Dialogic®

Dialogic® Host Media Processing Software Release 1.5LIN

Release Update

August 7, 2008

05-2449-026

Publication Date: August 7, 2008

Document Number: 05-2449-026

About This Publication

This section contains information about the following topics:

Purpose

This Release Update addresses identified issues associated with Dialogic® Host Media Processing (HMP) Software Release for 1.5LIN (also referred to as Dialogic® HMP 1.5LIN). In addition to summarizing issues that are known for the initial release of this product, the Release Update is updated to serve as the primary mechanism for communicating new issues, if any, that arise after the initial release date.

Intended Audience

This Release Update is intended for all users of the Dialogic® Host Media Processing (HMP) Software Release for 1.5LIN.

How to Use This Publication

This Release Update is organized into four sections (click the section name to jump to the corresponding section):

- Document Revision History: This section summarizes the ongoing changes and additions that are made to this Release Update after its original release. This section is organized by document revision and document section.
- Post-Release Developments: This section describes significant changes to the release subsequent to the general availability release date. For example, new features provided in service updates are described here.
- Release Issues: This section lists issues that may affect the system release hardware and software. The lists include both known issues as well as issues that have been resolved since the last release. Also included are any restrictions and limitations that apply to this release, as well as notes on compatibility.
- Documentation Updates: This section contains corrections and other changes that apply to the Dialogic® HMP 1.5LIN documentation set that could not be made to the documents prior to the release. The updates are organized by documentation category and by individual document.

Related Information

See the following for additional information:

- For information about the products and features supported in this release, see the *Dialogic® Host Media Processing Software Release 1.5LIN Release Guide*, which is included as part of the documentation bookshelf for the release.
- For further information on issues that have an associated defect number, you may use the Defect Tracking tool at *http://membersresource.dialogic.com/defects/*. When you select this link, you will be asked to either LOGIN or JOIN.
- *http://www.dialogic.com/support* (for Dialogic technical support)
- *http://www.dialogic.com/* (for Dialogic® product information)

Document Revision History

This revision history summarizes the changes made in each published version of the Release Update for Dialogic® Host Media Processing (HMP) Software Release for 1.5LIN, which is a document that is subject to updates during the lifetime of the release.

Document Rev 26, published August 7, 2008

Updated for **Service Update 132**.

The Release Issues section contains the following changes:

- Added the following item to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00042665.

Document Rev 25, published April 24, 2008

Additional Update for **Service Update 130**.

The Release Issues section contains the following changes:

- Removed the following item from Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00042665.

Document Rev 24, published April 21, 2008

Updated for **Service Update 130**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00042340, IPY00042665.

Document Rev 23, published March 27, 2008

Updated for **Service Update 128**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00042035, IPY00042329, IPY00042528.

Document Rev 22, published March 3, 2008

Updated for **Service Update 127**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00041280, IPY00041296, IPY00041405, IPY00041583, IPY00041686, IPY00041789, IPY00041815, IPY00041876, IPY00042146.

Document Rev 21, published February 21, 2008

Updated for **Service Update 126**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00041836, IPY00042016.

Document Rev 20, published December 13, 2007

Updated for **Service Update 123**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00038706, IPY00039823, IPY00040743, IPY00041118, IPY00041254, IPY00041300.

The Documentation Updates section contains the following change:

- In the Dialogic® Voice API Library Reference, added the **dx_resetch( )** function (IPY00038706).

Document Rev 19, published November 26, 2007

Updated for **Service Update 121**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00036779, IPY00037252, IPY00037386, IPY00037541, IPY00037613, IPY00037679,IPY00037699, IPY00037737, IPY00037778, IPY00037825, IPY00038060, IPY00038150, IPY00038217, IPY00038218, IPY00038240, IPY00038342, IPY00038343, IPY00038365, IPY00038475, IPY00038513, IPY00038549, IPY00038732, IPY00038747, IPY00038827, IPY00038848, IPY00038867, IPY00038868, IPY00038992, IPY00039315, IPY00039325, IPY00039333, IPY00039401, IPY00039409, IPY00039410, IPY00039451, IPY00039505, IPY00039564, IPY00039639, IPY00039677, IPY00039707, IPY00039832, IPY00039847, IPY00039874, IPY00039965, IPY00039985, IPY00040029, IPY00040440, IPY00040524, IPY00041063.

Document Rev 18, published June 29, 2007

Updated for **Service Update 114**.

In the Post-Release Developments section:

- Added information about Support for Turbo Linux 10.0 Server.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00038533.

Document Rev 17, published June 5, 2007

Updated for **Service Update 110**.

In the Post-Release Developments section:

- Added information about Current Operating System Support.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00037542, IPY00037708.

Document Rev 16, published May 1, 2007

Updated for **Service Update 105**.

The Release Issues section contains the following change:

- Added the following item to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00037696.

Document Rev 15, published March 21, 2007

Updated for **Service Update 103**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00036453, IPY00037172, IPY00037333.

Document Rev 14, published March 6, 2007

Updated for **Service Update 100**.

In the Post-Release Developments section:

- Added information about Using Multimedia API for Play and Record.

The Documentation Updates section contains the following changes:

- In the Dialogic® Multimedia API Library Reference, updated the MM_AUDIO_CODEC data structure.

Document Rev 13, published February 22, 2007

Updated for **Service Update 98**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00035819, IPY00035875.

Document Rev 12, published February 9, 2007

Updated for **Service Update 96**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00035585, IPY00035680.

Document Rev 11, published January 5, 2007

Updated for **Service Update 93**.

In the Post-Release Developments section:

- Added information about how to Disable Automatic SDP Message Generation.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00034413, IPY00035477, IPY00035761.
- Added the following item to Table 2, "Known Issues in Dialogic® HMP 1.5LIN", on page 55: IPY00035875.

The Documentation Updates section contains the following changes:

- In the Dialogic® Multimedia API Library Reference, added information about the MM_MEDIA_AUDIO data structure.
- In Dialogic® IP Media Library API Library Reference, added information about the new parameter for disabling SDP message generation.

Document Rev 10, published October 24, 2006

Additional update for **Service Update 87**.

In the Post-Release Developments section, added information about SIP Session Timer.

Document Rev 09, published October 16, 2006

Updated for **Service Update 87**.

The Release Issues section contains the following change:

- Added the following item to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00034415.

Document Rev 08, published August 22, 2006

Updated for **Service Update 84**.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00031519, IPY00034023, IPY00034035.

Document Rev 07, published July 5, 2006

Updated for **Service Update 79**.

Note: This Revision History and the Release Issues sections have been modified to show issues by Change Control System **defect number** and by PTR number. Issues reported prior to March 27, 2006, will be identified by both numbers. Issues reported after March 27, 2006, will only have a defect number.

The Post-Release Developments section contains the following changes:

- Added support for Preserving Data in User Configuration Files.
- Updated existing support for Red Hat Enterprise Linux to include Enterprise Server and Workstation (ES and WS) **Release 4.0** with Update 1 or Update 2. See Section 1.15, "Red Hat Enterprise Linux 4.0", on page 43.

The Release Issues section contains the following changes:

- Added the following item to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00032607.

The Documentation Updates section contains the following changes:

- In the Programming Libraries Documentation section, updated the Dialogic® Global Call IP Technology Guide to correct information on the **gc_MakeCall( )** Variances for IP (IPY00029956 = PTR 36646).
- In the Programming Libraries Documentation section, updated the Dialogic® Global Call IP Technology Guide to add information on support for H.323 Annex M (Tunneled Signaling Message).
- In the Programming Libraries Documentation section, updated the Dialogic® Audio Conferencing API Library Reference and the Dialogic® Audio Conferencing API Programming Guide to indicate that the MSG_ACTID parameter enables/disables

active talker identification (or notification) and not the active talker feature itself. (IPY00006584 = PTR 36199).

- In the Programming Libraries Documentation section, updated the Dialogic® Audio Conferencing API Programming Guide to change the description of how to implement background music in an application so that it doesn't refer to MSG_ACTID. (IPY00010946 = PTR 36323).
- In the Programming Libraries Documentation section, updated the Dialogic® Voice API Library Reference to add cautions on using **dx_listen( )**, **dx_unlisten( )**, **dx_listenEx( )**, and **dx_unlistenEx( )**. (IPY00032425).

Document Rev 06, published March 21, 2006

Updated for **Service Update 69**.

The Post-Release Developments section contains the following changes:

- Added more support for H.323 Annex M (Tunneled Signaling Message). Detailed documentation on this feature will be provided in the next Release Update.

The Release Issues section contains the following changes:

- Added the following items to Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49: IPY00024428 (PTR 36507) and IPY00031796 (PTR 36742).
- Added the following item to Table 2, "Known Issues in Dialogic® HMP 1.5LIN", on page 55: IPY00031514 (PTR 36849).
- Clarified item describing requirement that the /etc/hosts file have valid hostname and IP address. See "General" in Section 2.2, "Restrictions and Limitations", on page 56.

The Documentation Updates section contains the following changes:

- In the Programming Libraries Documentation section, updated the *Dialogic® IP Media Library API Library Reference* to include information on Multicast Receive in the **ipm_StartMedia( )** function. Also made minor clarifications in the **ipm_GetLocalMediaInfo( )** function and the IPMEV_QOS_ALARM event description.
- In the Programming Libraries Documentation section, updated the Dialogic® Global Call IP Technology Guide to change the descriptions for three fields in the IP_H221NONSTANDARD data structure.

Document Rev 05, published January 19, 2006

Updated for **Service Update 64**.

The Post Release Developments section contains the following changes:

- Added support for Red Hat Enterprise Linux Advanced Server (AS) **Release 4.0** with Update 1 or Update 2. See Section 1.15, "Red Hat Enterprise Linux 4.0", on page 43.
- Added support for the Intel Pentium D processor. See Section 1.14, "Intel Pentium D Processor", on page 42.

- Made updates to the Multimedia API file conversion utilities and documentation. See Section 1.13, “Updates to Multimedia File Conversion Tools”, on page 42.

The Release Issues section contains the following changes:

- Added the following items to Table 1, “Issues Resolved in Dialogic® HMP 1.5LIN”, on page 49: IPY00010567 (PTR 36632), IPY00011024 (PTR 36664).
- Added the following item to Table 2, “Known Issues in Dialogic® HMP 1.5LIN”, on page 55: IPY00006380 (PTR 36684).
- Added the following item as a permanent limitation to the Global Call, IP, and IPML category in Section 2.2, “Restrictions and Limitations”, on page 56: IPY00032237 (PTR 35806).

The Documentation Updates section contains the following changes:

- In the Release Documentation section, changed the URL for obtaining the Multimedia File Conversion Utilities.
- In the Programming Libraries Documentation section, updated the *Multimedia API for Linux Programming Guide* from 05-2455-001to 05-2455-002 to accommodate removal of Chapter 5 to the *Multimedia File Conversion Tools User Guide* (05-2453-001).
- In the Programming Libraries Documentation section, updated the *IP Media Library API for Host Media Processing Library Reference* to provide new values for the IPM_MEDIA data structure eMediaType field.

Document Rev 04, published December 28, 2005

Updated for **Service Update 55**.

The Post-Release Developments section includes the following changes:

- Added support for 120 channels of T.38 Fax. See Section 1.16, “120 Channels of T.38 Fax”, on page 43.

The Release Issues section contains the following changes:

- Added the following items to Table 1, “Issues Resolved in Dialogic® HMP 1.5LIN”, on page 49: IPY00032235 (PTR 36405).

Document Rev 03, published December 13, 2005

Updated for **Service Update 52**.

Did not make any documentation changes. This service update does not affect any documentation. This service update contains fixes for various minor issues that were discovered in-house and have not been reported by any customers.

Document Rev 02, published October 2005

Updated for **Service Update 49**.

Added a new section, Post-Release Developments, that describes the new features contained in the Service Update and provides information about installing the Service Update software.

The Documentation Updates section contains the following changes:

- In the Release Documentation section, added new Software Requirements to the *Release Guide* (mainly requirements for SUSE Linux Enterprise Server 9 and Debian GNU/Linux 3.1).
- In the Installation Documentation section, added corrections to the Software Installation Guide.

Document Rev. 01, published September 2005

Initial version of this document.

Post-Release Developments

This section describes significant changes to the Dialogic® Host Media Processing Software Release 1.5 Linux subsequent to the general availability release date.

1.1 Service Update for Dialogic® Host Media Processing Software Release 1.5LIN

This Release Update documents the features of the current service update.

Service updates provide fixes to known issues and may also introduce new functionality. Periodically, new versions of the service update may be released.

Note: See the Document Revision History for a summary of the current service update contents, including features and fixes. If you do not need this support, it is not necessary to install the service update.

1.2 Support for Turbo Linux 10.0 Server

With this Service Update, the Dialogic® Host Media Processing Software Release 1.5 Linux has been tested and certified to support the Turbo Linux 10.0 Server.

1.2.1 Feature Description

Turbo Linux 10.0 is a second-generation version of the Linux operating system distribution (OSD), which incorporates the latest advances available in the 2.6 kernel, in addition to a suite of feature rich open source and commercial software packages.

Prior to this release, the Dialogic® HMP Software Release 1.5LIN was certified for support with several commercially available Linux OSDs. In order to support Turbo Linux 10.0 Server, the Dialogic® HMP Software components were validated for interoperating with Turbo Linux 10.0 to provide component compatibility, feature functionality, resource densities and system performance. This validation was comparable to the quality metrics and acceptance criteria used to affirm product interoperability certification for all supported Linux OSDs, including Red Hat, SuSe, and Debian.

1.2.2 Installation

Refer to **http://www.turbolinux.com/products/server/10s/docs/install_guide/** for specific Turbo Linux 10.0 Server installation instructions.

Note: Not all versions of Turbo Linux 10.0 Server are Linux Standard Base (LSB) compliant. The install procedure has dependencies on LSB supplied functions and, if not present, the install is aborted. A specific version of Turbo Linux that is LSB compliant was provided by the target customer and used for our testing. This is the only version that is supported by Dialogic® HMP Software.

1.2.3 Configuration

The following can facilitate the consistent performance and functionality with Dialogic® HMP Software installed on Turbo Linux 10.0 Server as OSDs presently support.

- Kernel
  Kernel preemption is a feature of kernel version 2.6.x to ensure that system responsiveness and mission critical tasks are executed. It should be enabled to allow HMP to perform preemption during low level media processing operations.
- SE Linux
  During installation of the OSD, SE Linux can be configured to operate in any of three modes:

  Disable (feature is turned off)
  Warning Log (feature will log warning messages only)
  Enforcement (use targeted policies for confined set of daemons)

It is recommended that the SE Linux operational mode is set to either “Disable” or “Warning Log” if SNMP is used.

1.2.4 Documentation

The online bookshelf provided with Dialogic® HMP Software Release 1.5LIN contains information about the release features including features for application development, configuration, administration, and diagnostics.

For more information about system requirements and supported OSDs, see the following documents:

- *Dialogic® Host Media Processing Software Release 1.5LIN Release Guide*
- *Dialogic® Host Media Processing Software Release 1.5LIN Software Installation Guide*

1.3 Current Operating System Support

This release supports the following operating systems:

- Red Hat Enterprise Linux 3.0. Update 1
- Red Hat Enterprise Linux 3.0. Update 3
- Red Hat Enterprise Linux 4.0. Update 1
- Red Hat Enterprise Linux 4.0. Update 2
- Red Hat Enterprise Linux 4.0. Update 3
- Red Hat Enterprise Linux 4.0. Update 4
- Debian GNU/Linux 3.1
- SUSE Linux Professional 9.2
- SUSE Linux Enterprise Server 9 (no updates)

Refer to the *Dialogic® Host Media Processing Software Release 1.5LIN Software Installation Guide* for more information.

1.4 Using Multimedia API for Play and Record

With the Service Update, you can use the Dialogic® Multimedia API for playing and recording multimedia. The application can specify the encoding/decoding data scheme to be used when playing or recording multimedia data streams from/to a file in 3PCC mode.

1.4.1 Feature Description

Currently, the application uses a supplementary voice resource per channel for playing and recording audio file formats currently not supported by the multimedia resources. This method adds undesirable complexity to applications because it requires the use of the

Dialogic® Multimedia API (mm_) and Dialogic® Voice API (dx_), which increases the cost per channel. With the new feature, the API has been expanded to support the multimedia device and MM API by exposing an existing coding format, MM_DATA_FORMAT_ALAW. When the **mm_Play( )** and **mm_Record ( )** functions are invoked, they reference this coding format that specifies the characteristics of the audio coder for playing and recording multimedia.

Notes:1. This feature is applicable to Play and Record mode only. It is not applicable for any Native mode operation (that is, RTP play/record, RTP pass-through play and record).

2. Not all functionality available via the existing voice library and resource is supported for play/record of A-law format files. Features such as AGC, SCR, Speed control, volume control, VAD and others that are presently supported for voice resources (dx_), are not supported by the multimedia library and resource.

1.4.2 New Values for Data Structure

The *mmlib.h* header file contains the values that are sent to the multimedia resource when playing/recording using the **mm_Play( )** or **mm_Record( )** functions:

```
*
 * Voice Data format
 */
#define MM_DATA_FORMAT_DIALOGIC_ADPCM                0x1   /* OKI ADPCM (RFU) */
#define MM_DATA_FORMAT_ALAW                          0x3   /* alaw PCM (RFU) */
#define MM_DATA_FORMAT_G726                          0x4   /* G.726 (RFU) */
#define MM_DATA_FORMAT_MULAW                         0x7   /* mulaw PCM (RFU) */
#define MM_DATA_FORMAT_PCM                           0x8   /* PCM */
#define MM_DATA_FORMAT_G729A                         0x0C  /* CELP Coder (RFU) */
#define MM_DATA_FORMAT_GSM610                        0x0D  /* Microsoft GSM (RFU) */
#define MM_DATA_FORMAT_GSM610_MICROSOFT              0x0D  /* Microsoft GSM (RFU) */
#define MM_DATA_FORMAT_GSM610_ETSI                   0x0E  /* ETSI Standard Framing (RFU) */
#define MM_DATA_FORMAT_GSM610_TIPHON                 0x0F  /* ETSI TIPHON Bit Order (RFU) */
#define MM_DATA_FORMAT_TRUESPEECH                    0x10  /* TRUESPEECH Coder (RFU) */
#define MM_DATA_FORMAT_RFU1                          MM_DATA_FORMAT_TRUESPEECH /* Reserved 1
(RFU) */
#define MM_DATA_FORMAT_G711_ALAW                     MM_DATA_FORMAT_ALAW /* (RFU) */
#define MM_DATA_FORMAT_G711_ALAW_8BIT_REV            0x11  /* (RFU) */
#define MM_DATA_FORMAT_G711_ALAW_16BIT_REV           0x12  /* (RFU) */
#define MM_DATA_FORMAT_G711_MULAW                    MM_DATA_FORMAT_MULAW  /* (RFU) */
#define MM_DATA_FORMAT_G711_MULAW_8BIT_REV           0x13  /* (RFU) */
#define MM_DATA_FORMAT_G711_MULAW_16BIT_REV          0x14  /* (RFU) */
#define MM_DATA_FORMAT_G721                          0x15  /* (RFU) */
#define MM_DATA_FORMAT_G721_8BIT_REV                 0x16  /* (RFU) */
#define MM_DATA_FORMAT_G721_16BIT_REV                0x17  /* (RFU) */
#define MM_DATA_FORMAT_G721_16BIT_REV_NIBBLE_SWAP    0x18  /* (RFU) */
#define MM_DATA_FORMAT_IMA_ADPCM                     0x19  /* (RFU) */
#define MM_DATA_FORMAT_FFT                           0xFF  /* fft data (RFU) */
```

1.4.3 Code Example

```
#include <mmlib.h>
int main(int argc, char* argv[])
{
.
.
/*
* ASSUMPTION: A valid nDeviceHandle was obtained from prior call to mm_Open( ).
```

```
*/
MM_PLAY_INFO play_info;
play_info.unVersion = MM_PLAY_RECORD_INFO_VERSION_0;
MM_PLAY_RECORD_LIST playlist[2];
MM_MEDIA_ITEM_LIST mediaitemlist1[1];
MM_MEDIA_ITEM_LIST mediaitemlist2[1];
const MM_VIDEO_CODEC VideoCodecType1 = {
MM_VIDEO_CODEC_VERSION_0,
EMM_VIDEO_CODING_DEFAULT,
EMM_VIDEO_PROFILE_DEFAULT,
EMM_VIDEO_LEVEL_DEFAULT,
EMM_VIDEO_IMAGE_WIDTH_DEFAULT,
EMM_VIDEO_IMAGE_HEIGHT_DEFAULT,
EMM_VIDEO_BITRATE_DEFAULT,
EMM_VIDEO_FRAMESPERSEC_DEFAULT
};
const MM_AUDIO_CODEC AudioCodecType1 = {
MM_AUDIO_CODEC_VERSION_0,
MM_DATA_FORMAT_ALAW, /* play Alaw format */
MM_DRT_8KHZ,
8 /* 8-bit */
};
const char VideoFileName1[] = "/dir/file1.vid";
const char AudioFileName1[] = "/dir/file3.aud";
int cc;
int xx;


cc = 0;
// Build Video Item 1
mediaitemlist1[cc].unVersion = MM_MEDIA_ITEM_LIST_VERSION_0;
mediaitemlist1[cc].ItemChain = EMM_ITEM_EOT;
mediaitemlist1[cc].item.video.codec = VideoCodecType1;
mediaitemlist1[cc].item.video.unMode = 0;
mediaitemlist1[cc].item.video.szFileName = VideoFileName1;
cc++;
xx = 0;

// Add Video Items to the PlayList
playlist[xx].unVersion = MM_PLAY_RECORD_LIST_VERSION_0;
playlist[xx].ItemChain = EMM_ITEM_CONT;
playlist[xx].ItemType = EMM_MEDIA_TYPE_VIDEO;
playlist[xx].list = mediaitemlist1;
xx++;
cc = 0;

// Build Audio Item 1
mediaitemlist2[cc].unVersion = MM_MEDIA_ITEM_LIST_VERSION_0;
mediaitemlist2[cc].ItemChain = EMM_ITEM_EOT;
mediaitemlist2[cc].item.audio.codec = AudioCodecType1;
mediaitemlist2[cc].item.audio.unMode = MM_MODE_AUD_FILE_TYPE_VOX;
mediaitemlist2[cc].item.audio.ulOffset = 0;
mediaitemlist2[cc].item.audio.szFileName = AudioFileName1;
cc++;

// Add Audio Items to the PlayList
playlist[xx].unVersion = MM_PLAY_RECORD_LIST_VERSION_0;
playlist[xx].ItemChain = EMM_ITEM_EOT;
playlist[xx].ItemType = EMM_MEDIA_TYPE_AUDIO;
playlist[xx].list = mediaitemlist2;
xx++;

// Form Play Info
play_info.eFileFormat = EMM_FILE_FORMAT_PROPRIETARY;
play_info.list = playlist;
// Initiate Play
```

```
If (mm_Play(nDeviceHandle, &play_info, NULL, NULL) == EMM_ERROR)
{
/* process error */
}
}
```

1.4.4 Documentation

The online bookshelf provided with Dialogic® HMP Software Release for 1.5LIN contains information about the release features including features for application development, configuration, administration, and diagnostics.

For more information about the Dialogic® Multimedia API, see the following documents:

- *Dialogic® Multimedia API Programming Guide*
- *Dialogic® Multimedia API Library Reference*

1.5 Disable Automatic SDP Message Generation

With the Service Update, you have a new parameter that allows the application to disable the automatic inclusion of session description protocol (SDP) message generation in response to a SIP OPTIONS request in first party call control (1PCC) mode.

1.5.1 Feature Description

Currently, the default behavior of the Dialogic® Global Call API library is to send automatic responses to incoming SIP OPTIONS requests with an SDP message body. You can use this new parameter ID, **IPPARM_1PCC_OPTIONS_RESPONSE_NO_SDP**, with **gc_util_insert_parm_val( )** to disable inclusion of an SDP message with the OPTIONS response in 1PCC mode.

Code Example

The following example shows how to use the parameter with **gc_util_insert_parm_val( )** to disable inclusion of an SDP message body in the OPTIONS response.

```
GC_PARM_BLKP pParmBlock = NULL;
long t = 0;
gc_util_insert_parm_val(&pParmBlock, IPSET_CONFIG, IPPARM_1PCC_OPTIONS_RESPONSE_NO_SDP,
sizeof(int), 0);

int  rc = gc_SetConfigData(GCTGT_CCLIB_NETIF,
                BoardDev,
                pParmBlock,
                0,
                GCUPDATE_IMMEDIATE,
                &t,
                EV_ASYNC);
gc_util_delete_parm_blk(pParmBlock);
```

1.5.2 Documentation

The online bookshelf provided with Dialogic® HMP Software Release 1.5LIN contains information about the release features including features for application development, configuration, administration, and diagnostics.

For more information about Global Call APIs, see the following documents:

- *Dialogic® Global Call API Library Reference*
- *Dialogic® Global Call IP Technology Guide*

1.6 SIP Session Timer

With the Service Update, the user can set the session duration to keep the Session Initiation Protocol (SIP) sessions active. The feature includes a keepalive mechanism to set the value for the length and the minimum time of the session and to refresh the duration of the call; and allows User Agents and proxies to determine if the SIP session is still active.

1.6.1 Feature Description

The feature provides a keepalive mechanism for the SIP to determine whether a session is still active using an extension defined in RFC 4028. The session timer uses three new fields in the IP_VIRTBOARD data structure to enable the timer, and to set the values for session expiration and minimum time. For negotiation between the user agent server (UAS) and user agent client (UAC), the session timer uses new SIP IP parameters available via the Global Call API.A Session-Expires header included in the 2xx response of an initial INVITE determines the session duration. Periodic refresh enables extending the duration of the call and allows both User Agents and proxies to determine if the SIP session is still active. The refresh of a SIP session is done through a re-INVITE or UPDATE. The Session-Expires header in the 2xx response also indicates which side will be the refresher; that is, the side responsible for generating the refresh request. The refresher can be the UAC or UAS. The refresher should generate a refresh request (using re-INVITE or UPDATE) before the session expires. If no refresh request is sent or received before the session expires, or if the refresh request is rejected, both parties should send BYE. The session timer feature has two new headers, Session-Expires and Min-SE, and a new response code of 422. The Session-Expires header conveys the lifetime of the session, the Min-SE header conveys the minimum value for the session timer, and the 422 response code indicates that the session timer duration is too small. If the Min-SE header is missing, the minimum value for the session timer is 90 seconds by default, in compliance with RFC 4028.

Note: The session timer applies to both 1PCC and 3PCC mode.

1.6.2 Session Timer Virtual Board Configuration

There are three new fields for the IP_VIRTBOARD data structure, which can be adjusted for each virtual board:

E_SIP_SessionTimer_Enabled
: Enables session timer. Default is disabled. Each board has to have IP_VIRTBOARD enabled for it.

SIP_SessionTimer_SessionExpires
: Sets the session expires value. Used in timer negotiation for outgoing and incoming calls. Default timer value is 1800 seconds for all calls enabled.

SIP_SessionTimer_MinSE
: Sets the minimum session timer value. Used in timer negotiation for outgoing and incoming calls. Default timer value is 90 seconds for all calls enabled. A 422 response code indicates that the session timer duration is too small. That response contains a Min-SE header field identifying the minimum session interval it is willing to support.

Note: If the session timer is not enabled on the virtual board, the UPDATE method is not supported. Incoming UPDATE message is responded with a *501 Not Implemented* message.

1.6.3 Session Timer Negotiation

Session timer negotiation between UAS and UAC follows RFC 4028. The following SIP IP parameters have been added via the Dialogic® Global Call API for this feature. All parameters can be set on the board device, line device, or call reference number (CRN).

IPPARM_SESSION_EXPIRES
: The parameter value overwrites the configured SIP_SessionTimer_SessionExpires value on virtual board. When the application makes a new call or answers an incoming call, Global Call uses this value to negotiate session interval when functioning as either a UAC or UAS. **IPPARM_SESSION_EXPIRES** sets timer value in seconds.

IPPARM_MIN_SE
: The parameter value overwrites the configured SIP_SessionTimer_MinSE value on the virtual board. When the application makes a new call or answers an incoming call, Global Call uses this value to negotiate session interval when functioning as either a UAC or UAS. **IPPARM_MIN_SE** sets timer value in seconds.

Note: SIP_SessionTimer_MinSE in the virtual board is always used automatically to reject INVITE whose Session-Expires value is too small. The automatic rejection with 422 is not affected by this parameter because once the value is set in **gc_Start( )**, you can not use IPPARM_MIN_SE to change it for incoming calls. If rejected by 422 response, the application receives GCEV_DISCONNECTED with cause IPEC_SIPReasonStatus422SessionIntervalTooSmall.

IPPARM_REFRESHER_PREFERENCE
: The possible values are as follows:

- IP_REFRESHER_LOCAL

- IP_REFRESHER_REMOTE
- IP_REFRESHER_DONT_CARE (default)

If set to local or remote refresher preference, the refresher parameter is added in the Session-Expires header. If set to don't care, the refresher parameter is not added in the Session-Expires header in outgoing INVITE, and default refresher is selected, according to RFC 4028.

The actual refresher is decided by UAS and appears on 200 OK. The following table lists the refresher results if both UAS and UAC are Global Call applications using this refresher preference parameter.

UAC preference	UAS preference	Refresher
IP_REFRESHER_DONT_CARE	IP_REFRESHER_DONT_CARE	UAS
IP_REFRESHER_REMOTE	IP_REFRESHER_REMOTE	UAS
IP_REFRESHER_LOCAL	IP_REFRESHER_LOCAL	UAC
IP_REFRESHER_LOCAL	IP_REFRESHER_REMOTE	UAC
IP_REFRESHER_REMOTE	IP_REFRESHER_LOCAL	UAS

IPPARM_REFRESH_METHOD

The possible values are as follows:

- IP_REFRESH_REINVITE (default)
- IP_REFRESH_UPDATE

If selected as refresher, Global Call sends either a re-INVITE or UPDATE message to periodically refresh the session.

IPPARM_REFRESH_WITHOUT_REMOTE_SUPPORT

The possible values are as follows:

- IP_REFRESH_WITHOUT_REMOTE_SUPPORT_DISABLE
- IP_REFRESH_WITHOUT_REMOTE_SUPPORT_ENABLE (default)

The session timer mechanism can operate as long as one of the two User Agents in the call leg supports the timer extension. As call originator or terminator, the application may choose to execute the session timer if the remote side does not support the session timer. If enabled, Global Call operates the session timer if the remote side does not support session timer and sends the refresh. The other side sees the refreshes as repetitive re-INVITEs. If disabled, Global Call does not operate the session timer if the remote side does not support the session timer. When session timer is supported on only one side, re-INVITE is the only method supported in order to refresh the session. If UA uses UPDATE to refresh the session, it gets a *501 Not Implemented* message back. It is up to the application to decide if it wants to terminate the session or not once the session expires through the use of **IPPARM_TERMINATE_CALL_WHEN_EXPIRES**.

IPPARM_REFRESH_WITHOUT_PREFERENCE

The possible values are as follows:

- IP_REFRESH_WITHOUT_PREFERENCE_DISABLE
- IP_REFRESH_WITHOUT_PREFERENCE_ENABLE (default)

When the 2xx final response is received, but the refresher preference does not match the call refresher, the application may choose to execute the session timer. If enabled,

Global Call operates the session timer mechanism and the refresher is different from the application preference. If disabled, Global Call does not operate the session timer.

IPPARM_TERMINATE_CALL_WHEN_EXPIRES

The possible values are as follows:

- IP_TERMINATE_CALL_WHEN_EXPIRES_DISABLE
- IP_TERMINATE_CALL_WHEN_EXPIRES_ENABLE (default)

When the session timer is about to expire, Global Call sends GCEV_SIP_SESSION_EXPIRES event to application. If enabled, Global Call sends BYE to terminate the call. If disabled, Global Call does not send BYE and the call stays connected.

1.6.4 Global Call IP Defines

New data structure definitions in the gcip_defs.h and gclib.h files are used by the **gc_SetUserInfo( )**, **gc_SetConfigData( )**, **gc_MakeCall( )**, and **gc_AnswerCall( )** functions to determine the duration of the call.

Event	Type	Description
GCEV_SIP_SESSION_EXPIRES	Global Call Event	Global Call event notifies application that SIP session is about to expire

Enumeration	Type	Description
IPEC_SIPReasonStatus422SessionIntervalTooSmall	eIP_EC_TYPE	eIP_EC_TYPE enumeration for SIP response code 422 Session Internal Too Small

Parameter ID	Data Type & Size	Description
IPSET_SIP_SESSION_TIMER	Global Call Set ID	Set ID used to configure SIP session timer
IPPARM_SESSION_EXPIRES	Global Call Parameter ID Type: int Size: sizeof(int)	Session-Expires timer value in seconds
IPPARM_MIN_SE	Global Call Parameter ID Type: int Size: sizeof(int)	Min-SE timer value in seconds. IPPPARM_MIN_SE does not affect Global Call decision to automatically reject incoming call with 422 Session Interval Too Small. Automatically reject decision is based only on SIP_SessionTimer_MinSE from virtual board parameter
IPPARM_REFRESHER_PREFERENCE	Global Call Parameter ID	Refresher preference
IP_REFRESHER_LOCAL	Parameter value	The User Agent wishes to be the refresher.

Parameter ID	Data Type & Size	Description
IP_REFRESHER_REMOTE	Parameter value	The User Agent wishes the remote side to be the refresher.
IP_REFRESHER_DONT_CARE	Parameter value	The User Agent does not care who is the refresher.
IPPARM_REFRESH_METHOD	Global Call Parameter ID	Method for refresh
IP_REFRESH_REINVITE	Parameter value	The refresh method is re-INVITE.
IP_REFRESH_UPDATE	Parameter value	The refresh method is UPDATE.
IPPARM_REFRESH_WITHOUT_REMOTE_SUPPORT	Global Call Parameter ID	When 2xx final response was received, but the server does not support the session timer. The application may choose to execute session timer. The session timer mechanism can be operated as long as one of the two User Agents in the call leg supports the extension. If the application decides to operate the session timer, that side sends the refresh. The other side sees the refreshes as repetitive re-INVITEs. The default behavior is to execute the session timer mechanism for the call.
IP_REFRESH_WITHOUT_REMOTE_SUPPORT_DISABLE	Parameter value	Not to execute session timer without remote support.
IP_REFRESH_WITHOUT_REMOTE_SUPPORT_ENABLE	Parameter value	Execute session timer without remote support.
IPPARM_REFRESH_WITHOUT_PREFERENCE	Global Call Parameter ID	When 2xx final response was received, but refresher preference did not match the call refresher. The application may choose to execute the session timer. If the application decides to operate the session timer mechanism, the refresher is different from the application preference. The default behavior is to execute the session timer mechanism for the call.
IP_REFRESH_WITHOUT_PREFERENCE_DISABLE	Parameter value	Not to execute session timer without preference.
IP_REFRESH_WITHOUT_PREFERENCE_ENABLE	Parameter value	Execute session timer without preference.
IPPARM_TERMINATE_CALL_WHEN_EXPIRES	Global Call Parameter ID	When the session time is about to expire. Application may choose to send BYE to terminate the call. The default behavior is to terminate the call.
IP_TERMINATE_CALL_WHEN_EXPIRES_DISABLE	Parameter value	Not to terminate the call when the session time is about to expire.
IP_TERMINATE_CALL_WHEN_EXPIRES_ENABLE	Parameter value	Terminate the call when the session time is about to expire.

1.6.5 Code Example

This example shows configuration of default Session-Expires and Min-SE timer values on entire virtual board using **gc_Start( )**.

```
#include "gclib.h"
..
..
#define BOARDS_NUM 1
..
..
/* initialize start parameters */
IPCCLIB_START_DATA cclibStartData;
memset(&cclibStartData,0,sizeof(IPCCLIB_START_DATA));
IP_VIRTBOARD virtBoards[BOARDS_NUM];
memset(virtBoards,0,sizeof(IP_VIRTBOARD)*BOARDS_NUM);

/* initialize start data */
INIT_IPCCLIB_START_DATA(&cclibStartData, BOARDS_NUM, virtBoards);

/* initialize virtual board */
INIT_IP_VIRTBOARD(&virtBoards[0]);

/* Enable session timer and configure default parameters */
virtBoard[0].E_SIP_SessionTimer_Enabled= ENUM_Enabled;
virtBoard[0].SIP_SessionTimer_SessionExpires= 3600;
virtBoard[0].SIP_SessionTimer_MinSE= 1800;
```

1.6.6 Documentation

The online bookshelf provided with Dialogic® HMP software contains information about the release features including features for application development, configuration, administration, and diagnostics.

- *Dialogic® Global Call IP Technology Guide*
- *Dialogic® Global Call API Library Reference*
- *Dialogic® IP Media Library API Programming Guide*
- *Dialogic® IP Media Library API Library Reference*
- *Internet Engineering Task Force (IETF) Request for Comments RFC 4028, Session Timers in the Session Initiation Protocol (SIP), http://ietf.org/rfc/rfc4028*

1.7 H.323 Annex M (Tunneled Signaling Message)

The service update provides additional Dialogic® IP Media Library support for H.323 Annex M (Tunneled Signaling Message). See the following information for details on these enhancements. This information also updates existing contents in the *Dialogic® Global Call IP Technology Guide*, primarily in section 4.20 on Tunneled Signaling Messages, but also in the chapter on Parameters (specifically section 9.2.29) and the chapter on Data Structures (including new IP_TUNNELPROTOCOL_OBJECTID data structure).

Using H.323 Annex M Tunneled Signaling Messages

The Global Call IP call control library supports the tunneled signaling message (TSM) capability that is documented in Annex M of the ITU-T recommendations for H.323. This capability allows DSS/QSIG/ISUP messages to be encapsulated in common H.225 call signaling messages. Note that this tunneled message capability is separate and distinct from H.245 tunnelling.

The tunneled signaling message capabilities are described in the following topics:

- Tunneled Signaling Message Overview
- Enabling Tunneled Signaling Messages
- Composing Tunneled Signaling Messages
- Sending Tunneled Signaling Messages
- Receiving Tunneled Signaling Messages
- Reference Information for TSM Parameters

Tunneled Signaling Message Overview

The ITU-T Annex M recommendation specifies that tunneled signaling message fields may be contained in any of nine different H.225 messages: Setup, Call Proceeding, Alerting, Connect, Release Complete, Facility, Progress, Information, and Notify. The Global Call implementation of tunneled signaling messages allows applications to send and receive tunneled messages in the first six of the listed H.225 messages. The Global Call library does not support application access to the last three messages in the list of messages specified in Annex M (Progress, Information, and Notify) so these message types cannot be used for tunneled signaling messages.

The Global Call library supports the ability to send and receive tunneled signaling messages in supported H.225 message types as an optional feature that is disabled by default for backwards compatibility. The ability to send and receive TSMs can only be enabled when starting the system; once enabled, the tunneled signaling message feature cannot be disabled without restarting the system. The feature can be enabled for any virtual board by setting a specific bitmask value in a field of the appropriate IP_VIRTBOARD data structure; see the "Enabling Tunneled Signaling Messages" section for details. When the feature is enabled, applications can use the standard Global Call parameter mechanism to set up a TSM to be sent in the next outgoing H.225 message. To receive a TSM, the application requests the Global Call library to forward the TSM after it has received a Global Call state change event that is associated with one of the supported message types. For most H.225 message types, the application uses **gc_Extension( )** to request the TSM contents which the library returns via a GCEV_EXTENSIONCMPLT asynchronous completion event. In the singular case of the Facility message, the tunneled signaling message content is provided via the unsolicited GCEV_EXTENSION event that notifies the application of the Facility message itself.

An application has no ability to specify which H.225 message types it wishes to receive tunneled signaling message in, and should therefore be prepared to handle TSMs contained in any of the specified H.225 message types so that TSMs are not lost.

Applications construct a tunneled signaling message by constructing a GC_PARM_BLK composed of Global Call parameter elements that contain the TSM protocol identification and

message content. The TSM protocol identification can use either a protocol object ID, specified in an IP_TUNNELPROTOCOL_OBJECTID data structure, or alternate identification data, specified in an IP_TUNNELPROTOCOL_ALTID structure. Only one TSM can be sent per H.225 message.

A tunneled signaling message can also include nonstandard data. The nonstandard data is handled as additional parameter elements in the same GC_PARM_BLK that contains the TSM. As in other Global Call implementations of nonstandard data, the protocol used for the nonstandard data in a TSM can be identified by either H.221 protocol or a protocol object ID. Only one nonstandard data element can be sent per tunneled signaling message.

The maximum data length for the Global Call parameters used for the tunneled signaling message content and the optional nonstandard data content is configured at system start-up. The maximum data length for these parameters is configured by setting the max_parm_data_size field in the IPCCLIB_START_DATA structure. The default size is 255 bytes (for backwards compatibility), but applications may configure it to be as large as 4096 bytes. Applications *must* use the extended **gc_util_..._ex( )** functions to insert or extract any GC_PARM_BLK parameter elements whose data length has been configured to be greater than 255 bytes.

Note: In practice, applications may not be able to utilize the full maximum length of the tunneled signaling message content parameter element as configured in max_parm_data_size, particularly if the tunneled signaling message contains optional nonstandard data. The H.323 stack limits the overall size of messages to be max_parm_data_size + 512 bytes, and any messages that exceed this limit are truncated without any notification to the application.

For all supported H.225 message types except Setup, the application presets the TSM contents to send by passing the configured GC_PARM_BLK in a call to the **gc_SetUserInfo( )** function. When the application subsequently calls one of the Global Call functions listed in Table 1, “H.225 Messages and Global Call Functions for Sending Tunneled Signaling Messages”, on page 30, the library and stack use the preset data to construct and send a tunneled signaling message in the corresponding H.225 message. The duration parameter in the **gc_SetUserInfo( )** function must always be GC_SINGLECALL; TSM content cannot persist for more than one H.225 message.

The **gc_SetUserInfo( )** mechanism cannot be used to preset a tunneled signaling message to be sent in a Setup message because the function call requires a valid CRN, which does not yet exist at that point in the call setup process. When sending a TSM in Setup, the application must include the configured GC_PARM_BLK in the GC_MAKECALL_BLK data structure that is passes to the **gc_MakeCall( )** function.

Table 1, “H.225 Messages and Global Call Functions for Sending Tunneled Signaling Messages”, on page 30 lists the H.225 message types that can be used to send tunneled signaling messages along with the corresponding Global Call mechanism that is used set the TSM information and the Global Call function that is used to send each message type.

When reception of tunneled signaling messages is enabled as described in the “Enabling Tunneled Signaling Messages” section, applications must specifically request the message by calling the **gc_Extension( )** function and providing a tag value that identifies the specific type of H.225 message that was received. When the corresponding H.225 message contains a tunneled signaling message, the library generates an asynchronous GCEV_EXTENSIONCMPLT completion event which includes the tunneled signaling message information in the metaevent data. Tunneled signaling messages can only be retrieved within a call (the application must use a valid CRN when registering to receive tunneled signaling messages), but the call can be in any state.

Enabling Tunneled Signaling Messages

The ability to send tunneled signaling messages in outgoing H.225 messages and to retrieve TSM content from inbound H.225 messages is an optional feature that is enabled or disabled on a virtual board basis at the time the **gc_Start( )** function is called. Once the system is started with the TSM feature enabled (or disabled) for a given virtual board, the only way to disable (or enable) the feature is to stop the system, then reconfigure and restart it.

The **INIT_IPCCLIB_START_DATA( )** and **INIT_IP_VIRTBOARD( )** functions, populate the IPCCLIB_START_DATA and IP_VIRTBOARD structures, respectively, with default values. The default value of the h323_msginfo_mask field in the IP_VIRTBOARD structure does not enable either access to either Q.931 message information elements or to tunneled signaling messages. To enable either or both of these features for an IPT device, it is necessary to override the default value of the h323_msginfo_mask field with a value that represents the appropriate logical combination of the two defined mask values. To enable access to tunneled signaling messages in general, the value IP_H323_ANNEXMMSG_ENABLE must be set. The following code snippet enables Q.931 message IE access on two virtual boards and enables tunneled signaling messages on the second board only:

```
INIT_IPCCLIB_START_DATA(&ipcclibstart, 2, ip_virtboard);
INIT_IP_VIRTBOARD(&ip_virtboard[0]);
INIT_IP_VIRTBOARD(&ip_virtboard[1]);
ip_virtboard[0].h323_msginfo_mask = IP_H323_MSGINFO_ENABLE;
                                    /* override Q.931 message default */
ip_virtboard[1].h323_msginfo_mask = IP_H323_MSGINFO_ENABLE |  IP_H323_ANNEXMMSG_ENABLE;
                                    /* override Q.931 message and TSM defaults */
```

Note: Once the tunneled signaling message feature is enabled on a virtual board, there is no way to disable the reception of these messages other than stopping, reconfiguring, and restarting the virtual board.

Composing Tunneled Signaling Messages

The process of sending a tunneled signaling message begins by composing a GC_PARM_BLK that contains Global Call parameter elements for the message protocol, the message content, and any nonstandard data.

The first parameter element identifies the message protocol. It must be **one** of the following two forms:

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_OBJECTID
- value = protocol object ID information in an IP_TUNNELPROTOCOL_OBJECTID data structure and conforming to the appropriate ASN.1 format

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_ALTERNATEID
- value = alternate protocol ID information in an IP_TUNNELPROTOCOL_ALTID data structure (see the *Global Call IP Technology Guide* for details on this structure)

The second parameter element contains the actual message content. Applications should use the **gc_util_insert_parm_ref_ex( )** function to insert this parameter element because the parameter data may exceed 255 bytes.

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_CONTENT
- value = actual message content
  max. length = max_parm_data_size (configured at library start-up)

Note: In practice, applications may not be able to utilize the full maximum length of the TSM content parameter element as configured in max_parm_data_size, particularly if the TSM also contains non-standard data. The H.323 stack limits the overall size of messages to be max_parm_data_size + 512 bytes, and any messages that exceed this limit are truncated without any notification to the application.

If the tunneled signal message includes optional nonstandard data, the GC_PARM_BLK needs to contain two additional parameter elements. These parameters should only be inserted in the GC_PARM_BLK if nonstandard data is being sent in the message. The first parameter element for nonstandard data is:

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_NSDATA_DATA
- value = actual nonstandard data, max. length = max_parm_data_size (configured at library start-up)

Applications should use the **gc_util_insert_parm_ref_ex( )** function to insert this parameter element because the parameter data may exceed 255 bytes.

Note: In practice, applications may not be able to utilize full maximum parameter length configured in max_parm_data_size for nonstandard data content. The H.323 stack limits the overall size of messages to be max_parm_data_size + 512 bytes, which must contain the tunneled signaling message content as well as the nonstandard data.

The second parameter element for nonstandard data uses **one** of the following two forms:

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_NSDATA_OBJID
- value = nonstandard data object ID string conforming to appropriate ASN.1 format

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_NSDATA_H221NS
- value = H.221 nonstandard data protocol information in an IP_H221NONSTANDARD data structure

The following code example illustrates the process of composing the parameter block for a tunneled signaling message.

```
GC_PARM_BLKP               gcParmBlk = NULL;
IP_TUNNELPROTOCOL_ALTID    tsmTpAltId;
IP_TUNNELPROTOCOL_OBJECTID tsmTpObjId;

char *pMsgContent       = "00 11 22 33 44 55 44 33 33 66 66 55 77 22 11 11";
int asize = strlen(pMsgContent);
char *pTP_Oid           = "0 0 17 931";
                          // Note that the Object Id string must be in the correct ASN.1 format.
char TP_AltID_Type[]    = "Tunneled Protocol Alternate ID protocol type";
char TP_AltID_Variant[] = "Tunneled Protocol Alternate ID protocol variant";
```

```
char TP_AltID_SubId[]   = "Tunneled Protocol Alternate ID subidentifier - User";
char *ptsmNSData_Data   = "Tunneled Signaling Message Non Standard Data";
char *pTP_ObjID_Oid     = "0 0 17 931";
                        // Note that the Object Id string must be in the correct ASN.1 format.
char TP_ObjID_SubId[]   = "Tunneled Protocol Object ID subidentifier - User";
int bsize = strlen(TP_ObjID_SubId);

IP_H221NONSTANDARD tsmH221NS;
tsmH221NS.country_code = 91;
tsmH221NS.extension = 202;
tsmH221NS.manufacturer_code = 11;

INIT_IP_TUNNELPROTOCOL_ALTID(&tsmTpAltId);
strcpy(tsmTpAltId.protocolType, TP_AltID_Type);
tsmTpAltId.protocolTypeLength = strlen(TP_AltID_Type) + 1;
strcpy(tsmTpAltId.protocolVariant, TP_AltID_Variant);
tsmTpAltId.protocolVariantLength = strlen(TP_AltID_Variant) + 1;
strcpy(tsmTpAltId.subIdentifier, TP_AltID_SubId);
tsmTpAltId.subIdentifierLength = strlen(TP_AltID_SubId) + 1;

INIT_IP_TUNNELPROTOCOL_OBJECTID(&tsmTpObjId);
strcpy(tsmTpObjId.TunneledProtocol_Oid, pTP_ObjID_Oid);
tsmTpObjId.TunneledProtocol_OidLength = strlen(pTP_ObjID_Oid) + 1;
strcpy(tsmTpObjId.subIdentifier, TP_ObjID_SubId);
tsmTpObjId.subIdentifierLength = strlen(TP_ObjID_SubId) + 1;

choiceOfTSMProtocol = 1;
       /* App decides whether to use the tunneled signaling message Protocol Object ID/ AltID */
choiceOfNSData = 1;
       /* App decides which type of object identifier to use for TSM NS Data */

if (choiceOfTSMProtocol)
   /* App decides the choice of the tunneled signaling msg protocol object identifier */
   /* It cannot set both objid & alternate id */
{
   gc_util_insert_parm_ref(&gcParmBlk,
                           IPSET_TUNNELEDSIGNALMSG,
                           IPPARM_TUNNELEDSIGNALMSG_ALTERNATEID,
                           (unsigned char)sizeof(IP_TUNNELPROTOCOL_ALTID),
                           &tsmTpAltId);
}
else
{
   gc_util_insert_parm_ref(&gcParmBlk,
                           IPSET_TUNNELEDSIGNALMSG,
                           IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_OBJECTID,
                           (unsigned char)sizeof(IP_TUNNELPROTOCOL_OBJECTID),
                           &tsmTpObjId);

/* Note the use of the extended gc_util function because TSM data may exceed 255 bytes */
gc_util_insert_parm_ref_ex(&gcParmBlk,
                           IPSET_TUNNELEDSIGNALMSG,
                           IPPARM_TUNNELEDSIGNALMSG_CONTENT,
                           (unsigned char)(strlen(pMsgContent)+1),
                           pMsgContent);

/* Now fill in the Tunneled Signaling message Non Standard data fields, if used */
/* Note the use of the extended gc_util function because NSD data may exceed 255 bytes */
gc_util_insert_parm_ref_ex(&gcParmBlk,
                           IPSET_TUNNELEDSIGNALMSG,
                           IPPARM_TUNNELEDSIGNALMSG_NSDATA_DATA,
                           (unsigned char)(strlen(ptsmNSData_Data)+1),
                           ptsmNSData_Data);
```

```
if (choiceOfNSData)
   /* App decides the CHOICE of Non Standard OBJECTIDENTIFIER. */
   /* It cannot set both objid & H221 */
{
   // Set the NS Object ID
   gc_util_insert_parm_ref(&gcParmBlk,
                           IPSET_TUNNELEDSIGNALMSG,
                           IPPARM_TUNNELEDSIGNALMSG_NSDATA_OBJID,
                           (unsigned char) (strlen(ptsmNSData_Oid)+1),
                           ptsmNSData_Oid
}
else
{
   // Set the H221
   gc_util_insert_parm_ref(&gcParmBlk,
                           IPSET_TUNNELEDSIGNALMSG,
                           IPPARM_TUNNELEDSIGNALMSG_NSDATA_H221NS,
                           (unsigned char)sizeof(IP_H221NONSTANDARD),
                           &tsmH221NS);
```

Sending Tunneled Signaling Messages

Once the GC_PARM_BLK containing the TSM information has been composed by the application, the application must pass the parameter block to the call control library to be transformed into a tunneled message that can be inserted into an H.225 message. The mechanism used to hand the TSM information to the library varies depending on what Global Call function and corresponding H.225 message will be used to send the TSM.

Table 1 lists the H.225 message types that can be used to send tunneled signaling messages along with the corresponding Global Call mechanism that is used set the TSM information and the Global Call function that is used to send each message type.

Table 1. H.225 Messages and Global Call Functions for Sending Tunneled Signaling Messages

H.225 message to be used to send TSM	Mechanism used to set TSM to send	Global Call Function used to send H.225 message containing TSM
Setup	GC_MAKECALL_BLK	**gc_MakeCall( )**
Proceeding	**gc_SetUserInfo( )** (GC_SINGLECALL)	**gc_CallAck( )**
Alerting	**gc_SetUserInfo( )** (GC_SINGLECALL)	**gc_AcceptCall( )**
Connected	**gc_SetUserInfo( )** (GC_SINGLECALL)	**gc_AnswerCall( )**
Release Complete	**gc_SetUserInfo( )** (GC_SINGLECALL)	**gc_DropCall( )**
Facility	**gc_SetUserInfo( )** (GC_SINGLECALL)	**gc_Extension( )** (IPEXTID_SENDMSG, IPSET_MSG_Q931, IPPARM_MSGTYPE, IP_MSGTYPE_Q931_FACILITY)

Sending a TSM in a Setup Message

Once the GC_PARM_BLK is composed, the block is included in a GC_MAKECALL_BLK structure via the intermediate GCLIB_MAKECALL_BLK structure, and that block is then passed as a parameter in a call to **gc_MakeCall( )**. The Setup message that is sent as a result of the call to **gc_MakeCall( )** will contain a TSM with elements as specified in the GC_PARM_BLK.

The **gc_SetUserInfo( )** function *cannot* be used to preset TSM information to be sent in a Setup message because that function requires a valid CRN when setting a tunneled signaling message and the CRN does not exist at this point in the call setup. The TSM can only be specified in the GC_MAKECALL_BLK for a Setup message.

Sending a TSM in an Alerting, Connected, Facility, Proceeding, or ReleaseComplete Message

To include a tunneled signaling message in any H.225 message other than a Setup message, the application uses the **gc_SetUserInfo( )** to preset the message data before calling the Global Call function that causes the H.225 message to be sent. Data set via **gc_SetUserInfo( )** applies to the next outgoing message, so applications should be careful to call this function immediately before the function that will send the intended H.225 message.

When calling **gc_SetUserInfo( )**, the parameters should be set as follows:

- **target_type** must be set to GCTGT_GCLIB_CRN
- **target_id** is the CRN
- **infoparmblkp** is a pointer to the GC_PARM_BLK that was configured with the parameter elements for the tunneled signaling message
- **duration** must be set to GC_SINGLECALL

Four of the supported H.225 message types are sent as a direct result of a specific Global Call function call for the CRN and require no other preparation after the **gc_SetUserInfo( )**:

- **gc_AcceptCall( )** sends Alerting
- **gc_AnswerCall( )** sends Connected
- **gc_CallAck( )** sends Proceeding
- **gc_DropCall( )** sends Release Complete

But because there is no call state change associated with the Facility message, there is no dedicated Global Call function to send this message (nor is there a dedicated Global Call event to receive a Facility message). Instead, Global Call uses the generic **gc_Extension( )** mechanism to send and receive Facility messages. Because of this, an application must construct a GC_PARM_BLK to pass to the **gc_Extension( )** function call to specify that it wishes to send a Facility message; note that this GC_PARM_BLK is completely separate from the structure that sets up the TSM itself via the **gc_SetUserInfo( )** function. The GC_PARM_BLK passed to **gc_Extension( )** must contain a parameter element of the following type:

IPSET_MSG_Q931
    IPPARM_MSGTYPE
    - value = IP_MSGTYPE_Q931_FACILITY

The parameters for the **gc_Extension( )** function call should be set as follows:

- **target_type** must be set to GCTGT_GCLIB_CRN
- **target_id** is the CRN
- **ext_id** must be set to IPEXTID_SENDMSG
- **parmblkp** is a pointer to the GC_PARM_BLK that was configured with the parameter element for the Q.931 Facility message
- **retblkp** is NULL
- **mode** must be set to EV_ASYNC

Receiving Tunneled Signaling Messages

Assuming that the TSM feature was enabled when the virtual board was started, an application can request the tunneled signaling message content whenever it receives a Global Call event that corresponds to one of the supported H.225 message types. For all supported message types except Facility, the application uses the **gc_Extension( )** function and the extension ID IPEXTID_GETINFO to request the TSM, and the TSM contents are transmitted in the external data associated with the asynchronous GCEV_EXTENSIONCMPLT completion event for the function call. In the case of the Facility message, Global Call notifies the application that it has received the message via an unsolicited GCEV_EXTENSION event, and this event itself conveys the TSM in its external data.

Table 2 relates the message types of the supported H.225 messages that can contain TSM fields to the Global Call event types that are used to notify the application of the message's arrival and the tag that is used by the application when retrieving the TSM content.

Table 2. H.225 Messages and Global Call Events for Receiving Tunneled Signaling Messages

H.225 message	Global Call event used to notify application	Tag used to retrieve message fields via GCEV_EXTENSIONCMPLT
Setup	GCEV_OFFERED	TSM_CONTENT_OFFERED
Proceeding	GCEV_PROCEEDING †	TSM_CONTENT_PROCEEDING
Alerting	GCEV_ALERTING	TSM_CONTENT_ALERTING
Connected	GCEV_CONNECTED	TSM_CONTENT_CONNECTED
Release Complete	GCEV_DISCONNECTED	TSM_CONTENT_DISCONNECTED
Facility	GCEV_EXTENSION	TSM_CONTENT_EXTENSION
† The GCEV_PROCEEDING event is maskable. When Tunneled Signalling Messages are enabled, the application must ensure that this event is not masked.		

Retrieving TSMs from Alerting, Connected, Proceeding, ReleaseComplete, and Setup Messages

To retrieve TSM after receiving a Global Call state change event corresponding to an Alerting, Connected, Proceeding, ReleaseComplete, or Setup message, the application first constructs a GC_PARM_BLK that specifies the type of information it wishes to retrieve, then calls the **gc_Extension( )** function to request the information.

The GC_PARM_BLK must contain the following two parameter elements:

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TUNNELEDSIGNALMSG_CONTENT
    - value is unused; set to 1

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TSM_CONTENT_EVENT
    - value = appropriate TSM_CONTENT_... tag value as listed in Table 2

When the application calls **gc_Extension( )**, the parameters should be set up as follows:

- **target_type** must be GCTGT_GCLIB_CRN
- **target_id** must be a valid CRN
- **ext_id** must be IPEXTID_GETINFO
- **parmblkp** must point to the GC_PARM_BLK that was configured to contain the required parameter element as just described.
- **retblkp** must be a valid pointer to a GC_PARM_BLK
- **mode** must be EV_ASYNC

If the received H.225 message contained a tunneled signaling message, the library generates an asynchronous GCEV_EXTENSIONCMPLT completion event. The extevtdatap field in the METAEVENT structure for this event is a pointer to an EXTENSIONEVTBLK structure, which in turn contains a GC_PARM_BLK that contains the fields of the received tunneled signaling message. Applications are then able to extract the data of interest using the **gc_util_..._ex( )** functions.

The GC_PARM_BLK will always contain the following three parameter elements:

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TSM_CONTENT_EVENT
    - value = the appropriate TSM_CONTENT_... tag value

one or the other of the the following two elements:

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_OBJECTID
    - value = IP_TUNNELPROTOCOL_OBJECTID data structure

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TUNNELEDSIGNALMSG_ALTERNATEID
    - value = IP_TUNNELPROTOCOL_ALTID data structure

and the following element:

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TUNNELEDSIGNALMSG_CONTENT
    - value = tunneled signaling message content string

If the TSM includes optional nonstandard data, there will be two additional parameter elements:

IPSET_TUNNELEDSIGNALMSG
    IPPARM_TUNNELEDSIGNALMSG_NSDATA_DATA
    - value = nonstandard data string

and one of the following two elements:

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_NSDATA_OBJID
- value = nonstandard data object ID string in ASN.1 format

IPSET_TUNNELEDSIGNALMSG
IPPARM_TUNNELEDSIGNALMSG_NSDATA_H221NS
- value = IP_H221NONSTANDARD data structure

Notes: ***1.*** The application must take care to retrieve the Annex M Message information from any incoming H.225 message before the next H.225 message arrives. If the new message also contains TSM information, that new TSM overwrites the prior information.

2. The overall message size that the Global Call H.323 stack can handle is defined as max_parm_data_size (which is configured at library startup) + 512 bytes. Any message that is received which exceeds this length is truncated.

3. Parameter values that are contained in a GC_PARM_BLK are subject to maximum length limits that are defined for each parameter type. Any data received in a TSM that exceeds these defined limits is truncated without notification to the application.

4. The application should use the extended **gc_util_..._ex( )** functions when extracting parameters from a GC_PARM_BLK that contains TSM contents because some of the Global Call parameters for TSMs support data length that may exceed 255 bytes.

TSM Retrieval Code Example

The following code example shows how an application might handle the process of requesting tunneled signaling message after it has received a Global Call event associated with one of the supported H.225 message types.

```
GC_PARM_BLKP gcParmBlk = NULL;
GC_PARM_BLKP retParmBlk;
GC_PARM_DATA_EXT parm_data_ext;
INIT_GC_PARM_DATA_EXT(&parm_data_ext);
int frc;

switch(event)
{
   case GCEV_ALERTING:
      frc = gc_util_insert_parm_val(&gcParmBlk,
                                    IPSET_TUNNELEDSIGNALMSG,
                                    IPPARM_TUNNELEDSIGNALMSG_CONTENT,
                                    sizeof(int),
                                    1);
      frc = gc_util_insert_parm_val(&gcParmBlk,
                                    IPSET_TUNNELEDSIGNALMSG,
                                    IPPARM_TSM_CONTENT_EVENT,
                                    sizeof(int),
                                    TSM_CONTENT_ALERTING);
      frc = gc_Extension(GCTGT_GCLIB_CRN,
                         crn,
                         IPEXTID_GETINFO,
                         gcParmBlk,
                         &retParmBlk,
                         EV_ASYNC);
      break;
```

```
   case GCEV_CONNECTED:
      frc = gc_util_insert_parm_val(&gcParmBlk,
                                    IPSET_TUNNELEDSIGNALMSG,
                                    IPPARM_TUNNELEDSIGNALMSG_CONTENT,
                                    sizeof(int),
                                    1);
      frc = gc_util_insert_parm_val(&gcParmBlk,
                                    IPSET_TUNNELEDSIGNALMSG,
                                    IPPARM_TSM_CONTENT_EVENT,
                                    sizeof(int),
                                    TSM_CONTENT_CONNECTED);
      frc = gc_Extension(GCTGT_GCLIB_CRN,
                         crn,
                         IPEXTID_GETINFO,
                         gcParmBlk,
                         &retParmBlk,
                         EV_ASYNC);
      break;

   ...
//Similar cases for other event types of interest
   ...

   case GCEV_EXTNCMPLT:
      GC_PARM_DATA *parmp = NULL;
      while (GC_SUCCESS == (gc_util_next_parm_ex(parm_blk, &parm_data_ext)))
      {
         switch (parmp->set_ID)
         {
            case IPSET_TUNNELEDSIGNALMSG:
            switch (parm_data_ext.parm_ID)
            {
               case IPPARM_TSM_CONTENT_EVENT:
                  printf("\tReceived TSM in message type: %s\n",
                          parm_data_ext.value_buf);
                  break;

               case IPPARM_TUNNELEDSIGNALMSG_CONTENT:
                  printf("\tReceived extension data (TSM) Msg Content: %s\n",
                          parm_data_ext.value_buf);
                  break;

               case IPPARM_TUNNELEDSIGNALMSG_PROTOCOLOBJID:
                  printf("\tReceived extension data (TSM) PROTOCOL_OBJID:
                         %s\n", parm_data_ext.value_buf);
                  break;

               case IPPARM_TUNNELEDSIGNALMSG_ALTERNATEID:
                  printf("\tReceived extension data (TSM) TUNNELPROTOCOL_ALTID:
                         %s\n", parm_data_ext.value_buf);
                  break;

               // Additional cases for optional NSD
               ...

            }

      }

}
```

Reference Information for TSM Parameters

Table 3 represents updated entries for the Summary of Parameter Sets and Parameter Usage table covering the parameters used for tunneled signaling messages.

Table 3. Summary of TSM Parameter Usage

Set ID	Parameter ID	Set	Send	Retrieve	SIP/ H.323
IPSET_ TUNNELED SIGNALMSG	IPPARM TSM_CONTENT_ EVENT	GC_PARM_BLK	**gc_Extension( )** IPEXTID_GETINFO	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_ALTERNATEID	GC_MAKECALL_BLK for **gc_MakeCall( )** **gc_SetUserInfo( )** for other functions	**gc_MakeCall( )**, **gc_CallAck( )**, **gc_AcceptCall( )**, **gc_AnswerCall( )**, **gc_DropCall( )**, **gc_Extension( )** IPEXTID_SEND MSG for Q.931 Facility message	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_CONTENT	GC_MAKECALL_BLK for **gc_MakeCall( )** **gc_SetUserInfo( )** for other functions	**gc_MakeCall( )**, **gc_CallAck( )**, **gc_AcceptCall( )**, **gc_AnswerCall( )**, **gc_DropCall( )**, **gc_Extension( )** IPEXTID_SEND MSG for Q.931 Facility message	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_NSDATA_DAT A	GC_MAKECALL_BLK for **gc_MakeCall( )** **gc_SetUserInfo( )** for other functions	**gc_MakeCall( )**, **gc_CallAck( )**, **gc_AcceptCall( )**, **gc_AnswerCall( )**, **gc_DropCall( )**, **gc_Extension( )** IPEXTID_SEND MSG for Q.931 Facility message	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_NSDATA_ H221NS	GC_MAKECALL_BLK for **gc_MakeCall( )** **gc_SetUserInfo( )** for other functions	**gc_MakeCall( )**, **gc_CallAck( )**, **gc_AcceptCall( )**, **gc_AnswerCall( )**, **gc_DropCall( )**, **gc_Extension( )** IPEXTID_SEND MSG for Q.931 Facility message	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only

Table 3. Summary of TSM Parameter Usage (Continued)

Set ID	Parameter ID	Set	Send	Retrieve	SIP/ H.323
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_NSDATA_ OBJID	GC_MAKECALL_BLK for **gc_MakeCall( )** **gc_SetUserInfo( )** for other functions	**gc_MakeCall( )**, **gc_CallAck( )**, **gc_AcceptCall( )**, **gc_AnswerCall( )**, **gc_DropCall( )**, **gc_Extension( )** IPEXTID_SEND MSG for Q.931 Facility message	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_PROTOCOL_ OBJECTID	GC_MAKECALL_BLK for **gc_MakeCall( )** **gc_SetUserInfo( )** for other functions	**gc_MakeCall( )**, **gc_CallAck( )**, **gc_AcceptCall( )**, **gc_AnswerCall( )**, **gc_DropCall( )**, **gc_Extension( )** IPEXTID_SEND MSG for Q.931 Facility message	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG) GCEV_EXTENSION (IPEXTID_RECEIVE MSG) for Q.931 Facility message only	H.323 only
IPSET_ TUNNELED SIGNALMSG	IPPARM TUNNELEDSIGNAL MSG_PROTOCOL_ OBJID (deprecated)	GC_PARM_BLK	**gc_MakeCall( )**	GCEV_ EXTENSIONCMPLT (IPEXTID_RECEIVE MSG)	H.323 only

Table 4 shows the parameter IDs in the IPSET_TUNNELEDSIGNALMSG parameter set, which is used when sending or receiving tunneled signaling messages (TSMs) in the H.323 protocol.

Table 4. IPSET_TUNNELEDSIGNALMSG Parameter Set

Parameter IDs	Data Type & Size	Description	SIP/ H.323
IPPARM_TSM_ CONTENT_EVENT	Type: enum	Used to identify the type of Global Call event to retrieve TSM content from. Values include: • TSM_CONTENT_OFFERED • TSM_CONTENT_PROCEEDING • TSM_CONTENT_ALERTING • TSM_CONTENT_CONNECTED • TSM_CONTENT_DISCONNECTED • TSM_CONTENT_EXTENSION	H.323 only
IPPARM_TUNNELEDSIGNALMSG_ ALTERNATEID	Type: IP_ TUNNELPROTOCOL_ ALTID Size: sizeof(IP_ TUNNELPROTOCOL_ ALTID)	Used to contain a tunneled protocol alternate identifier in a tunneled signaling message (TSM). Either this or the tunneled protocol object ID must exist in a TSM. If the application is using a tunneled protocol object ID when sending a TSM, this parameter should not be inserted in the GC_PARM_BLK.	H.323 only
† For parameters with data of type String, the length in a GC_PARM_BLK is the length of the data string plus 1. ‡ The full maximum length that is configured may not be usable in practice because the H.323 stack limits total message size to max_parm_data_size + 512 bytes. Longer messages are truncated without notification to the application.			

Table 4. IPSET_TUNNELEDSIGNALMSG Parameter Set (Continued)

Parameter IDs	Data Type & Size	Description	SIP/ H.323
IPPARM_TUNNELEDSIGNALMSG_CONTENT	Type: string † Size: max. length= max_parm_data_ size ‡ (configured via IPCCLIB_START_DATA)	Used to contain any data content of a tunneled signaling message (TSM), which is a sequence of octet strings.	H.323 only
IPPARM_TUNNELEDSIGNALMSG_NSDATA_DATA	Type: string † Size: max. length= max_parm_data_ size ‡ (configured via IPCCLIB_START_DATA)	Used to contain any non-standard data in a tunneled signaling message (TSM). If no non-standard data is being sent in a TSM, this parameter should not be inserted in the GC_PARM_BLK.	H.323 only
IPPARM_TUNNELEDSIGNALMSG_NSDATA_H221NS	Type: IP_H221 NONSTANDARD Size: sizeof(IP_ H221NONSTANDARD)	Used to contain an H.221 non-standard data identifier in a tunneled signaling message (TSM). When sending non-standard data in a TSM, ether this ID or the non-standard data object ID must exist in the non-standard data. If non-standard data is not being sent, or if a non-standard data object ID is being used when sending a TSM, this parameter should not be inserted in the GC_PARM_BLK.	H.323 only
IPPARM_TUNNELEDSIGNALMSG_NSDATA_OBJID	Type: string † Size: max length = MAX_NS_PARAM_ OBJID_LENGTH (40)	Used to contain a non-standard data object identifier in a tunneled signaling message (TSM). When sending non-standard data in a TSM, either this ID or an H.221 non-standard data ID must exist in the non-standard data. If non-standard data is not being sent, or if an H.221 non-standard data ID is being used when sending a TSM, this parameter should not be inserted in the GC_PARM_BLK.	H.323 only
IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_OBJECTID	Type: IP_ TUNNELPROTOCOL_ OBJECTID Size: sizeof(IP_ TUNNELPROTOCOL_ OBJECTID)	Used to contain a tunneled protocol object identifier in a tunneled signaling message (TSM). Either this or the tunneled protocol alternate ID must exist in a TSM. If the application is using an alternate identifier when sending a TSM, this parameter should not be inserted in the GC_PARM_BLK.	H.323 only
IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_OBJID	Type: string † Size: max length = MAX_TSM_POID_ PARAM_LENGTH (128)	Deprecated parameter previously used to contain a tunneled protocol object identifier in a tunneled signaling message. Superseded by IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_ OBJECTID.	H.323 only

† For parameters with data of type String, the length in a GC_PARM_BLK is the length of the data string plus 1.
‡ The full maximum length that is configured may not be usable in practice because the H.323 stack limits total message size to max_parm_data_size + 512 bytes. Longer messages are truncated without notification to the application.

IP_TUNNELPROTOCOL_OBJECTID

```
typedef struct
{
    unsigned long  version;
    char           TunneledProtocol_Oid[MAX_TSM_OBJID_VARS_LENGTH];
    int            TunneledProtocol_OidLength;
    char           subIdentifier[MAX_TSM_OBJID_VARS_LENGTH];
    int            subIdentifierLength;
} IP_TUNNELPROTOCOL_OBJECTID;
```

Description

The IP_TUNNELPROTOCOL_OBJECTID data structure is used in H.323 Annex M tunneled signaling to identify the tunneling protocol using a protocol object ID. This data structure is used as the value of a Global Call parameter element of type IPSET_TUNNELEDSIGNALMSG / IPPARM_TUNNELEDSIGNALMSG_PROTOCOL_OBJECTID. This data structure is not used when the tunneled signaling message uses the alternate ID method to identify the protocol.

Applications should use the **INIT_IP_TUNNELPROTOCOL_OBJECTID( )** function to initialize the structure with the correct version number and initial field values.

Field Descriptions

The fields of the IP_TUNNELPROTOCOL_OBJECTID data structure are described as follows:

version
: the version number of the data structure; the correct value is set by the **INIT_IP_TUNNELPROTOCOL_OBJECTID( )** initialization function and should not be overridden by applications

TunneledProtocol_Oid
: a string that identifies the tunneled protocol object
maximum length: MAX_TSM_OBJECTID_VARS_LENGTH

TunneledProtocol_OidLength
: the actual length of the TunneledProtocol_Oid string

subIdentifier
: a string that provides additional tunneled protocol identification
maximum length: MAX_TSM_OBJECTID_VARS_LENGTH

subIdentifierLength
: the actual length of the subIdentifier string

1.8 Preserving Data in User Configuration Files

Configuration settings unique to your environment can be preserved and re-applied whenever a Dialogic® HMP license is changed or reactivated.

Previously, customized settings in the *.config* file were lost every time a new Service Update was installed or a new Dialogic® HMP license was activated, and had to be re-entered.

A new file called the user configuration file, *Hmp.Uconfig*, is introduced. This file, which has the same format as the *.config* file, contains the parameters and parameter values that differ from the default that are used in your environment.

Rather than editing the generated *.config* file, you create a separate *Hmp.Uconfig* file in the *data* directory under DIALOGIC_DIR, the environment variable for the directory in which the HMP software is installed. The contents of the *Hmp.Uconfig* file are not overwritten but are merged into the generated configuration file whenever a new Service Update is installed or a new HMP license is activated. You should also make a copy of the *Hmp.Uconfig* file and save it in a safe location as a backup.

Example

The following is a sample *Hmp.Uconfig* file, where the default AGC setting has been changed and a new parameter has been added:

```
[encoder]
SetParm=0x400,0   !AGC Enabled (1=Enable, 0=Disable)

[0xe]
SetParm=0xb17,4   !QFC3_ PrmResponseTimeout default 3 seconds
```

The following is a sample merged *HMP.config* file (the generated file). It shows the new AGC setting in the [encoder] section followed by the default value for AGC, introduced by "!^". It also shows a new parameter in [0xe], introduced by the "!<add>".

```
[encoder]
SetParm=0x400,0   !AGC Enabled (1=Enable, 0=Disable)
!^SetParm=0x400,1 !AGC Enabled (1=Enable, 0=Disable)

[0xe]
...
!<add>
SetParm=0xb17,4   !QFC3_ PrmResponseTimeout default 3 seconds
!</add>
```

1.9 Audio Speed Control

Audio speed control, or speed adjustment, which is provided through the voice API, is supported on HMP software.

The documentation for this feature exists in the *Dialogic® Voice API Library Reference* and the *Dialogic® Voice API Programming Guide*. All statements in these documents saying that speed adjustment is not supported on HMP software can be removed.

1.10 Global Call 1PCC Configuration to Reject Incoming Video Calls

The service update provides support for the Dialogic® Global Call API library configuration to reject incoming video calls in 1PCC mode (IPPARM_1PCC_REJECT_VIDEO).

Video is not supported when the Global Call call control library is operating in 1PCC mode, so any SDP offer that includes a valid video descriptor cannot be fully accepted by the library. The library's default behavior is to reject the video offer by setting the port number to 0 in the video media descriptor of the SDP answer, but to allow the audio offer to be accepted via an SDP answer in a 200OK response.

Optionally, the library can be configured to always reject SDP offers that contain a valid video descriptor. In this optional mode, if the SDP offer containing a video descriptor is contained in an INVITE or re-INVITE, the library responds with a 488NotAcceptableHere. If the SDP offer containing a video descriptor is contained in a 200OK response to an INVITE or re-INVITE sent by Global Call, the library ACKs the 200OK but then terminates the call with BYE.

The optional rejection behavior mode is enabled on a virtual board basis by calling **gc_SetConfigData( )** with a GC_PARM_BLK that contains an IPSET_CONFIG / IPPARM_1PCC_REJECT_VIDEO parameter element. Once enabled, the optional rejection behavior remains in effect until the system is restarted; there is no way to revert to the default rejection behavior without stopping and restarting.

1.11 IP Multicast Receive

The service update provides Dialogic® IP Media Library support for the multicast receive feature using RTP. (Multicast receive using RTCP is not supported.) For details, see Section 3.3.4, “Dialogic® IP Media Library API Library Reference”, on page 66.

1.12 400 Channels of IVR (G.711)

The service update provides support for 400 channels of IVR (G.711), which was tested in the following configuration:

- Dual Intel Xeon 3.6 GHz processors with Hyper-Threading (HT) Technology
- 1 GB RAM
- Using Red Hat Enterprise Linux Advanced Server 4.0 with Update 2
- 20 ms frame size

Table 5. New IVR (G.711) Resource Configuration Tested

Configuration	RTP	Enhanced RTP	Voice	Conferencing (DCB)	Fax	Speech	Multimedia
IVR (G.711)[1]	400	0	400	0	0	0	0

1.This configuration supports 20 and 30 ms frame sizes. (10 ms frame is not supported.)

1.13 Updates to Multimedia File Conversion Tools

The off-line multimedia file conversion tools are used to convert multimedia file data between industry-standard formats and the proprietary format used with the Multimedia API. These tools are not provided as part of the service update, but are available on the Dialogic web site. This notice is intended to inform customers of the following:

- The location where these tools are available on the web was originally documented in the Dialogic® HMP Software Release 1.5LIN *Release Guide* and the Dialogic® *Multimedia API Programming Guide*. The location has changed to the following web site: *http://dialogic.com/manuals*
- The tools were originally documented in Chapter 5 of the *Dialogic® Multimedia API Programming Guide*. The file conversion information from this programming guide has now been moved to a separate document, the *Dialogic® Multimedia File Conversion Tools User Guide* (05-2453-001), and updates have been made to it. This user guide is now the repository for all information regarding the tools, including all tool release updates. The revision history section in the document describes the document updates and can also be used to identify any tool updates. The user guide is available from the same web site as the tools.

1.14 Intel Pentium D Processor

The service update adds support for the Intel Pentium D processor, which was tested in the following configuration:

- Intel Pentium D 3.2 GHz processor
- 4 GB RAM
- Using Red Hat Enterprise Linux Advanced Server 4.0 with Update 2

Table 6. New Video Services Configuration Tested with Intel Pentium D Processor

Configuration	RTP	Enhanced RTP	Voice	Conferencing (DCB)	Fax	Speech	Multimedia
Video Services	240	0	120	0	0	0	120

1.15 Red Hat Enterprise Linux 4.0

The service update adds support for Red Hat Enterprise Linux 4.0 (in addition to the existing support for Release 3.0). The following software and installation requirements apply to this feature.

- Red Hat Enterprise Linux Advanced Server, Enterprise Server, or Workstation (AS, ES, or WS) **Release 4.0** with Update 1 or Update 2

  Note: **No kernel/OS changes are needed to use Red Hat Enterprise Linux 4.0.**

  Note: SNMP is not supported when using the default netsnmp (version 5.1.2) that is provided with Red Hat Enterprise Linux 4.0. **Workaround:** Use the earlier 5.0.10 version of netsnmp.

  – The Update requires a Certificate of Authorization from Red Hat. Use the following instructions to obtain the update.
    - Sign in to the Red Hat Network at https://rhn.redhat.com/index.pxt. If you do not have an existing login, you can select "Create Login" to create a Red Hat login and subscribe to the Red Hat network. You will then need to enter the Red Hat product ID.
    - Select **Channels** from the tab selections at the top of the page.
    - Select **Red Hat Enterprise Linux AS** (or ES or WS) from the Channel Name list.
    - Select **Downloads** from the menu selections.
    - After reading and accepting the License Agreement, follow the instructions provided on the web page to download the ISO images that comprise the updates. Also, refer to the Red Hat Installation Manual for additional information.

  – Finally, follow the instructions in the *Dialogic*® Host Media Processing Software Release 1.5 Linux *Software Installation Guide* and install the HMP software obtained from the Dialogic® HMP 1.5LIN software download site at http://dialogic.com/support.

1.16 120 Channels of T.38 Fax

The service update provides support for 120 channels of T.38 Fax. The following new Fax resource configuration has been tested:

Table 7. New Fax Resource Configuration Tested

Configuration	RTP	Enhanced RTP	Voice	Conferencing (DCB)	Fax	Speech	Multimedia
Unified Messaging	120	120	120	0	120	0	0

1.17 Debian GNU/Linux 3.1

The service update provides support for Debian GNU/Linux 3.1 (a.k.a. sarge). The following software and installation requirements apply to this feature.

- Debian GNU/Linux 3.1 (a.k.a. sarge), which includes kernel 2.6.8-2.
  – See http://www.debian.org for information on obtaining the download.

- At a minimum, the kernel *.config* file should be configured for an Intel Pentium 4 processor (which also includes the Mobile Intel Pentium 4 Processor-M and the Intel Xeon processor) as follows:
  - CONFIG_M586 should be disabled and CONFIG_MPENTIUM4 should be enabled
  - CONFIG_PREEMPT should be enabled
  - CONFIG_HPET_RTC_IRQ should be disabled
- After making the changes, you must recompile the kernel.
- Get and install the stable lsb-base package (version 2.0-7) from the Debian web site at http://packages.debian.org/lsb-base.
- Finally, follow the instructions in the Section 1.8, "Preserving Data in User Configuration Files", on page 39 to install the HMP software obtained from the Dialogic® HMP 1.5LIN software download site at http://dialogic/support/. Refer to the Dialogic® Host Media Processing Software Release 1.5 Linux *Software Installation Guide* for more information.

1.18 SUSE Linux Enterprise Server 9

The service update provides support for SUSE Linux Enterprise Server 9. The following software and installation requirements apply to this feature.

- SUSE Linux Enterprise Server 9, which includes kernel 2.6.5.
  - At a minimum, the kernel *.config* file should be configured for an Intel Pentium 4 processor (which also includes the Mobile Intel Pentium 4 Processor-M and the Intel Xeon processor) as follows:
    - CONFIG_M586 should be disabled and CONFIG_MPENTIUM4 should be enabled
    - CONFIG_PREEMPT should be enabled
    - CONFIG_HPET_RTC_IRQ should be disabled
  - After changing the configuration, you must recompile the kernel using the following commands:
    - make
    - make modules_install
    - make install
  - Finally, follow the instructions in the Section 1.8, "Preserving Data in User Configuration Files", on page 39 to install the HMP software obtained from the Dialogic® HMP 1.5LIN software download site at http://dialogic/support/. Refer to the *Dialogic*® Host Media Processing Software Release 1.5 Linux *Software Installation Guide* for more information.

1.19 Service Update Software Installation

Note: You must meet the installation requirements described in the *Dialogic®* Host Media Processing Software Release 1.5 Linux *Software Installation Guide* and the *Release Guide*. In addition, before you install the HMP software, see the documentation updates in Section 3.1, "Release Documentation", on page 62 and Section 1.8, "Preserving Data in User Configuration Files", on page 39.

As a reminder, this includes the following requirements:

- If you have a Dialogic® HMP Software version prior to HMP 1.5 installed on your system, you must uninstall it before proceeding with the service update installation.
- Also, if you have Dialogic® system release installed on your system, you must uninstall it before proceeding with the service update installation.

If you do not have Dialogic® HMP Software Release 1.5LIN or a previous service update installed on your system, installing the current service update will provide a **full install** of the Dialogic® HMP Software, including the service update.

Note: However, before you **use** the software, you must obtain a license file containing HMP software license data. Although a verification license supplied with the software will allow you to run the verification demo to confirm that you have installed the software properly, its features are limited and you may want to obtain another type of license. You can obtain another type of license before or after you install the HMP software. For more information, refer to the Dialogic® Host Media Processing Software Release 1.5LIN *License Manager Administration Guide*.

If you already have Dialogic® HMP Software Release 1.5LIN or a previous service update installed on your system, installing the current service update will provide an **update install** of the Dialogic® HMP Software, which adds the service update content to your existing Dialogic® HMP 1.5LIN Software.

The following procedure describes how to install the service update.

1. Log in to the Linux system as root.

2. Download the Dialogic® HMP 1.5LIN service update software package from http:/dialogic.com/telecom/support/hmplinux/hmp15/ and then untar the package. The service update software contains the following install scripts:
   - The top level script installs runtime components and header files (run this script to be able to compile your application) (**Recommended**)
   - The script in the redistributable-runtime directory installs runtime components (no header files except for demo programs)
   - The script in the sdk directory installs header files for application development

3. Select the desired script and cd to the appropriate directory. Then enter the following command to start the install script:

   ```
   ./install.sh
   ```

   Depending upon the install script used, messages similar to the following are displayed:

   ```
   ------------------------------------------------------------------------
   Installing Dialogic (R) Host Media Processing Software 1.5
   Redistributable Runtime for Linux

   Initializing install, please wait .......

   ========================================================================
        Dialogic ® Host Media Processing Software 1.5
                    Redistributable Runtime
                         INSTALLATION
   ```

```
You will now have the opportunity to install software packages.
After the menu is displayed, enter the package number(s) of the desired
packages, separated by a space. Enter A for all packages, Q to quit.

Package dependencies will be automatically resolved during installation. For
example, selecting a single package will automatically install all packages
required for that selection.

Press ENTER to display the menu packages.
```

4. Press **Enter**.

   The following menu is displayed:

```
Item   Package Description
---- -------------------------------------------------------------------
  1   Dialogic(R) Host Media Processing Software (74 MB)
  2   SNMP Component Manager (48 MB)
  3    Documentation (15 MB)

  A    Install All (96 MB)
  Q    Quit Installation

Enter the packages you wish installed, separated by a space, or [A,a,Q,q]:
```

5. Enter the numbers or letter that correspond to what you want to install. Unless space or other considerations dictate that you limit what you install, you can just enter **a** to install everything.

   You will see various messages indicating installation activity, such as package installation order, checking for previously installed packages, checking for sufficient disk space, and installation progress. Then the installation menu (shown in preceding step) will reappear.

6. Enter **q** to quit and complete the installation process. The following message is displayed:

```
Do you wish to run config.sh to configure your system [Yn] ?
```

7. At this point, you have two options:
   - Enter ***y*** if you already have obtained a license or want to use the verification license supplied with the software. Then follow the procedure configuring the system and activating a license using the License Manager application as described in Section 2.4 of the *Dialogic®* Host Media Processing Software Release 1.5LIN *Software Installation Guide*.
   - Enter ***n*** if you do not want to use the verification license or have not obtained another type of license. When you do obtain a license, make sure to run *config.sh* (located in */usr/dialogic/bin*) and follow the procedure configuring the system and activating a license using the License Manager application as described in Section 2.4 of the *Dialogic®* Host Media Processing Software Release 1.5LIN *Software Installation Guide*.

   Note: You must run `config.sh` at least once after installing the software.

After the configuration has been completed, the following messages are displayed:

```
Configuration is now complete.

Before using the software, you must ensure that the Dialogic(R)
environment variables are set using the following action:

       - Logout and login
```

```
The Dialogic(R) system services will automatically
start every time the system is reboooted.

NOTE: To start and stop system services manually, use the dlstop and
dlstart scripts found in /usr/dialogic/bin.


Do you wish to run the HMP demo on your system [y/n] ?
```

8. The Dialogic® HMP Verification Demo confirms the success of the HMP Software service update installation by establishing a basic level of functionality after the installation has been completed. Enter **y** to run the HMP Verification Demo.

   The system will then start the Dialogic services. After the Dialogic services have been started, the Dialogic®HMP Verification Demo will start automatically. After the demo has completed, the Dialogic services will be stopped and a message similar to the following will be displayed:

   ```
   Installation of the Dialogic(R) Host Media Processing Software Release 1.5
   Redistributable Runtime for Linux was successful.
   ```

9. Finally, make sure to log out and log back in to your system to set the environment variables. Dialogic®HMP services will start automatically whenever you reboot your system. You can also manually start Dialogic®HMP services using dlstart.

   Note: The DIALOGIC_LIB environment variable (for example -L${DIALOGIC_LIB} -lgc) should be used for linking to Dialogic® HMP Software Release 1.5 service update libraries. Failure to use this environment variable in *makefiles* may cause linking failures due to potential name conflicts.

Release Issues

This chapter describes issues that can affect the operation of the Dialogic® Host Media Processing (HMP) Software Release for 1.5LIN, and it covers the following topics:

2.1 Issues

The release issues are listed using the following types of information:

Issue Type

This classifies the type of release issue based on its effect on users and its disposition:

- **Known** - A minor issue that affects the operation of the product. Known issues are still open and may or may not be fixed in the future. This category also includes *interoperability issues* (i.e., issues relating to combining different Dialogic® telecom products in the same system) and *compatibility issues* (i.e., issues that affect the use of Dialogic® telecom products with third-party products). This type of issue is listed in Table 2, "Known Issues in Dialogic® HMP 1.5LIN", on page 55.
- **Resolved** - An issue that is resolved in this release. It may have been resolved by fixing the operation of the product or by clarifying the documentation. This type of issue is listed in Table 1, "Issues Resolved in Dialogic® HMP 1.5LIN", on page 49.

Defect Number

A unique identification number that is used to track each issue reported via a formal Change Control System. Additional information on defects may be available via the Defect Query tool at *http://membersresource.dialogic.com/defects/* (If you select this link, you will be asked to login. If you do not have a member login, you may create one.) For issues without an associated defect number, this column contains "NA" (Not Applicable).

PTR Number

Problem Tracking Report Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the defect number is used to track the issue. (The Defect Query tool also provides the ability to search by legacy PTR number.)

SU No.

For issues that are resolved in a Release Update or Service Update (SU), this indicates the SU number. For issues that are resolved for the General Availability (GA) of the base release (before any Service Updates), this column contains "GA."

Product or Component

This identifies the product or component to which the issue relates, typically one of the following types:

- A system-level product or component; for example, Dialogic® HMP software or configuration management
- A software product; for example, the Dialogic® Global Call library
- A software component; for example, the Dialogic® Voice API firmware

Description

This provides a summary description of the issue. For unresolved issues, a work-around is included when available.

The following tables list the issues for this release:

- Table 1 lists the issues resolved for this release.
- Table 2 lists the known issues for this release.

Note: The "Resolved" issues listed in Table 1 have been fixed since the original release of Dialogic® HMP Software Release 1.5LIN.

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00042665	--	132	Dialogic® HMP software	Installing KillTask overlay affects audio quality.
IPY00042340	--	130	Dialogic® HMP software	There are missing digits when using RFC 2833 for DTMF.
IPY00042528	--	128	Configuration	The *dlgsysmonitorserver.exe* process crashes.
IPY00042035	--	128	Configuration	The *pyramid.scd* file is not generated when an attempt to view a license after activating it causes an error.
IPY00042329	--	128	Global Call IP (SIP)	When programmed in 1PCC GCCAP_DONTCARE, the SIP G.729AB ingress call sets an incorrect RX media codec.
IPY00041583	--	127	Dialogic® HMP software	Dialogic® HMP software sends the bye message, IPEC_SIPReasonStatusBYE = 5800, upon making a call after the remote Nortel CS1000 sends an additional media type.
IPY00041686	--	127	Fax	When using manual modify mode, H.323 functionality breaks with regard to transitioning from audiio to fax.
IPY00042146	--	127	Global Call	The functions **gc_AttachResource( )**, **gc_GetXmitSlot( )**, and **gc_Listen( )** have a long delay returning their respective events.

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN (Continued)

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00041815	--	127	Global Call IP (SIP)	Call control failures occur when an early media SIP reINVITE parameter is added to the media load CONFIG file.
IPY00041789	--	127	Global Call IP (SIP)	There is no checking of UPDATE in the Allow Header of the Ingress INVITE/ 200 OK.
IPY00041296	--	127	H.323	In the LD state MediaDrop, the EVENT_CHANNELS_OPENED event is handled only when the call is in the Connected state. This event should be handled in the Alerting and Proceeding states as well.
IPY00041280	--	127	H.323	Codec validation does not occur when the remote side offers a fax-only codec for an H.323 slow start call.
IPY00041876	--	127	IP Media	H.323 transfer fails with the RTF log ERR1 message, EVENT_CONNECT_SIGNALING failed.
IPY00041405	--	127	IP Media	When the **gc_Listen( )** function is executes in async mode, an IMPEV_LISTEN event is received instead of a GCEV_LISTEN event.
IPY00041836	--	126	Dialogic® HMP software	A Qerror_killtask causes missing TDX events (QERROR_KILLTASK(0) encqerrorlocs.h 80000d qkernerr.h).
IPY00042016	--	126	Voice	**ATDX_BUFDIGS( )** returns an incorrect value.
IPY00041300	--	123	Global Call IP (SIP)	SIP calls are rejected with a "486 Busy Here" message.
IPY00041118	--	123	Global Call IP (SIP)	The application is unable to make a SIP call using the **gc_MakeCall( )** function on the same channels previously used to make an H323 call.
IPY00039823	--	123	Global Call IP (SIP)	When modifying the "ReferTo" header field,the Global Call API only adds content to the end of the SIP header instead of replacing the contents of the pre-existing header field. This results in two "ReferTo" header fields sent in the outgoing SIP REFER message when invoking the call transfer.
IPY00040743	--	123	IP	There is excessive logging generated by SIP INFO messages.
IPY00041254	--	123	Voice	When two **dx_open( )** calls are made to the same channel, it results in two inits of the same CSP Channel.
IPY00038706	--	123	Voice	Frequent calls to the **dx_stopch( )** function result in no event return and unavailable channels.
IPY00041063	--	121	Conferencing (DCB)	The **dcb_unmonconf( )** function unmonitors the wrong conferee (party).
IPY00040029	--	121	Conferencing (DCB)	A memory leak occurs when a conference is established and then deleted quickly and frequently.
IPY00038217	--	121	Conferencing (DCB)	The **dcb_setcde( )** function fails with an "invalid board" error.

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN (Continued)

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00038869	--	121	Configuration	Dialogic® HMP fails to start when Non Linear Processing (NLP) is defined in the configuration file.
IPY00037679	--	121	Configuration	Service startup fails when attempting to stop and restart the Dialogic® HMP service without rebooting the operating system on an Intel Dual Xeon 5160 3.0GHz system only.
IPY00038218	--	121	Device Management	An exception occurs in the Device Management library when **dev_SetError( )** keeps data in local thread storage.
IPY00039505	--	121	Fax	A request to switch to T.38 fax is rejected, but a gcev_taskfail event is not received.
IPY00039410	--	121	Fax	Fax channels do not respond to the **fx_stopch( )** function.
IPY00037252	--	121	Fax	The Fax resource hangs indefinitely when calling the **dev_Disconnect( )** function.
IPY00039677	--	121	Global Call	The **gc_ReleaseCall( )** function does not receive a termination event after the application attempts to handle a GCEV_TASKFAIL.
IPY00039639	--	121	Global Call	The **gc_SetAlarmParm( )** function causes an assert in *msvcrt.dll*.
IPY00039451	--	121	Global Call	The **gc_AcceptModifyCall( )** function fails to return a termination event.
IPY00038848	--	121	Global Call	When **gc_DropCall( )** returns a cause code of 5801 from the application, an invalid response code appears in the RTF log.
IPY00039965	--	121	Global Call (IP)	An outbound IP call fails with "IPEC_SIPReasonStatus503ServiceUnavailable" when the host name is passed for the destination address in the dial string.
IPY00039832	--	121	Global Call IP	Compiler warnings occur due to unused strings in the header files.
IPY00038732	--	121	Global Call IP	The transmit codec transmits at 6.3 kbps despite being programmed at G.723.1 5.3 kbps.
IPY00038150	--	121	Global Call IP	A simultaneous re-INVITE from both sides of a call causes a TASKFAIL for the **gc_ReqModifyCall( )** function.
IPY00037699	--	121	Global Call IP (H.323)	The **gc_Stop( )** function fails to return control to the application when called after the LAN cable is disconnected.
IPY00040440	--	121	Global Call IP (SIP)	A fallback occurs when DNS resolution fails while making a Dialogic® HMP SIP call.
IPY00039707	--	121	Global Call IP (SIP)	A "glare" condition resulting in a disconnected call occurs during an automatic SIP reINVITE when media switches from audio to fax.
IPY00039564	--	121	Global Call IP (SIP)	A GCEV_EXTENSION event (IPSET_SWITCH_CODEC, IPPARM_READY) is not received for T.38 call.

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN (Continued)

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00039401	--	121	Global Call IP (SIP)	The "Record-Route" field of the SIP header message gets incorrectly reported as the "Route" field in an incoming SIP message when using the Global Call API.
IPY00039333	--	121	Global Call IP (SIP)	The SIP INVITE fast-start codec G.729A should default to annex B.
IPY00039325	--	121	Global Call IP (SIP)	Dialogic® HMP sends a 'blank' RTP when not connected to the CT bus.
IPY00039315	--	121	Global Call IP (SIP)	An Access Violation (0xC0000005) occurs in *LIBSIPSIGAL.DLL* when receiving a SIP REFER message containing a header field parameter with content greater than 255 bytes.
IPY00038992	--	121	Global Call IP (SIP)	The application is missing a termination event after executing the **gc_InvokeXfer( )** function followed by a disconnect from the receiving end.
IPY00038868	--	121	Global Call IP (SIP)	A GCEV_CONNECTED event is not returned from outgoing calls using SIP.
IPY00038827	--	121	Global Call IP (SIP)	The **gc_AnswerCall( )** function does not receive a response when a SIP CANCEL is received after a SIP INVITE.
IPY00038365	--	121	Global Call IP (SIP)	Egress SIP calls work briefly, then omit SDP (G711A codecs) in egress INVITE messages.
IPY00038343	--	121	Global Call IP (SIP)	When creating registrations, the customer runs into a limit and gets an error unless he unregisters a previous user.
IPY00038342	--	121	Global Call IP (SIP)	Simultaneous disconnects cause an error at RTL after a number of calls.
IPY00038060	--	121	Global Call IP (SIP)	A rvSdpStringGetData crash occurs when there are no media attributes containing "rtpmap" in a SIP INVITE.
IPY00037825	--	121	Global Call IP (SIP)	The **gc_InvokeXfer( )** function fails with the AVAYA IP PBX.
IPY00037778	--	121	Global Call IP (SIP)	The Session Description Protocol (SDP) is missing from SIP re-INVITE after calling the **gc_ReqModifyCall( )** function.
IPY00037737	--	121	Global Call IP (SIP)	The **gc_MakeCall( )** function returns a GCEV_TASKFAIL event when frames per packet are set to 20.
IPY00037613	--	121	Global Call IP (SIP)	The **gc_InvokeXfer( )** function does not return a termination event.
IPY00036779	--	121	Global Call IP (SIP)	Assertion faults are received when “DON’T CARE” is used with the G.726 coder.
IPY00038240	--	121	Global Call IP (SIP)	A thread in the API is asserting when an inbound re-INVITE is received before GCEV_REQ_MODIFY_CALL.
IPY00038513	--	121	HMP	UIO offset does not account for data already played from a previous buffer and starts again at the beginning of the offset in the next buffer.

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN (Continued)

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00040524	--	121	Host Interface	Compiler warnings occur due to unused strings in the header files.
IPY00039985	--	121	IPML	Although the QOS fault threshold range is 50 - 1200 (x 100ms), the application can set the fault threshold to a range of 1 - 1200 (1000ms).
IPY00039409	--	121	IPML	An incorrect eQoSType, QOSTYPE_ROUNDTRIPLATENCY, is specified for QoSRTCPTimeout and QoSRTPTimeout.
IPY00039847	--	121	Signaling (H.323)	A RequestMode is not triggered upon CED detection in AUTO mode.
IPY00037541	--	121	SIP	When a GCEV_REQ_MODIFY_CALL event is returned, RTP port-address info in the event data stays unchanged until the **gc_AcceptModifyCall( )** function is called and a GCEV_ACCEPT_MODIFY_CALL event is received.
IPY00039874	--	121	Voice	A TDX_VAD event is not generated to the application when A-law code is used for an RTP connection.
IPY00038747	--	121	Voice	The channel hangs when calling the **dx_AdjSv( )** function while play is in progress.
IPY00038549	--	121	Voice	Ports are stuck in a CS_STOPD state when a caller hangs up during playback.
IPY00038475	--	121	Voice	The **dx_dial( )** function does not always a return a TDX_CALLP event when stopped with the **dx_stopch( )** function.
IPY00037386	--	121	Voice	There is always size discrepancy between the size specified in the iott.io_length and the file size created in the **dx_reciottdata( )** function.
IPY00038533	--	114	Runtime Libraries	A system wide global semaphore is not decremented correctly when a process exits.
IPY00037708	--	110	Diagnostics	The **`its_sysinfo`** tool, a standalone utility used to collect system data, is not collecting a full memory dump.
IPY00037542	--	110	Voice	Each call to a voice library function changes the syslog program.
IPY00037696	--	105	Installation	The **dlstop** operation does not stop the ssp_x86Linux_boot process.
IPY00037172	--	103	Global Call IP	The application returns a taskfail with IPERR_INVALID_PHONE_NUMBER as the reason.
IPY00037333	--	103	Global Call IP (SIP)	There is no audio when calling from a digital phone through Alcatel PBX to HMP.
IPY00036453	--	103	Runtime Tracing Facility (RTF)	The Temp directory created by the Runtime Tracing Facility, /tmp/RTF, does not get deleted. This causes subsequent starts to fail.

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN (Continued)

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00035875	--	98	Global Call IP	Global Call IP applications that were recompiled using Service Update 87, 93, or 96 need to be recompiled again with Service Update 98 or later.
IPY00035819	--	98	IP Media	**ipm_ResetQoSAlarmStatus( )** fails to reset QoS alarms allowing the same device to not receive the same alarm again.
IPY00035585	--	96	Global Call IP (SIP)	A send only re-invite message will cause HMP IPM stream to be interrupted.
IPY00035680	--	96	IP Media Session Control (RTP)	Audio gaps heard.
IPY00035477	--	93	IP	IPM channel hangs after receiving disconnect.
IPY00034413	--	93	IP Media Session Control (RTP)	Parameter checking behaves inconsistently when calling **ipm_StartMedia( )** function.
IPY00035761	--	93	Multimedia	Application crash when nonexistent file is added into MM_PLAY_INFO.
IPY00034415	--	87	IP Host	Missing GCEV_UNBLOCKED and GCEV_OPENEX events in RTF logging.
IPY00031519	--	84	HMP	If linked with libmml.so multimedia library, the child process could not quit cleanly even though the parent process had quit.
IPY00034035	--	84	IP Host	Received a “486 Busy Here” message instead of a “400 Bad Request” when sending an INVITE message containing content length and body fields but without Content Type field (that is, a malformed request).
IPY00034023	--	84	IP Host	Received an IPEC_SIPReasonStatus400BadRequest event reason instead of IPEC_SIPReasonStatusCANCEL when the call is canceled.
IPY00032607	34819	79	IP Host	Compiler error that IPPARM_1PCC_REJECT_VIDEO was not defined (missing from header file); related to feature support for introduced in SU 69.
IPY00031796	36742	69	Installation	When a SUSE* Linux* Enterprise Server 9 system does not have netsnmp installed, the dlgcsnmp service causes the OS to hang during startup. (Same as PTR 36507.)
IPY00024428	36507	69	Installation	When a SUSE* Linux* Enterprise Server 9 system does not have netsnmp installed, the dlgcsnmp service causes the OS to hang during startup. (Same as PTR 36742.)
IPY00010965	36127	64	Demos	Multimedia demo works on 1 channel only (resolved by improving to work on 4 channels).
IPY00011024	36664	64	IP	At 240 channel density when the intercall delay falls below 5 seconds, HMP call tear-down time results in in-coming calls being temporarily blocked (issue discovered through a Busy Hour Call Completion stress test).

Table 1. Issues Resolved in Dialogic® HMP 1.5LIN (Continued)

Defect Number	PTR Number	SU No.	Dialogic® Product or Component	Description
IPY00010567	36632	64	Voice	A dx_play( ) function call terminates unexpectly when it specifies in the IOTT playing from a combination of memory and disk file.
IPY00032235	36405	55	Kernel	Kernel fails with a QERROR_KILLTASK and error code 0x41107 after approximately 25 days.

Table 2 lists the known issues for this release.

Table 2. Known Issues in Dialogic® HMP 1.5LIN

Defect Number	PTR Number	Product or Component	Description
IPY00031514	36849	IP Host	Global Call SIP applications using SIP INFO and T.38 simultaneously may miss GCEV_EXTENSION events as call control library runs out of pre-allocated buffers. ***Workaround:*** Initialize more channels in **gc_Start( )** than being used.
IPY00006380	36684	OA&M	When using Red Hat Enterprise Linux 4, sporadic failure reports may appear on HMP startup for daemons Error Logger, Device Mapper, and Timeslot Doler. However, these appear to be false failure reports and do not appear to affect performance.
IPY00006505	34425	CSP	When running CSP applications with 120 or more channels, a stuck channel may occur occasionally when the TDX_BARGEIN event is missed during CSP operations. This issue has been observed in particular when Linux cron jobs are running at the same time. ***Workaround:*** Stop all cron jobs while running high channel density CSP applications.
IPY00032257	35742	DM/IP boards	When communicating with a DM/IP board in another server, some channels of the DM/IP board, during long duration calls, may affect the voice quality, causing missing digits and/or bad recordings; however, the channel will recover for the subsequent call. This is a known issue for DM/IP boards.
IPY00010586	35815	Global Call	A crash may occur when trying to run Global Call in 3PCC mode with SIP re-INVITE on more than 120 multimedia channels. ***Workaround:*** Do not use on more than 120 multimedia channels.
IPY00006322	36000	Multimedia	The **mm_Reset( )** function does not reset parameters to their defaults. ***Workaround:*** Use the **mm_SetParm( )** function to explicitly set parameters.

2.2 Restrictions and Limitations

Known restrictions and limitations in this release can be categorized as follows:

General

- For users who do not use Domain Name Service (DNS) or have a system with no DNS entry, the /etc/hosts file must contain an entry with the machine's hostname and IP address. The hostname must appear only once in the file. In a non-DNS environment, whenever the host IP address is changed, make sure to update it in the /etc/hosts file. The machine must be able to resolve its name to an IP address, either using DNS or the /etc/hosts file. If it cannot resolve its name, the HMP software will not start.
- SNMP is not supported when using the default netsnmp (version 5.1.2) that is provided with Red Hat* Enterprise Linux* 4.0. **Workaround:** Use the earlier 5.0.10 version of netsnmp.
- High I/O activity (for example, updatedb) during heavy HMP activity may cause an increase in error rates, such as degraded digit detection and voice quality.
- DM3Stderr and DebugAngel diagnostics tools are not supported in this release. As a way to debug Dialogic® DM3 firmware, all firmware prints are sent to /var/log/messages by default. They can be sent to stdout when the boot kernel is run in non-daemon mode. You must ensure that syslogger is enabled and running. To check, use "ps -ef | grep syslogger". If it is not running, you can start it using the Linux setup utility.
- If you run an application as root and then switch to a non-privileged user, any log files created in the first run will probably be read-only for the non-privileged user.
- The HMP system does not operate with board level products and system software releases installed on the same machine.
- All cron jobs should be scheduled for off-peak hours to avoid performance issues.

Fax

- For fax applications, the header file *srllib.h* must appear before the header file *faxlib.h* in the #include directive.

Global Call, IP, and IPML

- When sending nonstandard data in a Q.931 Facility message or an H.245 UII message, loss of the data has been observed in a small number of cases (0.03% failure rate) [IPY00032237 (PTR 35806), permanent limitation].

- Only IP-specific Global Call features are provided as described in the *Dialogic® Global Call IP Technology Guide*.
- If a call is made to HMP using NetMeeting* from a machine that does not have a sound card, the coder negotiation will fail and the call will be disconnected before a GCEV_ANSWERED event is generated.
- If a call is made to HMP using NetMeeting, the "secure outgoing calls" option must not be selected.
- Global Call applications using T.38 should call **gc_SetUserInfo(IPPARM_T38_CONNECT)** after GCEV_OFFERED, but before the next GC function call, for T.38 only calls. Otherwise, the call will fail and the application will get a GCEV_TASKFAIL event.
- Host applications should always clean up resources before exiting by using one of the following methods:
  - If using Global Call call control, the application should call **gc_DropCall()** followed by **gc_ReleaseCall()**, or use **gc_ResetLineDev()**.
  - If using a non-Global Call call control stack, the application should stop RTP sessions by using **ipm_Stop( )**.

Voice

- The **dx_reciottdata()** and **dx_playiottdata()** functions do not support recording or playing in WAVE format to/from memory; only VOX format.
- Global Tone Detection (GTD) does not support detecting user defined tones as digits via the digit queue; the tones are only detected as tone events via the event queue.
- For dual tone definition, the frequency deviation that is defined in the tone template for each frequency must be not less than ± 30 Hz.
- For single tone definition, the frequency deviation that is defined in the tone template for each frequency must be not less than ± 30 Hz.
- The number of tone templates which can be added to a voice device and enabled for detection is limited. The maximum number of events for each instance is 10.
- The **dx_dial()** function call progress analysis only detects CED tones for CR_FAXTONE event. CNG tones cannot be detected.
- The **dx_playtoneEx()** function will not terminate after the time specified by the DX_MAXTIME termination parameter has elapsed.
- DTMF digits not processed by the user application with **dx_getdig()** or other means remain in the device after it is closed with **dx_close()**, and remain with the device even after the application terminates. To ensure that the buffer is empty, clear the digit buffer using **dx_clrdigbuf()**.
- The **dx_getctinfo()** and **fx_getctinfo()** functions return incorrect values for Device Family and Bus Mode.

Demos

- Demos should be run using from 1-4 channels, and the channel numbers selected in the demo must be between 1-120; otherwise the demo could fail.

2.3 Compatibility Notes

Echo Cancellation With Continuous Speech Processing

For Dialogic® HMP Continuous Speech Processing (CSP), the Echo Cancellation parameter default value is set to OFF (echo cancellation disabled). To conserve CPU usage/MIPs, the application should keep this parameter value set to OFF.

For additional information about CSP, refer to the *Dialogic® Continuous Speech Processing API Library Reference* and the *Dialogic® Continuous Speech Processing API Programming Guide*.

Echo Cancellation With Conferencing

For HMP conferencing, the Echo Cancellation parameter default value is set to OFF (echo cancellation disabled) for compatibility with DCB applications. This parameter must not be turned ON by the application. Enabling echo cancellation causes poor conference audio quality.

For additional information about conferencing, refer to the *Dialogic® Audio Conferencing API Library Reference* and the *Dialogic® Audio Conferencing API Programming Guide*.

UDP/RTP Audio, Video, and Fax Port Ranges

The HMP system defaults to the following UDP/RTP port ranges:

- Ports 6000 through 6100 for **T.38 Fax**
- Ports 49152 through 49xxx for RTP **audio** streaming
- Ports 57344 through 57xxx for RTP **video** streaming

where xxx = port range base number + twice the maximum number of channels purchased under the licensing agreement. For example, if 120 voice resources are licensed, the port range for RTP audio streaming would be 49152 through (49152 + 240 = 49392).

See the following sections for information on configuring the ranges.

Configuring the UDP/RTP Audio Port Range

If the UDP/RTP audio port range used by the HMP system conflicts with other RTP service, the following procedure describes how to configure HMP to use a different port range for audio streaming:

1. Stop the system service.

2. Locate the .config file in the */usr/dialogic/data* directory that matches the FCD/PCD files associated with your licensed configuration.

3. Using a text editor, open the *.config* file.

4. Locate the [IPVSC] section in the .config file.

5. In the [IPVSC] section, locate the line

```
setparm 0x4005, 49152   !set the rtpPortBase on IPVSC
```

   The number 49152 is the default value for this parameter. You may change the beginning of the UDP/RTP port range by first editing this value and saving the *.config* file.

6. After you have saved the *.config* file with the new UDP/RTP port value, open the Command Prompt window.

7. From the Command Prompt, change the directory to /*usr/dialogic/data*.

8. Execute *fcdgen* as follows:

```
/usr/dialogic/bin/fcdgen -f <input filename>.config -o <output
filename>.fcd
```

   The resulting FCD file is created in the */usr/dialogic/data* directory. If the -o option is omitted from the command, the default output FCD file will have the same filename as the user-modified input .config file, but with an .fcd extension.

9. Restart the system service to download the new configuration to HMP.

Configuring the T.38 Fax Service Port Range

The HMP system currently defaults to port 6000 as the starting UDP/RTP port for the T.38 service port. If the T.38 service port range used by the HMP system conflicts with other RTP service, the following procedure describes how to configure HMP to use a different T.38 service port range:

1. Stop the system service.

2. Locate the .config file in the */usr/dialogic/data* directory that matches the FCD/PCD files associated with your licensed configuration.

3. Using a text editor, open the *.config* file.

4. Locate the [0xe] section in the *.config* file.

5. In the [0xe] section, locate the line

```
setparm 0x4c21, 6000 ! QFC3_PrmIPRxPortBase (QFC3_PrmLocalUDPPortBase)
```

The number 6000 is the default value for this parameter. You may change the beginning of the T.38 service port range by first editing this value and saving the *.config* file.

6. After you have saved the .config file with the new T.38 service port value, open the Command Prompt window.

7. From the Command Prompt, change the directory to */usr/dialogic/data*.

8. Execute *fcdgen* as follows:

```
/usr/dialogic/bin/fcdgen -f <input filename>.config -o <output
filename>.fcd
```

   The resulting FCD file is created in the */usr/dialogic/data* directory. If the -o option is omitted from the command, the default output FCD file will have the same filename as the user-modified input .config file, but with an .fcd extension.

9. Restart the system service to download the new configuration to HMP.

H.323 Coder Negotiation with Third Party Stacks

Use caution when restricting coder frame sizes or frames per packet while communicating with third-party H.323 stacks. The IPT H.323 protocol stack uses both coder type and frames per packet as part of the algorithm to determine a successful Tx/Rx media match. Restricting coder frame sizes can cause Fast Start calls to fallback to Slow Start or coder negotiation to fail. See the following example.

Example:

A Global Call application configures a channel for G.711 A-Law with a frame size of 10 milliseconds. A third-party H.323 stack, which is configured for G.711 A-Law with a frame size of 30 milliseconds, initiates a call to the application. The IPT H.323 protocol stack will use 10 milliseconds as an upper limit for both the Tx and Rx media directions. Even though both sides support the same coder, the frame size discrepancy can cause the coder negotiation to fail, resulting in a GCEV_DISCONNECTED event.

2.4 Operating System Notes

Assertions occur when NICs are under heavy load

When using multiple network interface cards (NICs) under heavy loads, kernel assertions may occur, resulting in call failure.

Symptom: Calls fail with the following message displayed:

```
kernel: KERNEL: assertion (flags & MSG_PEEK) > failed at net/ipv4/tcp.c
(1284)
```

Solutions

Resolution 1: If you have multiple NICs on one segment (e.g., eth0: 192.168.66.21, eth1: 192.162.66.72), turn off forwarding on at least one of the NICs.

```
sysctl -w net.ipv4.conf.eth0.forwarding=0
sysctl -w net.ipv4.conf.eth1.forwarding=0
```

Resolution 2: If you have a single NIC, do the following:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=2048
sysctl -w net.ipv4.route.max_size=1048576
sysctl -w net.ipv4.route.gc_thresh=65536
```

Elvtune Using IOCTLS

Elvtune using IOCTLS is now deprecated. To tune the IO scheduler, use the files exported to the *sysfs* directory.

```
/sys/block/<device>/queue/iosched
```

Documentation Updates

This chapter provides documentation updates for the documents in the on-line bookshelf for Dialogic® Host Media Processing (HMP) Software Release for 1.5LIN. The following sections present the document updates organized according to the type of document:

3.1 Release Documentation

This section contains updates to the following documents:

- Dialogic® Host Media Processing Software Release 1.5LIN Release Guide

3.1.1 Dialogic® Host Media Processing Software Release 1.5LIN Release Guide

There are currently no updates to this document.

3.2 Installation Documentation

This section contains updates to the following documents:

- Dialogic® Host Media Processing Software Release 1.5LIN Software Installation Guide

3.2.1 Dialogic® Host Media Processing Software Release 1.5LIN Software Installation Guide

The following information applies:

- In **Section 2.3 "Installing the HMP Software"** the instructions incorrectly say to install the software from a CD-ROM disk, whereas the software must be downloaded from the web using the following installation instructions.

Instructions for Downloading and Installation

1. Download the Dialogic® HMP 1.5LIN package (or a service update) from http://dialogic.com/telecom/support/hmplinux/hmp15/ and then untar the package. The software contains the following install scripts:
   - The top level script installs runtime components and header files (run this script to be able to compile your application) (**Recommended**)
   - The script in the redistributable-runtime directory installs runtime components (no header files except for demo programs)
   - The script in the sdk directory installs header files for application development

2. Select the desired script and cd to the appropriate directory. Then enter the following command to start the install script:

   ```
   ./install.sh
   ```

Continue with the installation instructions in **Section 2.3, "Installing the HMP Software,"** of the *Dialogic® Host Media Processing Software Release 1.5LIN Software Installation Guide*.

3.3 Programming Libraries Documentation

This section contains updates to the following documents:

- Dialogic® Audio Conferencing API Library Reference
- Dialogic® Audio Conferencing API Programming Guide
- Dialogic® Global Call IP Technology Guide
- Dialogic® IP Media Library API Library Reference
- Dialogic® Multimedia API Library Reference
- Dialogic® Multimedia API Programming Guide
- Dialogic® Voice API Library Reference
- Dialogic® Voice API Programming Guide

3.3.1 Dialogic® Audio Conferencing API Library Reference

The following information applies:

Update to **dcb_getbrdparm( )** and **dcb_setbrdparm( )** functions (IPY00006584 = PTR 36199)

Changed the description of the MSG_ACTID parameter to indicate that it enables/disables active talker identification (or notification) and not the active talker feature itself. Replace the MSG_ACTID parameter with the following:

MSG_ACTID (Active Talker Identification)

- Enables or disables Active Talker Identification (or Notification). Possible values are ACTID_ON or ACTID_OFF. ACTID_ON is the default. This parameter does not enable or disable the active talker *feature*, which is always enabled. It only

disables the *notification* to the application program. The active talker *feature* sums the 3 most active talkers in a conference, so that the conversation doesn't get drowned out when too many people talk at once. Active talker *notification* provides data on active talkers through the **dcb_gettalkers( )** and **dcb_GetAtiBitsEx( )** functions, which can be used by an application program to identify active talkers; for example, to provide a visual display highlighting the active talkers in a conference. Active talkers are determined by their loudness; i.e., the strength of their "non-silence" energy.

Note: In some cases, it is desirable to inactivate the active talker feature, such as for a background music application program. Although you cannot directly disable the active talker *feature*, you can set the noise level threshold by which signals are recognized as either speech or noise. For more information, see the background music feature in the *Audio Conferencing API Programming Guide*.

3.3.2 Dialogic® Audio Conferencing API Programming Guide

The following information applies:

Update to Section 6.2, "Initialization of DM3 Board Parameters" and to "Active Talker" chapter (IPY00006584 = PTR 36199)

Changed the description of the MSG_ACTID parameter to indicate that it enables/disables active talker identification (or notification) and not the active talker feature itself. Replace the MSG_ACTID parameter with the following:

MSG_ACTID (Active Talker Identification)

- Enables or disables Active Talker Identification (or Notification). Possible values are ACTID_ON or ACTID_OFF. ACTID_ON is the default. This parameter does not enable or disable the active talker *feature*, which is always enabled. It only disables the *notification* to the application program. The active talker *feature* sums the 3 most active talkers in a conference, so that the conversation doesn't get drowned out when too many people talk at once. Active talker *notification* provides data on active talkers through the **dcb_gettalkers( )** and **dcb_GetAtiBitsEx( )** functions, which can be used by an application program to identify active talkers; for example, to provide a visual display highlighting the active talkers in a conference. Active talkers are determined by their loudness; i.e., the strength of their "non-silence" energy.

Note: In some cases, it is desirable to inactivate the active talker feature, such as for a background music application program. Although you cannot directly disable the active talker *feature*, you can set the noise level threshhold by which signals are recognized as either speech or noise. For more information, see the background music feature in the *Audio Conferencing API Programming Guide*.

Update to "Background Music" chapter (IPY00010946 = PTR 36323)

Changed description of how to implement background music in an application so that it doesn't refer to MSG_ACTID. Replace the instructions on how to implement background music with the following:

Do the following to implement background music in a conference:

- Create a three-party conference, where one party is the music resource.

- When you add music to the conference, make the following parameter settings. Set its party attributes so that it uses transmit-only mode. That is, for the conference party that transmits music, enable the MSPA_MODEXMITONLY attribute in the MS_CDT data structure **chan_attr** field.
- The Conferencing AGC Noise Level Lower Threshold may need to be adjusted. This parameter is set at -40 dBm by default and filters out any signals below this level. If the background music levels are low, the music may not be summed into the conference by the active talker feature, or it may come in and out. In this case, the AGC Noise Level Lower Threshold should be reduced by adding or changing the CSUMS_AGC_low_threshold (0x3B1F) parameter in the configuration file for the board and re-starting the board. For more information on this parameter, see the applicable configuration guide; e.g., *Dialogic® on DM3 Architecture Configuration Guide*, or the configuration guide for the Dialogic® Host Media Processing (HMP) software.

3.3.3 Dialogic® Global Call IP Technology Guide

Note: For additional updates, see also Section 1.7, "H.323 Annex M (Tunneled Signaling Message)", on page 24 .

The following information applies:

- Update Section 8.3.17, "**gc_MakeCall( )** Variances for IP" (IPY00029956 = PTR 36646) the last paragraph before Section 8.3.17.1 (page 381) as follows:

  When using SIP, if the remote side does not send a final response to an outgoing INVITE (sent by the call control library) within 64 seconds, the **gc_MakeCall( )** function times out and the library generates a GCEV_DISCONNECTED event to the application. If the application attempts to drop the call before the 64 second time-out is reached, the library's behavior depends on whether a provisional response was received. When no provisional response was received before the application cancels the call, the library cleans up the call immediately. But if a provisional response was received before the application attempts to cancel the call, the library sends a CANCEL to the remote endpoint and generates a GCEV_DROPCALL to the application after it receives the 200OK response to the CANCEL and a 487RequestTerminated response for the original INVITE, or when an additional 32-second time-out expires.
- On the reference page for the IP_H221NONSTANDARD data structure (page 465), the descriptions of the three data fields are updated as follows:

  country_code
  : The country code. Range: 0 to 255; any value x>255 is treated as x%256.

  extension
  : The extension number. Range: 0 to 255; any value x>255 is treated as x%256.

  manufacturer_code
  : The manufacturer code. Range: 0 to 65535; any value x>65535 is treated as x%65636.

3.3.4 Dialogic® IP Media Library API Library Reference

The following information applies to the *Dialogic® IP Media Library API Library Reference* (05-2257-005).

- The IPMEV_QOS_ALARM event description, on page 84, indicates that “No data is returned in the event.” This statement is misleading because it implies that it is not possible to determine which alarm triggered the event. Replace the description with the following text:

  IPMEV_QOS_ALARM
  Unsolicited event enabled via **ipm_EnableEvents( )**. Event is returned when desired QoS alarm gets triggered. For information on determining which alarm triggered the event, see the code example, and specifically the **CheckEvent( )** function definition, in the *IP Media Library API for Host Media Processing Programming Guide* section on “Example Code for QoS Alarm Handling”.

- Add the following values to the eIPM_DATA_DIRECTION enumeration for the **eDirection** parameter of the **ipm_StartMedia( )** function.
  - DATA_MULTICAST_CLIENT - Multicast receive. (RTP only.)
  - DATA_MULTICAST_SERVER - Multicast transmit.
- Add the following code example to the **ipm_StartMedia( )** function, showing how to set up a channel for multicast receive.

Example (Multicast Receive)

```
#include <stdio.h>
#include <string>
#include <srllib.h>
#include <ipmlib.h>

typedef long int(*HDLR)(unsigned long);
void CheckEvent();

void main()
{
    int nDeviceHandle;

    // Register event handler function with srl
    sr_enbhdlr( EV_ANYDEV ,EV_ANYEVT ,(HDLR)CheckEvent);

    /*
    .
    .
    Main Processing
    .
    .
    */

    /*
    Set the media properties for a remote party using IP device handle, nDeviceHandle.
    ASSUMPTION: A valid nDeviceHandle was obtained from prior call to ipm_Open().
    */
    IPM_MEDIA_INFO MediaInfo;
    MediaInfo.unCount = 5;

    MediaInfo.MediaData[0].eMediaType = MEDIATYPE_AUDIO_LOCAL_RTP_INFO;
    MediaInfo.MediaData[0].mediaInfo.PortInfo.unPortId = 8910;
    strcpy(MediaInfo.MediaData[0].mediaInfo.PortInfo.cIPAddress,"234.5.6.7");
```

```
    MediaInfo.MediaData[1].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTP_INFO;
    MediaInfo.MediaData[1].mediaInfo.PortInfo.unPortId = 2328;
    strcpy(MediaInfo.MediaData[1].mediaInfo.PortInfo.cIPAddress,"192.168.86.18");

    MediaInfo.MediaData[2].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTCP_INFO;
    MediaInfo.MediaData[2].mediaInfo.PortInfo.unPortId = 2329;
    strcpy(MediaInfo.MediaData[2].mediaInfo.PortInfo.cIPAddress,"192.168.86.18");

    MediaInfo.MediaData[3].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_INFO;
    MediaInfo.MediaData[3].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ULAW64K;
    MediaInfo.MediaData[3].mediaInfo.CoderInfo.eFrameSize = (eIPM_CODER_FRAMESIZE) 10;
    MediaInfo.MediaData[3].mediaInfo.CoderInfo.unFramesPerPkt = 1;
    MediaInfo.MediaData[3].mediaInfo.CoderInfo.eVadEnable =CODER_VAD_DISABLE;
    MediaInfo.MediaData[3].mediaInfo.CoderInfo.unCoderPayloadType = 0;
    MediaInfo.MediaData[3].mediaInfo.CoderInfo.unRedPayloadType = 0;

    MediaInfo.MediaData[4].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
    MediaInfo.MediaData[4].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ULAW64K;
    MediaInfo.MediaData[4].mediaInfo.CoderInfo.eFrameSize = (eIPM_CODER_FRAMESIZE) 30;
    MediaInfo.MediaData[4].mediaInfo.CoderInfo.unFramesPerPkt = 1;
    MediaInfo.MediaData[4].mediaInfo.CoderInfo.eVadEnable = CODER_VAD_DISABLE;
    MediaInfo.MediaData[4].mediaInfo.CoderInfo.unCoderPayloadType = 0;
    MediaInfo.MediaData[4].mediaInfo.CoderInfo.unRedPayloadType = 0;


    if(ipm_StartMedia(nDeviceHandle, &MediaInfo, DATA_MULTICAST_CLIENT, EV_ASYNC) == -1)
    {
        printf("ipm_StartMediaInfo failed for device name = %s with error = %d\n",
            ATDV_NAMEP(nDeviceHandle), ATDV_LASTERR(nDeviceHandle));
        /*
        .
        .
        Perform Error Processing
        .
        .
        */
    }

    /*
    .
    .
    Continue processing
    .
    .
    */
}

void CheckEvent()
{
    int nDeviceID = sr_getevtdev();
    int nEventType = sr_getevttype();

    switch(nEventType)
    {
        /*
        .
        .
        Other events
        .
        .
        */

        /* Expected reply to ipm_StartMedia */
        case IPMEV_STARTMEDIA:
            printf("Received IPMEV_STARTMEDIA for device = %s\n", ATDV_NAMEP(nDeviceID));
            break;
```

```
        default:
            printf("Received unknown event = %d for device = %s\n",
                nEventType, ATDV_NAMEP(nDeviceID));
            break;
    }
}
```

- Remove the first caution of the **ipm_GetLocalMediaInfo( )** function (on page 27) and add the following text to the Description section. This clarifies eMediaType and unCount as members of data structures referenced by the **pMediaInfo** function parameter and enumerates the allowed values for the eMediaType field (in the IPM_MEDIA data structure) and corresponding unCount values.
  - To retrieve RTP port information for both audio and video in a multimedia session, set the eMediaType fields to MEDIATYPE_AUDIO_LOCAL_RTP_INFO and MEDIATYPE_VIDEO_LOCAL_RTP_INFO respectively and unCount to 2. See the code example. *[Corrected below.]*
  - To retrieve RTP port information for a video-only session, set the eMediaType field to MEDIATYPE_VIDEO_LOCAL_RTP_INFO and unCount to 1.
  - To retrieve RTP port information for an audio-only session, set the eMediaType field to MEDIATYPE_AUDIO_LOCAL_RTP_INFO and unCount to 1.
  - To retrieve T.38 fax port information, set the eMediaType field to MEDIATYPE_LOCAL_UDPTL_T38_INFO and unCount to 1.

  Note: It is not possible to retrieve T.38 fax port information together with audio and/or video port information.

  Note: The RTCP port number is the RTP port number + 1.
- In the code example of the **ipm_GetLocalMediaInfo( )** function, replace the line:
  MediaInfo.unCount = 1;
  with:
  MediaInfo.unCount = 2;
- Replace the IPM_MEDIA data structure Field Description for eIPM_MEDIA_TYPE with the following corrected content:

  eMediaType

  type of media used to start an IP session

  The eIPM_MEDIA_TYPE data type is an enumeration which defines the following values:

  Audio Values:
  - MEDIATYPE_AUDIO_LOCAL_RTP_INFO
  - local RTP audio port information
  - MEDIATYPE_AUDIO_LOCAL_RTCP_INFO
  - local RTCP audio port information
  - MEDIATYPE_AUDIO_LOCAL_CODER_INFO
  - local receive audio coder information
  - MEDIATYPE_AUDIO_REMOTE_RTP_INFO
  - remote RTP audio port information
  - MEDIATYPE_AUDIO_REMOTE_RTCP_INFO

- remote RTCP audio port information
- MEDIATYPE_AUDIO_REMOTE_CODER_INFO
- remote receive audio coder information

Video Values:

- MEDIATYPE_VIDEO_LOCAL_RTP_INFO
- local RTP video port information
- MEDIATYPE_VIDEO_LOCAL_RTCP_INFO
- local RTCP video port information
- MEDIATYPE_VIDEO_LOCAL_CODER_INFO
- local receive video coder information
- MEDIATYPE_VIDEO_REMOTE_RTP_INFO
- remote RTP video port information
- MEDIATYPE_VIDEO_REMOTE_RTCP_INFO
- remote RTCP video port information
- MEDIATYPE_VIDEO_REMOTE_CODER_INFO
- remote receive video coder information

Fax Values:

- MEDIATYPE_FAX_SIGNAL
- fax signal information to be transmitted towards IP during fax transmissions
- MEDIATYPE_LOCAL_UDPTL_T38_INFO
- local UDP packet T.38 information
- MEDIATYPE_REMOTE_UDPTL_T38_INFO
- remote UDP packet T.38 information

Note: The following eMediaType equates are also provided for backward compatibility (the earlier generic names are equated with audio port values). However, it is recommended that the AUDIO values be used.

```
#define MEDIATYPE_REMOTE_RTP_INFO    MEDIATYPE_AUDIO_REMOTE_RTP_INFO
#define MEDIATYPE_LOCAL_RTP_INFO     MEDIATYPE_AUDIO_LOCAL_RTP_INFO
#define MEDIATYPE_REMOTE_RTCP_INFO   MEDIATYPE_AUDIO_REMOTE_RTCP_INFO
#define MEDIATYPE_LOCAL_RTCP_INFO    MEDIATYPE_AUDIO_LOCAL_RTCP_INFO
#define MEDIATYPE_REMOTE_CODER_INFO  MEDIATYPE_AUDIO_REMOTE_CODER_INFO
#define MEDIATYPE_LOCAL_CODER_INFO   MEDIATYPE_AUDIO_LOCAL_CODER_INFO
```

Update to Chapter 2, **Function Information**

- Added a parameter ID, **IPPARM_1PCC_OPTIONS_RESPONSE_NO_SDP**, to **gc_util_insert_parm_val( )** to disable inclusion of an SDP message with the OPTIONS response in 1PCC mode. For more information, see Section 1.5, “Disable Automatic SDP Message Generation”, on page 18.

3.3.5 Dialogic® Multimedia API Library Reference

Update to Chapter 5, **Data Structure Reference**

- Added MM_DATA_FORMAT_ALAW coding format to the MM_AUDIO_CODEC unCoding field. For more information about this feature, see Section 1.4, “Using Multimedia API for Play and Record”, on page 15.
- The **unOffset** field is missing from the MM_MEDIA_AUDIO data structure data structure. This field is defined as unsigned integer and is reserved for future use.

3.3.6 Dialogic® Multimedia API Programming Guide

The following information applies:

- Updated the *Dialogic® Multimedia API Programming Guide* from 05-2455-001to 05-2455-002 to accommodate removal of Chapter 5, which was updated and moved to the *Dialogic® Multimedia File Conversion Tools User Guide* (05-2453-001), now available at:

  http://dialogic.com/telecom/support/hmpmedia/index.htm

3.3.7 Dialogic® Voice API Library Reference

The following new API is added to resolve IPY00038706:

- **dx_resetch( )** - Call this API to recover the media channel when the channel is stuck and in a recoverable state. If the channel is recovered, a TDX_RESET event is generated to the application, which enables the application to reuse the channel for more media functions. If the channel is not in a recoverable state, a TDX_RESETERR event is sent back to the application indicating that the specific channel is not recoverable.

dx_resetch( )

Name: dx_resetch (chdev, mode)

Inputs: int chdev
- valid channel device handle

int mode
- mode of operation

Returns: 0 if success
-1 if failure

Includes: srllib.h
dxxxlib.h

Category: I/O

Mode: asynchronous or synchronous

Dialogic® Platform: DM3

Description

The **dx_resetch( )** function recovers a channel that is "stuck" (busy or hung) and in a recoverable state, and brings it to an idle and usable state. This function blocks all other functions from operating on the channel until the function completes.

Parameter	Description
chdev	Specifies the valid device handle obtained when the channel was opened using **dx_open( )**
mode	Specifies the mode of operation: • EV_ASYNC – asynchronous mode. The calling thread returns immediately so it can process media functionality on other channels. • EV_SYNC – synchronous mode. The calling thread waits until the channel is recovered or discovers that the channel is not in a recoverable state.

In synchronous mode, 0 is returned if the function completes successfully, and -1 is returned in case of error.

In asynchronous mode, the TDX_RESET event is generated to indicate that the channel was recovered and is in an idle and usable state. The TDX_RESETERR event is generated to indicate that the channel is not recoverable. Issuing any other media calls on this channel will result in an error.

Cautions

- The **dx_resetch( )** function is intended for use on channels that are stuck and not responding. Do **not** use it in place of **dx_stopch( )**. Use **dx_resetch( )** only if you do not receive an event within 30 seconds of when it's expected. Overuse of this function creates unnecessary overhead and may affect system performance.

Errors

If the function returns -1, use the Dialogic® Standard Runtime Library (SRL) Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

EDX_BADPARM
Invalid parameter

EDX_FWERROR
Firmware error

EDX_NOERROR
No error

Example

```
#include <srllib.h>
#include <dxxxlib.h>

main()
{
   int chdev, srlmode;
   /* Set SRL to run in polled mode. */
   srlmode = SR_POLLMODE;

   if (sr_setparm(SRL_DEVICE, SR_MODEID, (void *)&srlmode) == -1) {
   /* process error */
   }

   /* Open the channel using dx_open( ). Get channel device descriptor in
   * chdev.
   */

   if ((chdev = dx_open("dxxxB1C1",NULL)) == -1) {
   /* process error */
   }

   /* continue processing */
   . .
   /* Force the channel to idle state. The I/O function that the channel
    * is executing will be terminated, and control passed to the handler
    * function previously enabled, using sr_enbhdlr(), for the
    * termination event corresponding to that I/O function.
    * In asynchronous mode, dx_stopch() returns immediately,
    * without waiting for the channel to go idle.
    */

   if ( dx_stopch(chdev, EV_ASYNC) == -1) {
   /* process error */
   }

   /* Wait for dx_stopch() to stop the channel and return the termination event
    * for the present media function.
    */

   /* After waiting for 30 secs if the termination event is not returned, issue a
    * dx_resetch() to reset the channel.
    */

   if (dx_resetch(chdev, EV_ASYNC) <0 )
   {
      /*process error */
```

```
    }

    /* Wait for TDX_RESET or TDX_RESETERR events */

}
```

In the DX_XPB data structure reference section in Chapter 4, the following corrections should be made to indicate that Linear PCM, 8 kHz, 16-bit (128 kbps) encoding method is supported:

- In Field Descriptions, wDataFormat, the correct description for DATA_FORMAT_PCM is "8-bit or 16-bit PCM".
- In Examples, Table 20, Linear PCM Voice Coder Support Fields, the correct values for nSamplesPerSec are "DRT_8KHZ or DRT_11KHZ". The correct values for wBitsPerSample are "8 or 16".

The following caution should be added to the Cautions section of the **dx_listenEx( )** function:

- It is recommended that you use **dx_listenEx( )** and **dx_unlistenEx( )** in your application, rather than **dx_listen( )** and **dx_unlisten( )**. In particular, do not use both pairs of functions on the same channel. Doing so may result in unpredictable behavior.

The following caution should be added to the Cautions section of the **dx_unlistenEx( )** function:

- It is recommended that you use **dx_listenEx( )** and **dx_unlistenEx( )** in your application, rather than **dx_listen( )** and **dx_unlisten( )**. In particular, do not use both pairs of functions on the same channel. Doing so may result in unpredictable behavior.

3.3.8 Dialogic® Voice API Programming Guide

The following voice encoding method is supported and should be added to Chapter 8, Recording and Playback, in Table 6, Voice Encoding Methods:

- Linear PCM, 8 kHz 16-bit (128 kbps) VOX and WAVE file formats

3.4 Demonstration Software Documentation

This section contains updates to the following documents:

- Dialogic® Continuous Speech Processing API Demo Guide

3.4.1 Dialogic® Continuous Speech Processing API Demo Guide

For Linux, the Continuous Speech Processing API Demo is located in:

$(DIALOGIC_DIR)/demos/SpeechProcessing/CSPDemo/Release/

The demo is started by executing the following command:

```
./CSPDemo -<option>
```