functions as needed. The Global Call library in this mode is acting as a pure SIP application. The application is responsible for monitoring the various SIP transactions and invoking appropriate IPML functions to start, stop, or modify RTP streaming and create or parse SDP bodies.

Apart from the capabilities discussed above, Global Call allows the application to manipulate any standard SIP header (read or write), while also providing the flexibility to add custom headers to SIP messages.

Note: To play and record multimedia with the Multimedia API, you must configure and use the Global Call library for 3PCC mode.

1.5 SDP Functionality

This section discusses Session Description Protocol (SDP) functionality.

SDP, which is defined in IETF RFC 2327, provides a standard way to specify the media information for an IP call. It is used by SIP, MGCP, and H.248 (or Megaco) protocols. The media information could be anything from RTP audio and video details such as UDP port numbers, IP addresses, audio/video coder information, or bandwidth. A sample SDP body is shown below:

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
e=mjh@isi.edu (Mark Handley)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

In this HMP release, the application is responsible for creating and parsing these SDP bodies that are embedded in SIP messages. For convenience, a sample class library is provided “as is” for developers to use to create and parse SDP bodies. It is a simple C++ interface provided as source and can be modified to suit application needs.

Event Handling 2

This chapter provides information about event handling:

An event indicates that a specific activity has occurred. Events provide feedback on the progress and completion of functions and indicate the occurrence of other activities. Multimedia library events are defined in the *mmevts.h* header file.

For a list of events that may be returned by the multimedia software, see the *Multimedia API Library Reference*.

Following is an example of how Multimedia API events and errors are handled. The function used in the example is **mm_Play( )**.

A typical application scenario when doing **mm_Play( )** is as follows:

1. The application initiates the multimedia play operation by calling the **mm_Play( )** function.
2. A return value of the call (EMM_SUCCESS or EMM_ERROR) indicates the success or failure of the call. Failure usually indicates that the parameter list is incorrect. No Standard Runtime Library (SRL) events are generated for this.

   If EMM_ERROR is returned, use the **mm_ErrorInfo( )** function to obtain the reason for the failure. This function should be called immediately after the failed **mm_Play( )** call and in the same execution thread. The **mm_ErrorInfo( )** function returns a structure with an error code (one of the EMM_ prefixed values defined in the *mmerrs.h* header file) and character strings with text messages describing the error.
3. Upon initiation of the play operation, either the MMEV_PLAY_ACK or MMEV_PLAY_ACK_FAIL event is posted to the SRL event queue with event data pointing to the MM_PLAY_ACK structure. This structure holds the return code of the operation. Return codes are defined in the *mmerrs.h* header file and have a EMMRC_ prefix. Success is indicated with the EMMRC_OK return code.

   If the play initiation fails, no further processing takes place and no more SRL events are generated for this operation.
4. When the play operation finishes (because the end of data or stop operation is issued), either the MMEV_PLAY or MMEV_PLAY_FAIL event is posted to the SRL queue with event data pointing to the MM_PLAY_CMPLT structure. This structure holds the details of the operation.

It is recommended that you use the **mm_GetMetaEvent( )** function to obtain detailed information about SRL events related to Multimedia API calls. This function returns a structure with all details of the event.

At any point of operation execution, a general error event (MMEV_ERROR) could be posted to the SRL queue. This error indicates that some unexpected system error event occurred. It could indicate a problems such as resources depletion, memory corruption, or internal communication failures.

Error Handling 3

This chapter describes error handling for the multimedia software.

All Multimedia API library functions return a value that indicates the success or failure of the function call. Success is indicated by a return value of EMM_SUCCESS for success or EMM_ERROR for failure. Failure usually indicates that the parameter list is incorrect. No Standard Runtime Library (SRL) events are generated for this.

If a function fails, use the **mm_ErrorInfo( )** function to obtain the reason for the failure. This function should be called immediately after the failed function call and in the same execution thread. It returns a structure with the error code (one of the EMM_ prefixed values defined in the *mmerrs.h* header file) and character strings with test messages describing the error.

At any point of operation execution, a general error event (MMEV_ERROR) could be posted to the SRL queue. This error indicates that some unexpected system error event occurred. It could be indication of a problem such as resource depletion, memory corruption, or internal communication failures.

Application Development Guidelines 4

This chapter provides some guidelines for developing multimedia applications. The following topics are included:

4.1 Developing Multimedia Applications

This section describes the following multimedia applications:

- Video Mail
- Video Color Ring
- Video Caller ID
- Video Location Based Services

4.1.1 Video Mail

In the context of HMP, video mail is an extension of voice mail services. The HMP IP Multimedia Server (IMS), in conjunction with other network elements, can provide the following capabilities to phone users, both mobile and wired, including soft-phone users:

- a party of an a/v (audio/video) call can leave an a/v message (or greeting)
- a party of an a/v call can be greeted by an a/v message
- a party of an a/v call can retrieve an a/v message
- a party of an a/v call can retrieve an audio only message while being presented a status menu, a still image, or no video
- a party of an a/v call can navigate through a menu, both visual and audio, by pressing phone keys
- a party of an audio only call can retrieve the audio portion of an a/v message

The HMP IMS, in conjunction with other system elements (primarily the application), can provide the following capabilities to PC web browser users:

- leave an a/v message or greeting
- retrieve an a/v message

The HMP IMS, along with tools and utilities provided with HMP, allows the output (such as a/v files) of popular content creation tools to be used by the HMP IMS as a source of the a/v content that is streamed to a caller. One use of this a/v content would be to support the creation and use of video menus.

4.1.2 Video Color Ring

Video color ring is a telecom service where instead of standard ring-back tones, a caller receives a specially prepared audio or audio/visual greeting while the called phone is ringing. The greeting is specific to the called number and may also be specific to the calling number. For example, special greetings may be provided to family or friends and a general greeting may be provided to all other callers.

Unlike video mail, where a full duplex, normal media session is established between the HMP IMS and the calling party, for video color ring, a half duplex, early media session (pre-answer) needs to be established between the HMP IMS and the calling party.

For this HMP release, HMP will not directly support early media between the HMP IMS and the caller and therefore depends on other network elements, such as a gateway, to provide this capability. Phones that support video color ring are also required.

In a 3G or IP environment, where the HMP IMS resides behind a gateway, video color ring can be supported without the HMP IMS directly supporting early media. Early media support can be established to the caller by another subsystem in the network, such as by the gateway. Again, this assumes that the gateway supports early media. A full duplex call between the HMP IMS and the gateway could be established and the HMP IMS could provide the IP address and RTP port of the caller, allowing the HMP IMS to stream the video color ring to the caller. As an alternate approach, call signaling to the HMP IMS is not even required. The HMP IMS could be instructed to just stream the appropriate media to a specified IP address and RTP port(s).

Assuming that early media is handled by other network elements and phones supporting video color ring are available, the HMP IMS can provide the following capabilities in support of the video color ring service:

- set up an a/v ring back greeting or greetings from a phone or a PC web browser
- set up an a/v session and stream an a/v ring back greeting to a specified IP address and port.

4.1.3 Video Caller ID

Video caller identification is a telecom service where instead of the caller's phone number appearing on the called phone, a pre-recorded audio/visual message is sent and presented according to the capability of the called phone.

The capabilities needed to be supported by the IMS are similar to those required to support video mail and video color ring services (refer to Section 4.1.1, “Video Mail”, on page 21 and Section 4.1.2, “Video Color Ring”, on page 22).

Similar to the video color ring service, where a half duplex, early media session (pre-answer) is established between the calling party and the IMS, for video caller ID, a half duplex, early media session (pre-answer) is also established but in this case it is between the IMS and the called party.

For this HMP release, HMP will not directly support early media between the HMP IMS and the called party and therefore depends on other network elements, such as a gateway, to provide this capability. Phones that support video caller ID will also be required. Most phones that support video caller ID today support a different form of video caller ID. These phones allow photos to be stored locally on the phone and the stored picture is associated with a caller ID number and are displayed when a incoming call with one of these numbers arrives at the phone.

In a 3G or IP environment, where the HMP IMS resides behind a gateway, video caller ID can be supported without the HMP IMS directly supporting early media. Early media support can be established to the called party by another subsystem in the network, such as by the gateway. Again, this assumes that the gateway supports early media. A full duplex call between the HMP IMS and the gateway could be established and the HMP IMS could provide the IP address and RTP port of the called party, allowing the HMP IMS to stream the video caller ID to the called party. As an alternate approach, call signaling to the HMP IMS is not even required. The HMP IMS could be instructed to just stream the appropriate media to a specified IP address and RTP port(s).

Assuming that early media is handled by other network elements and phones supporting this type of video caller ID are available, the HMP IMS can provide the following capabilities in support of the video caller ID service:

- set up an a/v caller ID greeting (or greetings) from a phone or a PC web browser.
- set up an a/v session and stream an a/v caller ID greeting to a specified IP address and port.

4.1.4 Video Location Based Services

Video location based service is a telecom service where subscribers can access local information such as local weather, restaurant guides and reviews, movie listings, and sports highlight clips. Video content is stored on a central server where it can be accessed by video subscribers.

Like video mail, a full duplex, normal media session is established between the HMP IMS and the calling party.

The HMP IMS, in conjunction with other network elements, can provide the following capabilities to phone users, both mobile and wired:

- a party of an a/v call can be greeted by an a/v message
- a party of an a/v call can navigate through a menu by pressing phone keys
- a party of an a/v call can retrieve a/v content

The HMP IMS, along with tools and utilities provided with HMP, will allow the output (such as a/v files) of popular content creation tools to be used by the HMP IMS as a source of the a/v content that is streamed to a caller.

4.2 Building Blocks for Multimedia Applications

The HMP software release provides APIs and libraries for developing a variety of applications. The APIs or building blocks that are used to develop audio/video applications include the following:

- Global Call Library for SIP call control (gc_)
- IP Media Library for RTP control (ipm_)
- Multimedia Library for playing and recording audio/video files (mm_)
- Voice Library for DTMF detection/generation (dx_)
- Device management library for setting up virtual connections between devices (dm_)
- SDP library to create and parse SDP bodies

Using the above sets of APIs, the applications that have been discussed in previous sections can be easily developed. The HMP software release includes a sample demo (multimedia demo) that demonstrates a video mail and a video portal application using the above APIs and their associated libraries.

4.3 Call Flow of Video Mail

This section provides illustrations of a typical call flow and the steps to take to develop a video mail application using Global Call.

Figure 1 shows a typical scenario of the video mail application.

Figure 1. Typical Call Flow of Video Mail

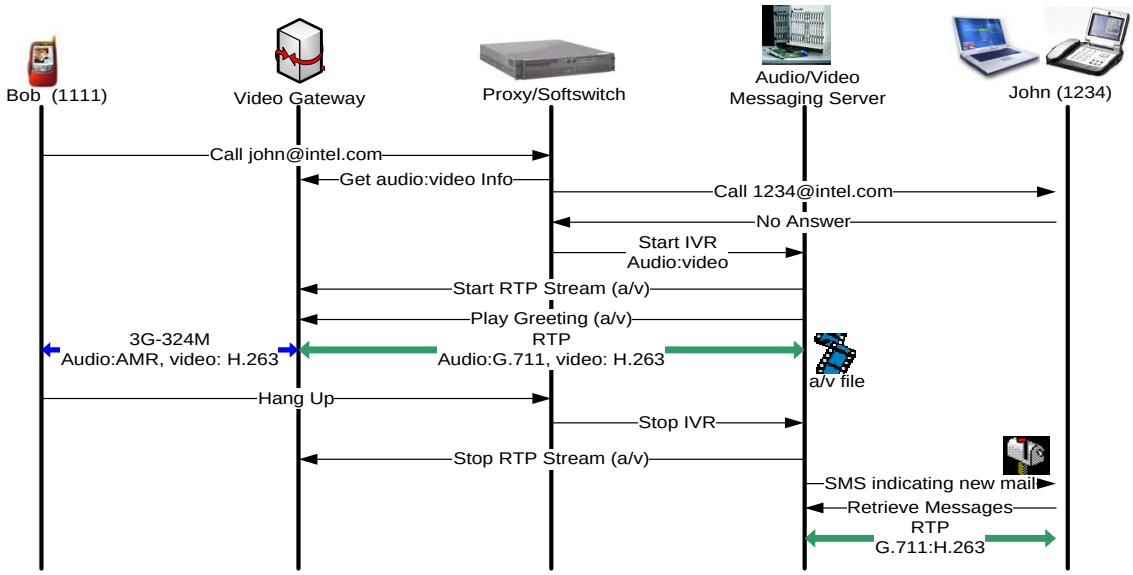


Figure 2 and Figure 3 show the steps to take to develop a video mail application using Global Call. Figure 2 shows a scenario in which the SIP message has both audio and video information in the SDP.

Figure 2. Incoming SIP Call with Audio and Video SDP

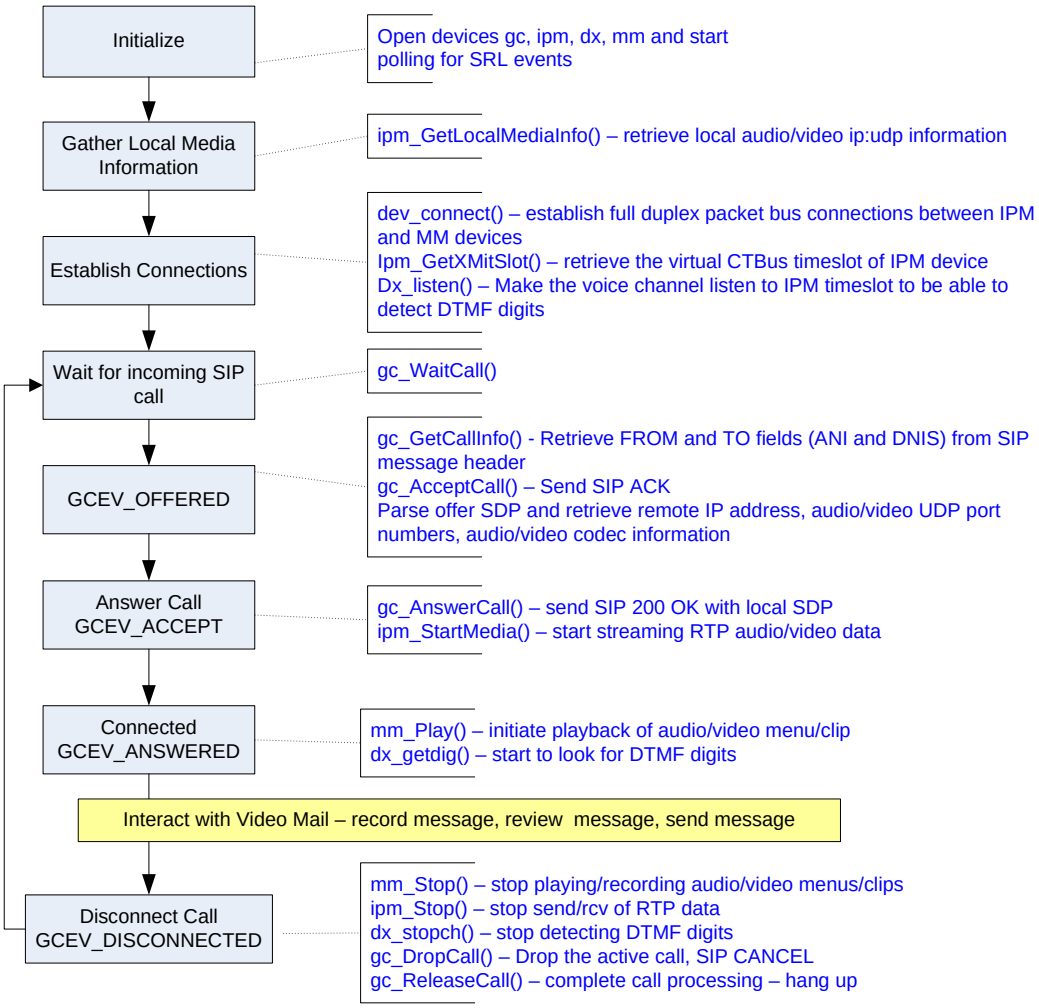
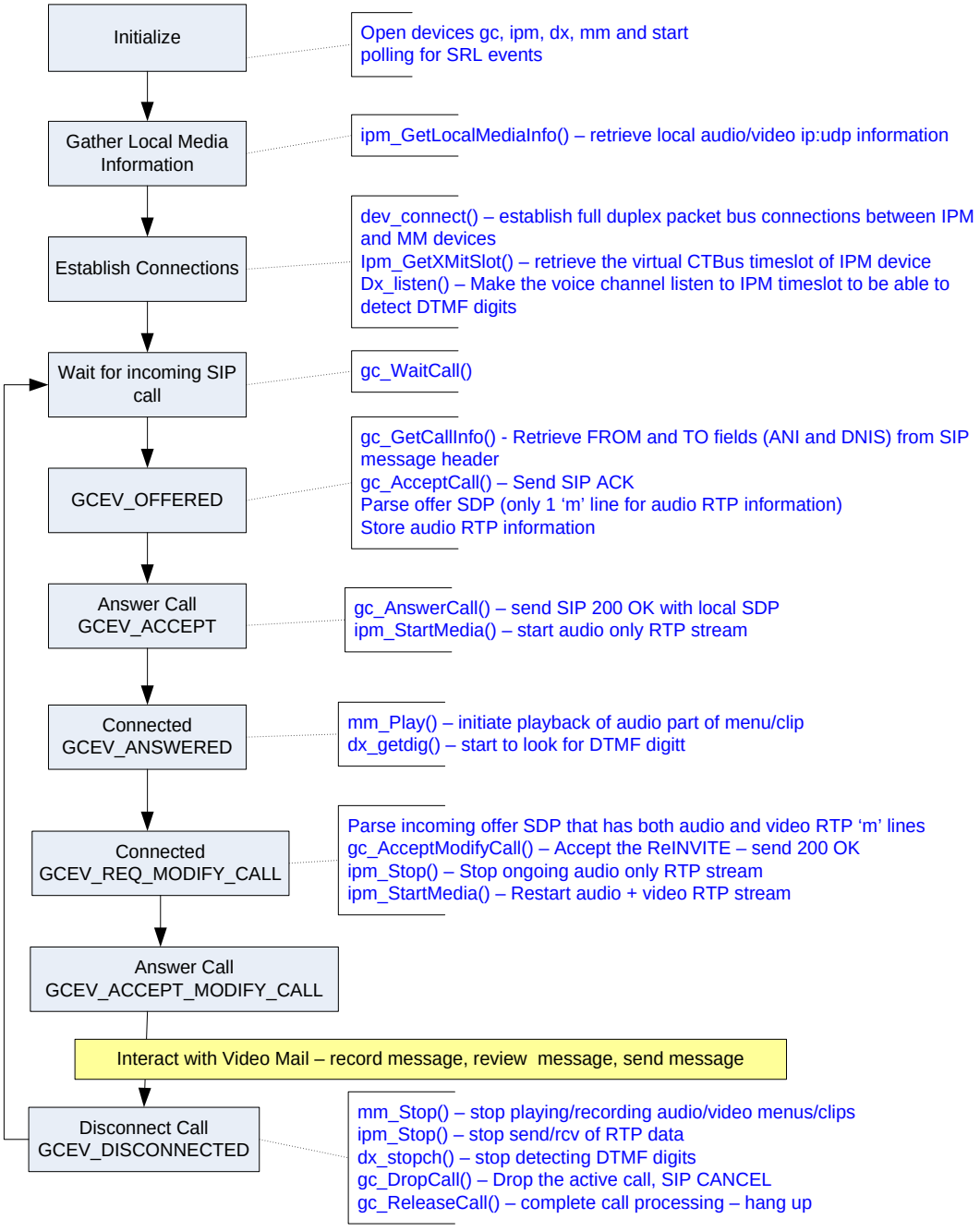


Figure 3 shows a scenario in which the SIP message has only audio first and then with a subsequent re-INVITE the audio plus video information is provided in the SDP.

Figure 3. Incoming SIP Call with Only Audio SDP (video part is sent via re-INVITE)



4.4 Requesting I-Frame Using SIP INFO

As previously discussed in Section 1.1.4, “Intraframe (I-Frame)”, on page 13, Intraframe, or I-frame, is a frame that contains complete information about the scene or picture of video. The other types of frames in the video stream that contain only partial information are called P or B-frames. When a SIP User Agent (UA) detects that its video picture quality is being compromised, it may request the source to send an I-frame so that it can refresh the screen and improve picture quality. This mechanism has been defined by the IETF working group mmusic: Multiparty Multimedia Session Control (draft-levin-mmusic-xml-schema-media-control-03). Although the draft never got ratified and has expired, it has been through a few versions and has already been used in the industry: some SIP phone vendors and SIP servers currently implement it.

Using Global Call SIP in 3PCC mode, an application is able to attach the XML schema as the message body to a SIP INFO method and sent that to a SIP endpoint to request an I-frame. Shown below is the XML schema and a code snippet from the multimedia demo that is part of the HMP software:

```
INFO sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK7
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/media_control+xml
Message Body
<?xml version="1.0" encoding="utf-8" ?>
<media_control>
          <vc_primitive>
                <to_encoder>
          <picture_fast_update>
          </picture_fast_update>
                </to_encoder>
          </vc_primitive>
</media_control>
```

```
//*****************************************************************************
//         NAME : bool CMMStream::SendIFrameRequest()
// DESCRIPTION : Sends the IFrame Request
//        INPUT : None
//        OUTPUT : None
//       RETURNS : Bool - True if the function succeeded, False otherwise
//      CAUTIONS : None
//*****************************************************************************
bool CMMStream::SendIFrameRequest()
{
       // Local Variable Declaration
       GC_PARM_BLKP   gcParmBlk_mime = 0;
       GC_PARM_BLKP   gcParmBlk_mime1 = 0;
       GC_PARM_BLKP   gcParmBlk_info = 0;
       bool        bOk            = true;
       char        *pBodyType     = "Content-Type:application/media_control+xml"; // specify the
body type
```

```
        mmReport(INFO_MSG, s_eType, "SendIFrameRequest()");
        if (gc_util_insert_parm_ref(&gcParmBlk_mime,
                IPSET_MIME,
                IPPARM_MIME_PART_TYPE,
                (unsigned char)(strlen(pBodyType) + 1),
                pBodyType) < 0)
        {
                mmReport(ERROR_GCALL, s_eType, "SendIFrameRequest() -> gc_util_insert_parm_ref()
failed on %s for IPPARM_MIME_PART_TYPE ", m_devName);
                bOk = false;
        }

        // insert the body size
        if (gc_util_insert_parm_val(&gcParmBlk_mime,
                IPSET_MIME,
                IPPARM_MIME_PART_BODY_SIZE,
                sizeof(unsigned long),
                strlen(c_iFrameRequest)) < 0)
        {
                mmReport(ERROR_GCALL, s_eType, "SendIFrameRequest() -> gc_util_insert_parm_val()
failed on %s for IPPARM_MIME_PART_BODY_SIZE ", m_devName);
                bOk = false;
        }

        // insert the body
        if (gc_util_insert_parm_val(&gcParmBlk_mime,
                IPSET_MIME,
                IPPARM_MIME_PART_BODY,
                sizeof(unsigned long),
                (unsigned long)(c_iFrameRequest)) < 0)
        {
                mmReport(ERROR_GCALL, s_eType, "SendIFrameRequest() -> gc_util_insert_parm_val()
failed on %s for IPPARM_MIME_PART_BODY ", m_devName);
                bOk = false;
        }

        // insert the list of parmBlks into the top level parmBlk
        if (gc_util_insert_parm_val(&gcParmBlk_mime1,
                IPSET_MIME,
                IPPARM_MIME_PART,
                sizeof(unsigned long),
                (unsigned long)gcParmBlk_mime) < 0)
        {
                mmReport(ERROR_GCALL, s_eType, "SendIFrameRequest() -> gc_util_insert_parm_val()
failed on %s for IPPARM_MIME_PART", m_devName);
                bOk = false;
        }



        // now set it on the device
        if (gc_SetUserInfo(GCTGT_GCLIB_CRN,
                m_gcCurrentCrn,
                gcParmBlk_mime1,
                GC_SINGLECALL) < 0) // for this call only
        {
                mmReport(ERROR_GCALL, s_eType, "gc_SetUserInfo() failed on %s for MIME body in
INFO");
                bOk = false;
         }
         // insert the message type
         if (gc_util_insert_parm_val(&gcParmBlk_info,
                 IPSET_MSG_SIP,
                 IPPARM_MSGTYPE,
                 sizeof(int),
                 IP_MSGTYPE_SIP_INFO) < 0)
```

```
        {
               mmReport(ERROR_GCALL, s_eType, "SendIFrameRequest() -> gc_util_insert_parm_val()
failed on %s for SIP INFO", m_devName);
               bOk = false;
        }

       if (gc_Extension(GCTGT_GCLIB_CRN, m_gcCurrentCrn, IPEXTID_SENDMSG, gcParmBlk_info, NULL,
EV_ASYNC) < 0)
        {
               mmReport(ERROR_GCALL, s_eType, "SendIFrameRequest() -> gc_Extension failed");
               bOk = false;
        }
        gc_util_delete_parm_blk(gcParmBlk_info);
        gc_util_delete_parm_blk(gcParmBlk_mime);

        return bOk;
}
```

The above code snippet can be found in the *mmstream.cpp* file of the multimedia demo application.

4.5 Making Connections Using Virtual CT Bus and New Packet Bus

This section shows several connection scenarios.

Figure 4 shows a connection diagram for a multimedia play and/or record session. Connections are made between the MM (multimedia) Device and the IPM Device using the **dev_Connect( )** API.

Figure 4. Connection Scenario 1

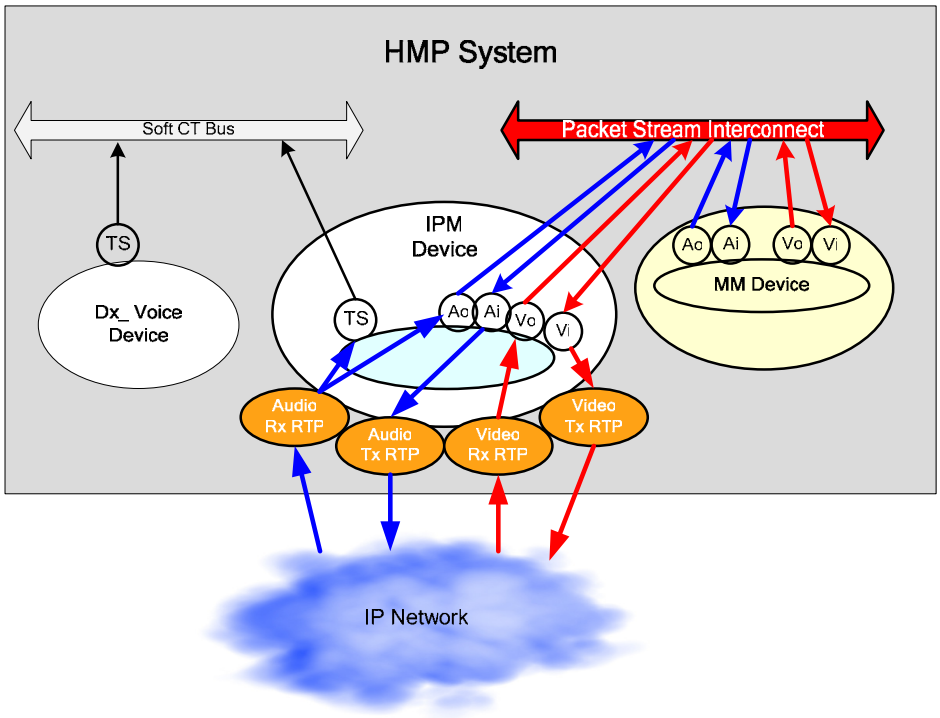


Figure 5 shows a connection diagram for a multimedia play and/or record session in which a dx_ Voice Device is used simultaneously to listen for inband digits received from the audio RTP stream received from the IP Network via the IPM Device. Connections are made between the MM (multimedia) Device and the IPM Device using the **dev_Connect( )** API followed by a **dx_Listen( )** to the IPM Device transmit timeslot.

Figure 5. Connection Scenario 2

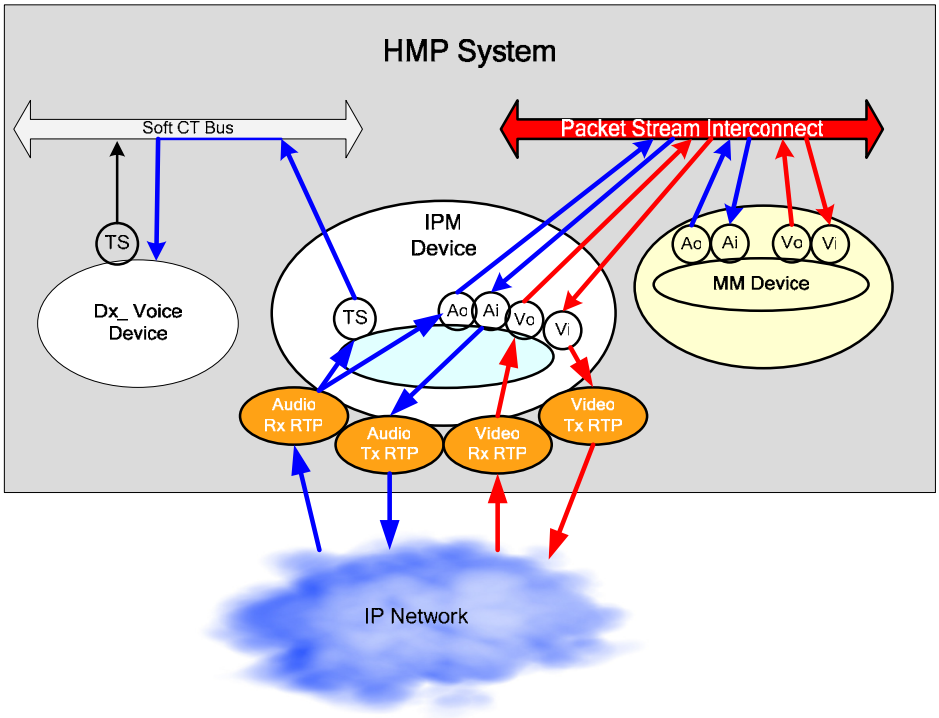
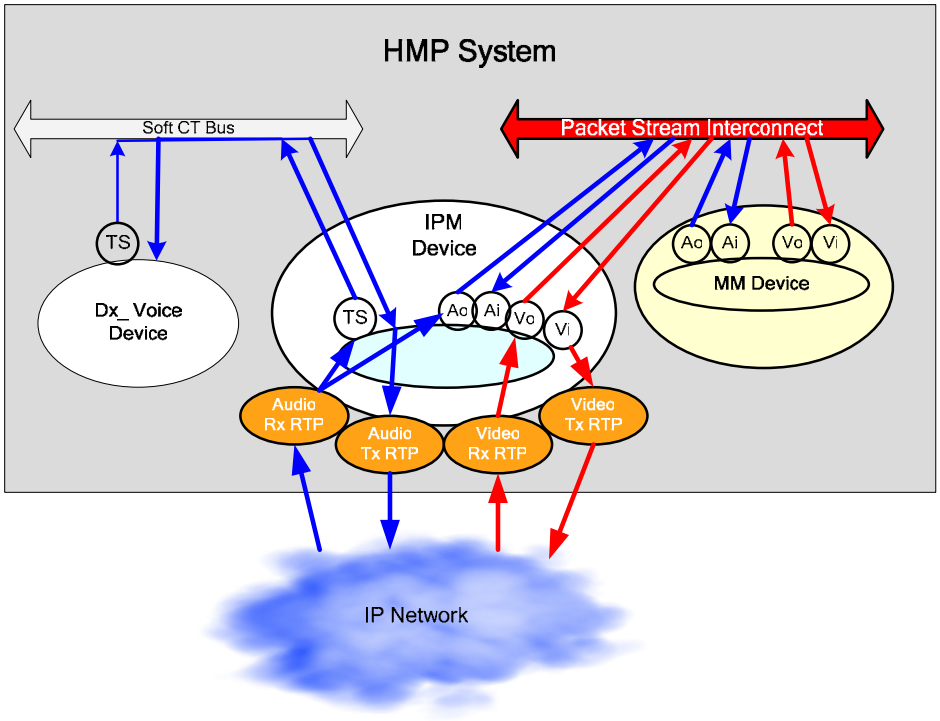


Figure 6 shows a connection diagram for a multimedia play session where the MM (multimedia) Device plays video, the dx_ Voice Device plays audio, and the dx_ Voice Device is also used to listen for inband digits received from the audio RTP stream received from the IP Network via the IPM Device. Connections are first made between the MM Device and the IPM Device using the **dev_Connect( )** API followed by a **dx_Listen( )** to the IPM Device transmit timeslot and an **ipm_Listen( )** to the dx_ Voice Device transmit timeslot. To satisfy this use case, the order of connections *must* be **dev_Connect( )** followed by the **ipm_Listen( )**.

Note: The audio data sent from the audio Tx RTP port of the IPM Device to the IP Network depends on the last active connection established. If an **ipm_UnListen( )** is now called, the source of the audio data will become the MM Device again.

Figure 6. Connection Scenario 3



Note: If calls are recorded using a dx device, you can still play these recordings to a multimedia (audio/video) session. When you do this, the video is not tightly synchronized with the audio since the audio was recorded separately, but you may wish to play a video menu or status display while playing back the audio message. You can do this by making a dev_connect between the IPM device and the MM Device and follow this with an **ipm_Listen( )** to a dx device. Then play the video file using the MM Device and the audio file using the dx_ Voice Device. If you then do an **ipm_UnListen( )**, the audio connection to the IPM device (towards IP) will reconnect to the MM Device as the audio source.

4.6 Enabling Digit Detection Using DX

DTMF digits can be detected using the Voice Library API **dx_getdig( )**. However, to specify the type of digits (inband or RFC 2833), the application should use **ipm_SetParm( )** to specify the type of digits. This information can be retrieved in the SDP body that is provided in the incoming SIP INVITE from the SIP endpoint. The following code snippet shows how this is done in the multimedia demo (the sample program provided in the HMP software release).

Note: For more information about the multimedia demo, refer to the *Multimedia for Host Media Processing Demo Guide*.

```
//*****************************************************************************
//                NAME : bool CMMStream::SetDtmfMode(dtmfMode_e eMode)
// DESCRIPTION : Sets the DTMF XFer Mode based on the input
//               INPUT : eMode - Object of type dtmfMode containing the input
//          OUTPUT : None
//         RETURNS : Bool - True if function succeeds, False otherwise
//        CAUTIONS : None
//*****************************************************************************
bool CMMStream::SetDtmfMode(dtmfMode_e eMode)
{
        // Local Variable Declaration
        IPM_PARM_INFO           parmInfo;
            eIPM_DTMFXFERMODE       value;
        int                   PLType  = 0;
        switch (eMode)
        {

            case dtmfMode_rfc2833:
            {
                 // set the dtmf mode to RFC2833
                            value                                       = DTMFXFERMODE_RFC2833;
                 parmInfo.eParm                 = PARMCH_DTMFXFERMODE;
                            parmInfo.pvParmValue    = &value;

                            mmReport(INFO_MSG, s_eType, "[%s] Setting Parameter
PARMCH_DTMFXFERMODE to DTMFXFERMODE_RFC2833", m_ipmDevName);

                 if (ipm_SetParm(m_ipmH, &parmInfo, EV_SYNC) < 0)
                            {
                    mmReport(ERROR_IPM, s_eType, "ipm_SetParm() on %s", m_ipmDevName);
                            }

                 // set the TX Payload Type

                 PLType                                 = m_nRfc2833PayloadType;
                 parmInfo.eParm                 = PARMCH_RFC2833EVT_TX_PLT;
                 parmInfo.pvParmValue    = &PLType;

                mmReport(INFO_MSG, s_eType, "[%s] Setting Parameter PARMCH_RFC2833EVT_TX_PLT to
%d", m_ipmDevName, m_nRfc2833PayloadType);

                 if (ipm_SetParm(m_ipmH, &parmInfo, EV_SYNC) < 0)
                 {
                    mmReport(ERROR_IPM, s_eType, "ipm_SetParm() on %s", m_ipmDevName);
                 }

                    // set the RX Payload Type
                    PLType                                 = m_nRfc2833PayloadType;
                    parmInfo.eParm                 = PARMCH_RFC2833EVT_RX_PLT;
                    parmInfo.pvParmValue = &PLType;

                   mmReport(INFO_MSG, s_eType, "[%s] Setting Parameter PARMCH_RFC2833EVT_RX_PLT
```

```
to %d", m_ipmDevName, m_nRfc2833PayloadType);

                    if (ipm_SetParm(m_ipmH, &parmInfo, EV_SYNC) < 0)
                    {
                       mmReport(ERROR_IPM, s_eType, "ipm_SetParm() on %s", m_ipmDevName);
                    }
                }
                break;

                case dtmfMode_inband:
                {
                        eIPM_DTMFXFERMODE value = DTMFXFERMODE_INBAND;
                        parmInfo.eParm                  = PARMCH_DTMFXFERMODE;
                        parmInfo.pvParmValue    = &value;

                        mmReport(INFO_MSG, s_eType, "[%s] Setting Parameter PARMCH_DTMFXFERMODE
to DTMFXFERMODE_INBAND", m_ipmDevName);

                        if (ipm_SetParm(m_ipmH, &parmInfo, EV_SYNC) < 0)
                        {
                             mmReport(ERROR_IPM, s_eType, "ipm_SetParm() on %s", m_ipmDevName);

                        }
                }
                break;
        }
        return true;
}
```

Building Applications 5

This chapter provides general information on building applications using the Multimedia API library. The following information is provided:

5.1 Compiling and Linking

The following topics discuss compiling and linking requirements:

- Include Files
- Required Libraries
- Variables for Compiling and Linking

5.1.1 Include Files

The following Multimedia API header file is required when building your C or C++ application: *mmlib.h*

Other Intel® Dialogic® header files may also be required for your application, depending on the functionality. A commonly included header file is the *srllib.h* file with prototypes and definitions for the Standard Runtime Library (SRL).

5.1.2 Required Libraries

The following Multimedia API library is required when building your application: *libmml.so*

Other Intel Dialogic libraries may also be required for your application, depending on the functionality. A commonly used library is *libsrl.so* with implementation of the Standard Runtime Library (SRL).

5.1.3 Variables for Compiling and Linking

When building your applications you may use the following variables to reference the directories that contain header files and libraries:

INTEL_DIALOGIC_INC
: Variable that points to the directory where header files are stored.

INTEL_DIALOGIC_LIB
: Variable that points to the directory where library files are stored.

These variables are automatically set at login and should be used in compiling and linking commands.

Glossary

1PCC: See First Party Call Control.

3G: See Third Generation Mobile System

3GPP: See Third Generation Partnership Program.

3PCC: See Third Party Call Control.

4CIF: Four times the resolution of CIF. See Common Intermediate Format.

16CIF: Sixteen times the resolution of CIF. See Common Intermediate Format.

CIF: See Common Intermediate Format.

Codec: See COder/DECoder.

COder/DECoder: A device or program that can transform a data stream or signal.

Common Intermediate Format (CIF): A standard size for images produced by digital and video cameras. CIF images are 352 pixels wide and 288 pixels tall (352 x 288).

Computer Telephony (CT): Adding computer intelligence to the making, receiving, and managing of telephone calls.

CT: See Computer Telephony.

DTMF: See Dual-Tone Multi-Frequency.

Dual-Tone Multi-Frequency (DTMF): A way of signaling consisting of a push-button or touch-tone dial that sends out a sound consisting of two discrete tones that are picked up and interpreted by telephone switches (either PBXs or central offices).

First Party Call Control (1PCC): First Party Call Control (1PCC) mode is where the Global Call call control library manages IP signaling (SIP or H.323) as well as the RTP media control via the IP Media Library (IPML). IPML is a complementary API that is used to manipulate RTP media streams (start, stop, or modify them). The call control library synchronizes the call control and media control state machines, thus abstracting the application from the details of creating and parsing Session Description Protocol (SDP) bodies that are embedded in SIP messages and invoking the IPML functions to start, stop, or modify RTP streams. See also Third Party Call Control.

Gatekeeper: An H.323 entity on the Internet that provides address translation and control access to the network for H.323 terminals and gateways. The gatekeeper may also provide other services to the H.323 terminals and gateways, such as bandwidth management and locating gateways.

Gateway: A device that converts data into the IP protocol. It often refers to a voice-to-IP device that converts an analog voice stream, or a digitized version of the voice, into IP packets.

Global Call: A unified, high-level API that shields developers from the low-level signaling protocol details that differ in countries around the world. Allows the same application to easily work on multiple signaling systems worldwide (for example, ISDN, T1 robbed bit, R2/MF, pulsed, SS7, IP, H.323).

Global System for Mobile Communications (GSM): one of the leading digital cellular systems. First introduced in 1991, GSM is used in more than 100 countries and has become the standard in Europe and Asia.

GSM: See Global System for Mobile Communications.

GSM-AMR-NB: Global System for Mobile communication Adaptive Multi-Rate Narrow Band

H.248: The ITU-T recommendation for MGCP. See Media Gateway Control Protocol.

H.323: A set of International Telecommunication Union (ITU) standards that define a framework for the transmission of real-time voice communications through Internet protocol (IP)-based packet-switched networks. The H.323 standards define a gateway and a gatekeeper for customers who need their existing IP networks to support voice communications.

I-frame: To create motion in a video, individual frames of pictures are grouped together and played back. An Intraframe (I-frame) is a single frame of digital content that the compressor examines independent of the frames that precede and follow it and stores all of the data needed to display that frame.

Integrated Services Digital Network (ISDN): A collection of standards for defining interfaces and operation of digital switching equipment for voice and data transmissions.

International Telecommunication Union (ITU): An organization established by the United Nations to set telecommunications standards, allocate frequencies to various uses, and hold trade shows every four years.

Internet: An inter-network of networks interconnected by bridges or routers. LANs described in H.323 may be considered part of such inter-networks.

Internet Protocol (IP): The network layer protocol of the transmission control protocol/Internet protocol (TCP/IP) suite. Defined in STD 5, Request for Comments (RFC) 791. It is a connectionless, best-effort packet switching protocol.

Intraframe: See I-frame.

IP: See Internet Protocol.

ISDN: See Integrated Services Digital Network.

ITU: See International Telecommunication Union.

Media Gateway Control Protocol (MGCP): MGCP is a control and signal standard developed to compete with the older H.323 standard for the conversion of audio signals carried on the PSTN to data packets carried over packet networks and the Internet. This standard was developed by Telcordia Technologies and Level 3 Communications. The growing popularity of Voice over IP (VoIP) has prompted the development of new standards. MGCP eliminates the need for complex, processor-intense IP telephony devices, which simplifies and reduces the cost of these terminals.

MGCP: See Media Gateway Control Protocol.

Network Interface Card (NIC): Adapter card inserted into a computer that contains necessary software and electronics to enable a station to communicate over network.

NIC: See Network Interface Card.

PRI: See Primary Rate Interface.

Primary Rate Interface (PRI): A standard digital telecommunication service, available in many countries and most of the United States, that allows the transfer of voice and data over T-1 or E-1 lines. The T-1 ISDN Primary Rate protocol consists of 23 voice/data channels (B-channels) and one signaling channel (D-channel). The E-1 ISDN Primary Rate protocol consists of 30 voice/data channels, one signaling channel (D-channel), and one framing channel to handle synchronization.

PSTN: See Public Switched Telephone Network.

Public Switched Telephone Network (PSTN): The telecommunications network commonly accessed by standard telephones, key systems, Private Branch Exchange (PBX) trunks, and data equipment.

QCIF: See Quarter CIF.

QoS: See Quality of Service.

Quality of Service (QoS): A measure of the telephone service quality provided to the subscriber.

Quarter CIF (QCIF): This is a variation of CIF (see Common Intermediate Format). A standard size for images produced by digital and video cameras. QCIF images are 176 pixels wide and 144 pixels tall (176 x 144).

Real-time Transport Protocol (RTP): Defines a standardized packet format for delivering audio and video over the Internet.

RTP: See Real-time Transport Protocol.

SDP: Session Description Protocol

SIP: Session Initiation Protocol: an Internet standard specified by the Internet Engineering Task Force (IETF) in RFC 3261. SIP is used to initiate, manage, and terminate interactive sessions between one or more users on the Internet.

SQCIF: Sub-QCIF. Half the resolution of QCIF. See Quarter CIF.

Standard Runtime Library (SRL): An Intel® Dialogic® library that contains C functions common to all Intel® Dialogic® devices, a data structure to support application development, and a common interface for event handling.

T1: A digital transmission link with a capacity of 1.544 Mbps used in North America. Typically channeled into 24 digital subscriber level zeros (DS0s), each capable of carrying a single voice conversation or data stream. T1 uses two pairs of twisted pair wires.

TCP: See Transmission Control Protocol.

TDM: See Time Division Multiplexing.

Terminal: An H.323 Terminal is an endpoint on the local area network which provides for real-time, two-way communications with another H.323 terminal, Gateway, or Multipoint Control Unit. This communication consists of control, indications, audio, moving color video pictures, and/or data between the two terminals. A terminal may provide speech only, speech and data, speech and video, or speech, data, and video.

Third Generation Mobile System (3G): 3G is an ITU specification for the next generation of wireless mobile communications networks. 3G is commonly considered a non-disruptive enhancement of the GSM cellular standards. The enhancement includes greater bandwidth (up to 384 Kbps when stationary or moving at pedestrian speed, 128 Kbps in a vehicle, and 2 Mbps from fixed locations), more advanced compression techniques, and the inclusion of in-building systems.

Third Generation Partnership Program (3GPP): The partnership brings together a number of telecommunications standards bodies. These partners have agreed to work together to create technical specifications for a 3rd Generation Mobile System based on the GSM core networks and the radio access technologies that the partners support. For more information, refer to www.3gpp.org and www.etsi.org.

Third Party Call Control (3PCC): In the Third Party Call Control (3PCC) mode, the Global Call call control library provides maximum flexibility to the application allowing it to manipulate both SDP bodies and invoke IPML functions as needed. The Global Call library in this mode is acting as a pure SIP application. The application is responsible for monitoring the various SIP transactions and invoking appropriate IPML functions to start, stop, or modify RTP streaming and create or parse SDP bodies. See also First Party Call Control.

Time Division Multiplexing (TDM): A technique for transmitting a number of separate data, voice, and/or video signals simultaneously over one communications medium by quickly interleaving a piece of each signal one after another.

Transmission Control Protocol (TCP): The TCP/IP standard transport level protocol that provides the reliable, full duplex, stream service on which many application protocols depend. TCP allows a process on one machine to send a stream of data to a process on another. It is connection-oriented in the sense that before transmitting data, participants must establish a connection.

UA: User Agent - In a SIP context, user agents (UAs) are appliances or applications, such as, SIP phones, residential gateways and software that initiate and receive calls over a SIP network.

UDP: See User Datagram Protocol.

User Datagram Protocol (UDP): The TCP/IP standard protocol that allows an application program on one machine to send a datagram to an application program on another machine. Conceptually, the important difference between UDP datagrams and IP datagrams is that UDP includes a protocol port number, allowing the sender to distinguish among multiple destinations on the remote machine.

Index

O

P

R

S

T

V