



Dialogic® PowerMedia™ HMP for Linux Release 4.1

Release Update

May 16, 2018

Copyright and Legal Notice

Copyright © 2009-2018 Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation and its affiliates or subsidiaries ("Dialogic"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see <http://www.dialogic.com/company/terms-of-use.aspx> for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at www.dialogic.com.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic's legal department at 3300 Boulevard de la Côte-Vertu, Suite 112, Montreal, Quebec, Canada H4R 1P8. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Brooktrout, BorderNet, PowerMedia, PowerVille, PowerNova, ControlSwitch, I-Gate, Veraz, Cantata, TruFax, and NMS Communications, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 3300 Boulevard de la Côte-Vertu, Suite 112, Montreal, Quebec, Canada H4R 1P8. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

Using the AMR-NB resource in connection with a Dialogic® PowerMedia™ HMP product does not grant the right to practice the AMR-NB standard. To seek a patent license agreement to practice the standard in connection with PowerMedia HMP, contact the VoiceAge Corporation at licensing@voiceage.com.

Publication Date: May 16, 2018

Document Number: 05-2680-035

About This Publication

This section contains information about the following topics:

- [Purpose](#)
- [Intended Audience](#)
- [How to Use This Publication](#)
- [Related Information](#)

Purpose

This Release Update addresses issues associated with the Dialogic® PowerMedia™ HMP for Linux Release 4.1 (formerly known as Dialogic® Host Media Processing Software Release 4.1LIN). In addition to summarizing issues that were known as of this release, it is intended that the Release Update will continue to be updated to serve as the primary mechanism for communicating any new issues that may arise after the release date.

Intended Audience

This Release Update is intended for all users of the Dialogic® PowerMedia™ HMP for Linux Release 4.1.

How to Use This Publication

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

- [Document Revision History](#): This section summarizes changes and additions that have been 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 the Service Update are described in this section.
- [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 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 documentation not made prior to the release. These 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 4.1LIN Release Guide*, which is included as part of the documentation bookshelf for the release.
- <http://www.dialogic.com/manuals> (for Dialogic® product documentation)
- <http://www.dialogic.com/support> (for Dialogic technical support)
- <http://www.dialogic.com> (for Dialogic® product information)

Document Revision History

This Revision History summarizes the changes made in this and each previously published version of the Release Update for Dialogic® PowerMedia™ HMP for Linux Release 4.1, which is a document that is planned to be periodically updated throughout the lifetime of the release.

Document Rev 35 (Updated) - published May 16, 2018

Updates for Service Update 236.

In the [Post-Release Developments](#) chapter:

- Updated [DTLS-SRTP Support](#).

Document Rev 35 (Updated) - published April 19, 2018

Updates for Service Update 236.

In the [Post-Release Developments](#) chapter:

- Added [MSML Media Server Software Deprecation](#).

Document Rev 35 - published March 29, 2018

Updates for Service Update 236.

In the [Post-Release Developments](#) chapter:

- Added [Installation Package Policy](#).
- Added [Updated Operating System Support](#).

In the [Documentation Updates](#) chapter:

- Added [Installation Package Policy](#) in the [Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide](#).

Document Rev 34 (Updated) - published January 24, 2018

Updates for Service Update 236.

In the [Post-Release Developments](#) chapter:

- Updated [STUN Message Support](#).

Document Rev 34 - published October 13, 2017

Updates for Service Update 236.

In the [Release Issues](#) chapter:

- Added the following Known (permanent) Issues: IPY00116826.

Document Rev 33 - published April 11, 2017

Updates for Service Update 236.

In the [Release Issues](#) chapter:

- Added the following Resolved Defects: IPY00115803, IPY00117467, IPY00117518, IPY00117588, IPY00117628, IPY00117649, IPY00117723, IPY00117825, IPY00117826, IPY00117846, IPY00117860, IPY00117870, IPY00117871, IPY00117917, IPY00117919, IPY00117925, IPY00117953, IPY00117979, IPY00117984, IPY00117987, IPY00118012, IPY00118061, IPY00118064, IPY00118141, IPY00118186, IPY00118219.
- Added the following Known (permanent) Issues: IPY00118040.

Document Rev 32 - published June 21, 2016

In the [Post-Release Developments](#) chapter:

- Added [VMware ESXi 6.x Support](#).
- Updated [AMR2 Audio Codec Support](#).

Document Rev 31 - published January 8, 2016

Updates for Service Update 213.

In the [Post-Release Developments](#) chapter:

- Added [DTLS-SRTP Support](#).
- Added [STUN Message Support](#).
- Added [Opus Audio Codec Support](#).
- Added [iLBC Audio Codec Support](#).
- Added [VP8 Video Codec Support](#).
- Added [RTCP Feedback Support](#).
- Added [VGA 480x640 Aspect Ratio H.264 Video Support](#).
- Added [3GP Multimedia Container for Play and Record Support](#).
- Added [G.729 Codec with 60ms Packet Support](#).
- Added [AMR2 Audio Codec Support](#).
- Added [Send/Receive RFC 2833/RFC 4733 Tone Events Support](#).
- Added [Expanded Interface Identification Support](#).
- Added [Red Hat Enterprise Linux Release 7 Update 1 Support](#).
- Added [SUSE Linux Enterprise Server 11 Service Pack 3 64-bit Support](#).
- Added [Oracle Enterprise Linux 6.2 64-bit Support](#).

In the [Post-Release Developments](#) chapter:

- Added a note about recompiling applications in the [Service Update](#) section.
- Removed the GCC 3.2.3 and GCC 3.4.3 compiled packages in the [Operating System Distributions No Longer Supported](#) section.

In the [Release Issues](#) chapter:

- Added the following Resolved Defects: IPY00101895, IPY00115335, IPY00115554, IPY00115874, IPY00115919, IPY00115963, IPY00116366, IPY00116411, IPY00116430, IPY00116443, IPY00116463, IPY00116477, IPY00116482, IPY00116490, IPY00116509, IPY00116541, IPY00116593, IPY00116597, IPY00116616, IPY00116681, IPY00116709, IPY00116714, IPY00116740, IPY00116741, IPY00116745, IPY00116793, IPY00116798, IPY00116840, IPY00116922, IPY00117002, IPY00117011, IPY00117073, IPY00117109, IPY00117156, IPY00117161, IPY00117209, IPY00117262, IPY00117285, IPY00117286, IPY00117300, IPY00117362, IPY00117377, IPY00117383, IPY00117401, IPY00117478.
- Added the following Known Issue: HMP-245, HMP-263.
- Added the following Known (permanent) Issues: HMP-126, IPY00102460.

In the [Documentation Updates](#) chapter:

- Updated the minimum required memory to 8 GB of RAM in the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#).

Document Rev 30 - published April 11, 2014

Updates for Service Update 165.

In the Post-Release Developments chapter, added:

- [Operating System Distributions No Longer Supported](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00116226, IPY00116257.

In the Documentation Updates chapter:

- Updated with list of supported operating systems in the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#).

Document Rev 29 - published January 29, 2014

Updates for Service Update 161.

In the Post-Release Developments chapter, added:

- [Support for Multiple NICs for Audio Media Sessions in 1PCC Mode](#).
- [Configuring a Network Interface in Tristate or Line Monitor Modes](#).
- [Support for GSM-FR and GSM-EFR Codecs](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00102159, IPY00102264, IPY00102351, IPY00102398, IPY00102449, IPY00102452, IPY00102512, IPY00102516, IPY00102592, IPY00102771, IPY00115350, IPY00115365, IPY00115371, IPY00115417, IPY00115438, IPY00115534, IPY00115559, IPY00115568, IPY00115655, IPY00115659, IPY00115769.

Document Rev 28 - published September 30, 2013

Updates for Service Update 151.

In the Post-Release Developments chapter, added:

- [Support for 256-bit Master Key Length in Secure RTP](#).
- [Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNxxxxTEPE2HMP Digital Network Interface Boards](#).
- [SIP Session Timer Modifications](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00057301, IPY00094573, IPY00102351, IPY00102398, IPY00102729, IPY00115365, IPY00115515.
- Added the following Known Issues: IPY00102460.

Document Rev 27 - published March 29, 2013

Updates for Service Update 141.

Removed references to the Defect Tracking tool at <http://membersresource.dialogic.com/defects> because it is no longer valid.

In the Post-Release Developments chapter, added:

- [Line Loopback Control on DNI2410AMCTEHMP Board](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00100886, IPY00100927, IPY00101111, IPY00101124, IPY00101133, IPY00101238, IPY00101256, IPY00101259, IPY00101269, IPY00101283, IPY00101289, IPY00101292, IPY00101338, IPY00101441, IPY00101513, IPY00101523, IPY00101595, IPY00101733.

In the Documentation Updates chapter:

- Added the [Dialogic® Conferencing API Programming Guide and Library Reference](#). This document replaces the *Dialogic® Conferencing API Programming Guide* and *Dialogic® Conferencing API Library Reference* on the documentation bookshelf.
- Added an update to the [Dialogic® Global Call IP Technology Guide](#) to address IPY00100886.
- A new version of the [Dialogic® Multimedia API Programming Guide and Library Reference](#) is now available. This version contains information relevant to the wideband audio format. See the document's Revision History section for additional information.

Document Rev 26 - published October 23, 2012

Updates for Service Update 129.

In the Post-Release Developments chapter, added:

- [Updated Operating System Distributions](#).
- [Standalone Licensing Server](#).
- [Enhanced Nb UP Transcode Support](#).
- [MSML Native Hairpinning Support](#).
- [720p Resolution Support](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00099743, IPY00099770, IPY00099784, IPY00099842, IPY00100006, IPY00100146, IPY00100455, IPY00100466, IPY00100752, IPY00100802, IPY00100883, IPY00101082, IPY00101140, IPY00101141, IPY00101185.

In the Documentation Updates chapter:

- Due to the SLS feature added in this release update, the [Dialogic® Standalone License Server User's Guide](#) is added to the documentation bookshelf.
- Removed documentation updates to the [Dialogic® Global Call IP Technology Guide](#) because a new version is available on the bookshelf.
- Due to the Nb UP feature added in this release update, a new version of the [Dialogic® IP Media Library API Programming Guide and Library Reference](#) is now available on the documentation bookshelf.

Document Rev 25 - published June 8, 2012

Updates for Service Update 118.

In the Post-Release Developments chapter, added:

- [MSML <var> Element Support](#).
- [Improved Audio Conferencing Support](#).
- [Echo Cancellation Support for MSML](#).
- [Multiple NIC Support using MSML for RTP](#).
- [Additional Single and Dual DNI Support](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00094390, IPY00094485, IPY00099028, IPY00099177, IPY00099307, IPY00099319, IPY00099434, IPY00099465, IPY00099495, IPY00099563, IPY00099585, IPY00099601, IPY00099602, IPY00099618, IPY00099669, IPY00099736, IPY00099774, IPY00099789, IPY00099801, IPY00099830, IPY00099861, IPY00099906, IPY00099948, IPY00099949, IPY00099975, IPY00100050, IPY00100058, IPY00100242, IPY00100289.
- Added the following Known (permanent) Issues: IPY00100421.

In the Documentation Updates chapter:

- Corrected a capitalization error for *Hmp.Uconfig* in the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#). (IPY00082353)
- Due to the MSML features added in this release update, a new version of the [Dialogic® MSML Media Server Software User's Guide](#) is now available on the documentation bookshelf.

Document Rev 24 - published April 13, 2012

Updates for Service Update 115.

In the Post-Release Developments chapter, added:

- [64-bit SSP Support](#).

In the Release Issues chapter:

- Added the following Resolved Defect: IPY00099594.

Document Rev 23 - published February 7, 2012

Updates for Service Update 108.

In the Post-Release Developments chapter, added:

- [Red Hat Enterprise Linux Release 5 Update 6 Support](#).
- [Support for Party Type in MSML Conferencing](#).
- [Early MSML Connection Identifier Support](#).
- [Loudest Talker Attribute Support for MSML](#).
- [Mute/Un-mute Audio Support for MSML](#).
- [Wildcard Identifiers Support for MSML](#).
- [VMware ESXi 5.0 Support](#).
- [SIP “To tag” for Inbound Calls](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093899, IPY00094465, IPY00094477, IPY00094628, IPY00098916, IPY00098981, IPY00098993, IPY00099007, IPY00099084.

In the Documentation Updates chapter:

- Added a new section to the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#) for Disk IO Performance (IPY00099507).
- Update to the [Dialogic® Conferencing API Programming Guide and Library Reference](#) due to a new feature.
- Removed documentation updates to the [Dialogic® MSML Media Server Software User's Guide](#) because a new version is available on the bookshelf.

Document Rev 22 - published December 16, 2011

Updates for Service Update 103.

In the Post-Release Developments chapter, added:

- [IPv6 Call Control Support](#).

- Added G.722 and AMR-WB to the licensable densities table in [Increased Density for G.711, Low Bit Rate Coders, and High Density Conferencing Placeholder](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056756, IPY00094359, IPY00098980.

In the Documentation Updates chapter:

- Updates to the [Dialogic® Global Call IP Technology Guide](#) to support the IPv6 feature.
- Update to the [Dialogic® IP Media Library API Programming Guide and Library Reference](#) for AMR-WB codec. (IPY00094592)

Document Rev 21 - published October 31, 2011

Updates for Service Update 100.

In the Post-Release Developments chapter, added:

- [Higher Resolution Video Support](#).
- [Support for Dialogic® DSI SS7LD Network Interface Board](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093791, IPY00093859, IPY00093971, IPY00094016, IPY00094163, IPY00094202, IPY00094247, IPY00094356.

Document Rev 20 - published August 24, 2011

Updates for Service Update 94.

In the Post-Release Developments chapter, added:

- [Enhanced Direct 3GP File Play Capabilities](#).
- [Enhanced Video Active Talker Design](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056610, IPY00056632, IPY00056642, IPY00056659, IPY00093101, IPY00093517, IPY00093585, IPY00093633, IPY00093800, IPY00093910, IPY00093944 IPY00093988.

In the Documentation Updates chapter:

- Update to the [Dialogic® Conferencing API Programming Guide and Library Reference](#) due to a feature enhancement. (IPY00093585)
- Removed documentation updates to the [Dialogic® Media Toolkit API Library Reference](#) because a new version is available on the bookshelf.
- Removed documentation updates to the [Dialogic® Multimedia API Programming Guide and Library Reference](#) because a new version is available on the bookshelf.

Document Rev 19 - published June 24, 2011

Updates for Service Update 87.

In the Post-Release Developments chapter, added:

- [CentOS 5 Update 6 Support](#).
- [Increased Density for G.711, Low Bit Rate Coders, and High Density Conferencing Placeholder](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093523, IPY00092358.

In the Documentation Updates chapter:

- Added CentOS 5 Update 6 under the Supported Operating Systems heading in the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#).

Document Rev 18 - published June 3, 2011

Updates for Service Update 86.

In the Post-Release Developments chapter, added:

- [Support for Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP Boards](#).
- [Mute Audio Attribute Support](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093244, IPY00093555.

In the Documentation Updates chapter:

- Removed documentation updates to the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#) because a new version is available on the bookshelf.
- Removed documentation updates to the [Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide](#) because a new version is available on the bookshelf.
- Removed documentation updates to the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#) because a new version is available on the bookshelf.
- Removed documentation updates to the [Dialogic® 3G-324M API Programming Guide and Library Reference](#) because a new version is available on the bookshelf.
- Update to the [Dialogic® Conferencing API Programming Guide and Library Reference](#) due to an added feature.
- Removed documentation updates to the [Dialogic® IP Media Library API Programming Guide and Library Reference](#) because a new version is available on the bookshelf.

Document Rev 17 - published May 25, 2011

Updates for Service Update 85.

In the Post-Release Developments chapter, added:

- [Privilege Talker Attribute Support](#).
- [Support for RFC 3311 UPDATE Message](#).
- [SIP Session Timer](#).
- [Support for Initial Silence Termination](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093283, IPY00093382, IPY00093399, IPY00093409.

In the Documentation Updates chapter:

- Removed Windows information from [Dialogic® Host Media Processing Software for Linux Configuration Guide](#) update.
- Updates to the [Dialogic® Conferencing API Programming Guide and Library Reference](#) due to an added feature.

Document Rev 16 - published April 12, 2011

Updates for Service Update 82.

In the Post-Release Developments chapter, added:

- [PAE Support](#).
- [Using MCX Resources for Audio Conferencing](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056289, IPY00091574, IPY00092840, IPY00092850, IPY00092868, IPY00092944, IPY00092965.
- Added the following Known (permanent) issue: IPY00081246.

In the Documentation Updates chapter:

- Update to the [Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide](#) regarding changes to install script for HPET.
- Added a caution regarding spawning child processes to the “Performance Considerations” section of the [Dialogic® Standard Runtime Library API Programming Guide](#).
- Update to Table 1. in the Feature Support chapter of the [Dialogic® MSML Media Server Software User's Guide](#).

Document Rev 15 - published February 9, 2011

Updates for Service Update 71.

In the Post-Release Developments chapter, added:

- [Support for Multiple NICs.](#)
- [dx_setchxfercnt\(\) API Function Support.](#)
- [H263+ Support.](#)
- [Support for Enhanced RTCP Reports.](#)
- [Red Hat Enterprise Linux Release 5 Update 5 Support.](#)

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00091981, IPY00092510, IPY00092468, IPY00092519, IPY00092529, IPY00092565.

In the Documentation Updates chapter:

- Updated the Basic Software Requirements section of the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#) with support for Red Hat Enterprise Linux Release 5.0 Update 5.
- Added a new section, Configuring the Ephemeral Port Range, to Chapter 10 in the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#).
- In the [Dialogic® Multimedia API Programming Guide and Library Reference](#), updated Tables 1. and 2. in the Features Supported by Platform chapter.

Document Rev 14 - published December 21, 2010

Updates for Service Update 65.

In the Post-Release Developments chapter, added:

- [MSML Server Software Support.](#)
- Update to [IPv6 Support](#). A mixed (both IPv4 and IPv6 addressing) network environment is supported.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00054971, IPY00056010, IPY00056033, IPY00056036, IPY00092357.

In the Documentation Updates chapter:

- Added additional latency parameters to Section 13.4, “[IPVSC] IP Media Parameters” in the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#).
- In the [Dialogic® IP Media Library API Programming Guide and Library Reference](#), updated the [ipm_ModifyMedia\(\)](#) and [ipm_StartMedia\(\)](#) reference pages (IPY00092351).

- Added a new section, [Remote Control Interfaces Documentation](#), for the [Dialogic® MSML Media Server Software User's Guide](#). This guide appears under the “Application Scenario” heading on the documentation bookshelf.

Document Rev 13 - published December 8, 2010

Additional Update for Service Update 61.

In the Post-Release Developments chapter, added:

- Detailed information about [IPv6 Support](#).

Document Rev 12 - published November 16, 2010

Updates for Service Update 61.

In the Post-Release Developments chapter, added:

- [GCC 4.1.1 Compiler Support](#).
- [Playing and Recording Raw E1/T1 DS0 64 Kbps Bit Streams](#).
- [IPv6 Support](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00091419, IPY00091502, IPY00091783, IPY00091959, IPY00091982, IPY00092200, IPY00092209, IPY00092243, IPY00092278, IPY00092280, IPY00092285, IPY00092289 (resolved in Service Update 53).

In the Documentation Updates chapter:

- Removed documentation updates to the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#) because a new version is available on the bookshelf. This guide was previously named the Dialogic® System Configuration Guide.

Document Rev 11 - published October 12, 2010

Additional updates for Service Update 56.

In the Post-Release Developments chapter, added:

- [Support for Dialogic® DNI/300TEPHMPW, DNI/601TEPHMPW, and DNI/1200TEPHMPW Boards](#).
- [Increase in Channels Density for Conferencing and CSP](#).
- [Recording and Playback of G.729A Files](#).

Document Rev 10 - published September 23, 2010

Updates for Service Update 56.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055853, IPY00081808, IPY00091792, IPY00091910, IPY00091918.

In the Documentation Updates chapter:

- Added updated information about HPET disable to Steps 4.a. and 4.c. in the “Testing the Required Operation of the Linux RTC Device” in the [Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide](#).

Document Rev 09 - published August 31, 2010

Updates for Service Update 53.

In the Post-Release Developments chapter, added:

- [Virtualization Support](#).
- [Increased Channel Support for Conferencing](#).
- [H.264 Transcoding Support for 3G-324M Devices](#).
- Updates to the [Support for HD Voice Conferencing \(Wideband Audio Conferencing\)](#) section: changed text in the Licensing Requirements section and added a new section, Conferencing Licenses Exceeded Event.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00091419, IPY00091478.

In the Documentation Updates chapter:

- Because of a new feature ([Support for HD Voice Conferencing \(Wideband Audio Conferencing\)](#)), updated the [Dialogic® Conferencing API Programming Guide and Library Reference](#).

Document Rev 08 - published August 12, 2010

Updates for Service Update 49.

In the Post-Release Developments chapter, added:

- [Support for HD Voice Conferencing \(Wideband Audio Conferencing\)](#).

In the Release Issues chapter:

- Added the following Resolved Defect: IPY00091445.

Document Rev 07 - published July 27, 2010

Updates for Service Update 46.

In the Post-Release Developments chapter, added:

- [Support for Additive Licensing.](#)
- [3G-324M H.264 Native Support.](#)
- [Unspecified G.723.1 Bit Rate in Outgoing SIP Requests with SDP.](#)
- [H.264 Transcoding Support.](#)(Available in SU 42)

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055506, IPY00055625, IPY00079689, IPY00080124, IPY00080125, IPY00090932, IPY00091091, IPY00091132, IPY00091162, IPY00091292, IPY00091348, IPY00091361, IPY00091383, IPY00091422, IPY00091492, IPY00091495, IPY00091529, IPY00091536.

In the Documentation Updates chapter:

- Removed a note under Supported Operating Systems heading in the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#).
- Because of a new feature (Support for Additive Licensing), made changes to Chapter 5. "License Administration" in the [Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide](#).
- Because of a new feature (Support for Additive Licensing), made changes to Chapter 6. "Configuration Procedures Using CLI" in the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#).
- Because of a new feature (3G-324M H.264 Native Support), made several updates in the [Dialogic® 3G-324M API Programming Guide and Library Reference](#).
- Because of a new feature (H.264 Transcoding Support), made several updates in the [Dialogic® IP Media Library API Programming Guide and Library Reference](#).
- Because of a new feature (H.264 Transcoding Support), made several updates in the [Dialogic® Multimedia API Programming Guide and Library Reference](#).

Document Rev 06 - published July 8, 2010

Updates for Service Update 42.

In the Post-Release Developments chapter, added:

- [Support for Dialogic® DNI2410AMCTEHMP AdvancedMC Module](#).
- [32-bit Compatibility Mode on 64-bit Linux Systems](#).
- [3G-324 Gateway + Media Server for IVVR](#).
- [Overlap-Receive Support for Limited SIP-I Interworking Scenarios](#).
- [Processing Multiple 18x Provisional Responses](#).
- [TLS and SRTP Channel Support Increase](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055507, IPY00081685, IPY00082125, IPY00090932, IPY00090994, IPY00091023, IPY00091093, IPY00091138, IPY00091139, IPY00091140, IPY00091439, IPY00091441, IPY00091475.

Document Rev 05 - published May 27, 2010

Updates for Service Update 37.

Note: If you are using Red Hat Enterprise Linux Release 5.x, you need the backward compatibility RPM package **compat-libstdc++-xxxx**, or something similar, containing libstdc++.so.5 installed on your system. For Build 37, we have changed the installation script to load the 3.4.3 RPM packages. This temporary restriction will be removed in a future release.

In the Post-Release Developments chapter, added:

- [33 Frames Per Packet Support \(AMR\)](#).
- [G.722.2 Adaptive Multi-Rate Wideband Codec \(AMR-WB\) Support](#).
- [Registering Authentication Data without Realm String](#).
- [Handling non-2xx Responses to T.38 Switch](#).
- [MIME Insertion in Outgoing ACK](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055439, IPY00081656, IPY00081664, IPY00081732, IPY00082301, IPY00090645, IPY00090646, IPY00090750, IPY00090953, IPY00091076.

In the Documentation Updates chapter:

- Added Red Hat Enterprise Linux Release 5.0 with Service Update 4 under the Supported Operating Systems heading in the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#).
- Made updates to Sections 3.9 and 5.1 in the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#) as a result of resolving IPY00081664.
- Added updates to the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#) due to an increase in frames per packet for AMR.
- Modifications were made to several sections of the [Dialogic® Multimedia API Programming Guide and Library Reference](#) as a result of resolving IPY00081664.

Document Rev 04 - published April 19, 2010

Updates for Service Update 32.

Note: If you are using Red Hat Enterprise Linux Release 5.x, you need the backward compatibility RPM package **compat-libstdc++-xxxx**, or something similar, containing libstdc++.so.5 installed on your system. This is required because not all the Dialogic libraries are fully linked to the GCC 4.1.1 libraries. Once the linking process is completed in a future service update, this requirement will be lifted.

In the Post-Release Developments chapter, added:

- [G.722 Wideband Codec Support](#).
- [Increase in Channel Density for G.711 Codec](#).
- [Monitor Mode Support for HMP Conferencing](#).
- [Increase in TLS and SRTP Channels](#).
- [IPM\(H.263+\) to IPM\(H.263\) Connection Support](#).
- [3PCC Support for Dynamic Selection of Outbound SIP Proxy](#).
- Added updates to content as a result of adding 3PCC support in [Support for Dynamic Selection of Outbound SIP Proxy](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055290, IPY00080472, IPY00081135, IPY00081136, IPY00081235, IPY00081664, IPY00081676, IPY00081807, IPY00082081, IPY00082088, IPY00082137, IPY00082163, IPY00082165, IPY00082294, IPY00082344, IPY00090656.

In the Documentation Updates chapter:

- Updated the [Dialogic® IP Media Library API Programming Guide and Library Reference](#) with new information to support the IPM(H.263+) to IPM(H.263) Connection feature.

Document Rev 03 - published February 16, 2010

Updates for Service Update 25.

In the Post-Release Developments chapter:

- Added [Native H.264 Support](#).
- Added [32-bit Compatibility Mode](#) (available in Service Update 23).
- Added [Support for Four Octal-Span Boards](#) (available in Service Update 23).

In the Release Issues chapter:

- Added the following Resolved Defect: 81807.

In the Documentation Updates chapter:

- A new version of the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#) is now available on the documentation bookshelf. See the Revision History for details about updates.
- Corrected information about protocol groups in the [Dialogic® Host Media Processing Software for Linux Configuration Guide](#). (IPY00082247)
- Updated the [Dialogic® IP Media Library API Programming Guide and Library Reference](#) for native H.264 support.
- Updated the [Dialogic® Multimedia API Programming Guide and Library Reference](#) for native H.264 support.

Document Rev 02 - published January 15, 2010

Updates for Service Update 23.

In the Post-Release Developments chapter:

- Included general information about this [Service Update](#).
- Added [Support for HMP 3.1LIN Features](#).
- Added [Support for WaitCall Cancellation](#).
- Added the ability to [Defer the Sending of SIP Messages](#).
- Added [Support for Dynamic Selection of Outbound SIP Proxy](#).
- Added [Retrieving SIP Inbound RFC 2833](#).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00054750, IPY00054969, IPY00054970, IPY00055020, IPY00080010, IPY00080011, IPY00080141, IPY00080308, IPY00080339, IPY00080468, IPY00080693, IPY00080694, IPY00080772, IPY00080800, IPY00081132, IPY00081142, IPY00081264, IPY00081277, IPY00081284, IPY00081399, IPY00081489, IPY00081571, IPY00081661, IPY00081665, IPY00081756, IPY00081826, IPY00081880.

In the Documentation Updates chapter:

- A new version of the [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#) is now available on the documentation bookshelf.
- Added the [Dialogic® Continuous Speech Processing API Library Reference](#) and the [Dialogic® Continuous Speech Processing API Programming Guide](#) to the documentation bookshelf.

Document Rev 01 - published October 16, 2009

Initial version of document for Dialogic® PowerMedia™ HMP for Linux Release 4.1.

Note: Global Call H.323 protocol support for Call Control has been deprecated. Functionality remains in this release but has not been tested nor will defects be addressed on this technology. Contact your Dialogic sales representative if this technology is required for your application.

Post-Release Developments

1

This section describes significant changes to Dialogic® PowerMedia™ HMP for Linux Release 4.1 subsequent to the general availability release.

• Service Update	26
• MSML Media Server Software Deprecation	26
• Installation Package Policy	27
• Updated Operating System Support.	27
• VMware ESXi 6.x Support	27
• DTLS-SRTP Support	27
• STUN Message Support.	31
• Opus Audio Codec Support	34
• iLBC Audio Codec Support.	35
• VP8 Video Codec Support	36
• RTCP Feedback Support	36
• VGA 480x640 Aspect Ratio H.264 Video Support	37
• 3GP Multimedia Container for Play and Record Support	37
• G.729 Codec with 60ms Packet Support	38
• AMR2 Audio Codec Support.	38
• Send/Receive RFC 2833/RFC 4733 Tone Events Support	41
• Expanded Interface Identification Support	50
• Red Hat Enterprise Linux Release 7 Update 1 Support.	51
• Oracle Enterprise Linux 6.2 64-bit Support	51
• Operating System Distributions No Longer Supported.	51
• Support for Multiple NICs for Audio Media Sessions in 1PCC Mode	52
• Configuring a Network Interface in Tristate or Line Monitor Modes	57
• Support for GSM-FR and GSM-EFR Codecs.	59
• Support for 256-bit Master Key Length in Secure RTP.	59
• Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNxxxxTEPE2HMP Digital Network Interface Boards	61
• SIP Session Timer Modifications	85
• Line Loopback Control on DNI2410AMCTEHMP Board.	85
• Updated Operating System Distributions	89

• Standalone Licensing Server	89
• Enhanced Nb UP Transcode Support	89
• MSML Native Hairpinning Support	90
• 720p Resolution Support	93
• MSML <var> Element Support	93
• Improved Audio Conferencing Support	93
• Echo Cancellation Support for MSML	94
• Multiple NIC Support using MSML for RTP	95
• Additional Single and Dual DNI Support	95
• 64-bit SSP Support	96
• Red Hat Enterprise Linux Release 5 Update 6 Support	96
• Support for Party Type in MSML Conferencing	96
• Early MSML Connection Identifier Support	97
• Loudest Talker Attribute Support for MSML	98
• Mute/Un-mute Audio Support for MSML	100
• Wildcard Identifiers Support for MSML	100
• VMware ESXi 5.0 Support	101
• SIP “To tag” for Inbound Calls	101
• IPv6 Call Control Support	104
• Higher Resolution Video Support	105
• Support for Dialogic® DSI SS7LD Network Interface Board	110
• Enhanced Direct 3GP File Play Capabilities	111
• Enhanced Video Active Talker Design	112
• CentOS 5 Update 6 Support	113
• Increased Density for G.711, Low Bit Rate Coders, and High Density Conferencing Placeholder	113
• Support for Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP Boards	115
• Mute Audio Attribute Support	122
• Privilege Talker Attribute Support	123
• Support for RFC 3311 UPDATE Message	123
• SIP Session Timer	131
• Support for Initial Silence Termination	137
• PAE Support	138
• Using MCX Resources for Audio Conferencing	138

• Support for Multiple NICs	138
• dx_setchxfercnt() API Function Support	144
• H263+ Support	144
• Support for Enhanced RTCP Reports	145
• Red Hat Enterprise Linux Release 5 Update 5 Support	145
• MSML Server Software Support	145
• GCC 4.1.1 Compiler Support	145
• Playing and Recording Raw E1/T1 DS0 64 Kbps Bit Streams	146
• IPv6 Support	155
• Support for Dialogic® DNI/300TEPHMPW, DNI/601TEPHMPW, and DNI/1200TEPHMPW Boards	169
• Increase in Channels Density for Conferencing and CSP	170
• Recording and Playback of G.729A Files	170
• Virtualization Support	170
• Increased Channel Support for Conferencing	173
• H.264 Transcoding Support for 3G-324M Devices	174
• Support for HD Voice Conferencing (Wideband Audio Conferencing)	174
• Support for Additive Licensing	174
• 3G-324M H.264 Native Support	177
• H.264 Transcoding Support	184
• Support for Dialogic® DNI2410AMCTEHMP AdvancedMC Module	184
• 32-bit Compatibility Mode on 64-bit Linux Systems	185
• 3G-324 Gateway + Media Server for IVVR	185
• Overlap-Receive Support for Limited SIP-I Interworking Scenarios	185
• Processing Multiple 18x Provisional Responses	191
• TLS and SRTP Channel Support Increase	194
• 33 Frames Per Packet Support (AMR)	194
• G.722.2 Adaptive Multi-Rate Wideband Codec (AMR-WB) Support	194
• Registering Authentication Data without Realm String	195
• Handling non-2xx Responses to T.38 Switch	195
• MIME Insertion in Outgoing ACK	200
• G.722 Wideband Codec Support	208
• Increase in Channel Density for G.711 Codec	212
• Monitor Mode Support for HMP Conferencing	212

• Increase in TLS and SRTP Channels	213
• IPM(H.263+) to IPM(H.263) Connection Support	213
• 3PCC Support for Dynamic Selection of Outbound SIP Proxy	213
• Native H.264 Support	214
• 32-bit Compatibility Mode	214
• Support for Four Octal-Span Boards	214
• Support for HMP 3.1LIN Features	214
• Support for WaitCall Cancellation	214
• Defer the Sending of SIP Messages	217
• Support for Dynamic Selection of Outbound SIP Proxy	221
• Retrieving SIP Inbound RFC 2833	226

1.1 Service Update

This Service Update for Dialogic® PowerMedia™ HMP for Linux Release 4.1 is now available. Service Updates provide fixes to known problems, and may also introduce new functionality. It is intended that new versions of the Service Update will be released periodically.

For information about installing this Service Update, refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide*.

Note: For Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 adds support for new features, APIs, and capabilities. It is recommended to recompile and validate applications against Service Update 213 because there may be changes in memory and CPU usage.

1.2 MSML Media Server Software Deprecation

As of Service Update 236 for Dialogic® PowerMedia™ HMP for Linux Release 4.1, support for the MSML Media Server Software has been deprecated. It is recommended to migrate to Dialogic® PowerMedia® XMS (<http://www.dialogic.com/xms>) which supports the MSML interface.

For details on the differences between Legacy MSML implementation in HMP versus Native MSML implementation in XMS, refer to the *Dialogic® PowerMedia™ XMS MSML Media Server Software User's Guide* (http://www.dialogic.com/webhelp/XMS/3.5/XMS_MSMLUser.pdf).

1.3 Installation Package Policy

Dialogic® PowerMedia™ HMP for Linux Release 4.1 is an RPM-based installation packaged delivered as a g-zipped tar (.tgz) for installing HMP on an existing Linux system. It is recommended that users apply required updates in line with their applicable security policy/policies and to ensure that the updates are tested on a non-production HMP server prior to deployment. It is also recommended that a system backup and rollback procedure be put into place prior to deployment, in the event that any issues arise as a result of any updates being applied in production servers. Any issue(s) affecting the operation of HMP due to a security update should be reported to Dialogic.

1.4 Updated Operating System Support

With Service Update 236, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports the following Red Hat Enterprise Linux (RHEL) and Community ENTERprise Operating System (CentOS) versions in both 32-bit and 64-bit:

- Red Hat 7.1 or greater
- CentOS 7.1 or greater

1.5 VMware ESXi 6.x Support

VMware ESXi 6.x is supported by Dialogic® PowerMedia™ HMP for Linux Release 4.1. Refer to the [Virtualization Support](#) section for HMP configuration and other relevant information. For VMware documentation, refer to the <http://www.vmware.com/support/pubs>.

1.6 DTLS-SRTP Support

RFC 5764 describes an extension to DTLS (RFC 4347) known as DTLS-SRTP. DTLS-SRTP is used to establish symmetric keys for an SRTP session. All key information is exchanged in the media path. This is different than other mechanisms such as SDES-SRTP where the symmetric SRTP key information is exchanged through SDP. The IPM device performs the DTLS handshake with the remote endpoint. Once the DTLS handshake is complete, the IPM device is able to transmit SRTP packets and process incoming SRTP packets.

1.6.1 Configuring an IPM Device for a DTLS-SRTP Session

To configure an IPM Device for a DTLS-SRTP session, proceed as follows:

1. Retrieve the fingerprint of the X.509 certificate that the IPM device will send to the remote endpoint during the DTLS handshake. The fingerprint is retrieved from the IPM device by calling **ipm_GetFingerprint()** and processing the resulting IPMEV_GET_FINGERPRINT event.
2. Exchange information about the DTLS session to be created with the remote endpoint (e.g., through SDP). This includes certificate fingerprints and negotiating DTLS client and DTLS server. See RFC 5763 for more information.
3. In the IPM_MEDIA structure, fill the IPM_DTLS_SRTP_INFO substructure. This includes the remote endpoint's certificate fingerprint and whether the IPM device will be the DTLS client or DTLS server.
4. In the IPM_MEDIA structure, set eMediaType to MEDIATYPE_DTLS_SRTP_INFO (session level) or MEDIATYPE_DTLS_SRTP_AUDIO_INFO / MEDIATYPE_DTLS_SRTP_VIDEO_INFO (media level). Two IPM_MEDIA structures are required if audio and video are configured separately.
5. Pass the IPM_MEDIA structure(s) to **ipm_StartMedia()**. Set the direction parameter to bi-directional. The IPM device will perform the DTLS handshake.

1.6.2 ipm_GetFingerprint()

Name:	int ipm_GetFingerprint(nDeviceHandle, *pInfo, usMode)	
Inputs:	int nDeviceHandle	• IP Media device handle
	IPM_FINGERPRINT_INFO *pInfo	• pointer to fingerprint information structure
	unsigned short usMode	• async or sync mode setting
Returns:	0 if success -1 if failure	
Includes:	srllib.h ipmlib.h	
Category:	Media Session	
Mode:	asynchronous or synchronous	

■ Description

The **ipm_GetFingerprint()** function retrieves the fingerprint of the X.509 certificate that will be transmitted to the remote endpoint during the DTLS handshake. The caller must send the fingerprint to the remote endpoint (e.g., using SDP as defined in RFC 5763). The **INIT_IPM_FINGERPRINT_INFO()** function must be called to initialize the **IPM_FINGERPRINT_INFO** structure. **ipm_GetFingerprint()** must be called before each DTLS-SRTP session is started since the fingerprint can be different for each session.

■ Termination Events

IPMEV_GET_FINGERPRINT

Indicates successful completion, that is, the fingerprint information was received. Once the event has been returned, use Dialogic® SRL functions to retrieve **IPM_FINGERPRINT_INFO** structure fields.

IPMEV_GET_FINGERPRINT_FAIL

Indicates that the function failed.

■ Errors

If the function returns -1 to indicate failure, call **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to return one of the following errors:

EIPM_BADPARAM

Invalid parameter.

EIPM_INTERNAL

Internal error.

EIPM_INV_MODE

Invalid mode.

EIPM_INV_STATE

Invalid state. Initial command did not complete before another function call was made.

EIPM_SYSTEM

System error.

■ Example

```
int GetFingerprint()
{
    IPM_FINGERPRINT_INFO info;

    INIT_IPM_FINGERPRINT_INFO(&info);
    info.eHashType = IPM_HASH_TYPE_SHA_256;

    if( ipm_GetFingerprint(handle, &info, EV_ASYNC) != 0 )
    {
        return -1;
    }
    return 0;
}
```

```

void OnGetFingerprint()
{
    IPM_FINGERPRINT_INFO* pInfo = (IPM_FINGERPRINT_INFO *)sr_getevtdatap();

    if( sr_getevtlen() == (int)(sizeof(IPM_FINGERPRINT_INFO)) )
    {
        printf("fingerprint=%s\n", pInfo->pcFingerprint);
    }
    else
    {
        printf("Invalid size %ld expected %d\n", sr_getevtlen(), sizeof(IPM_FINGERPRINT_INFO));
    }
}

```

1.6.3 IPM_FINGERPRINT_INFO

```

typedef struct ipm_fingerprint_info_tag
{
    unsigned int    unVersion;
    eIPM_HASH_TYPE  eHashType;
    char            pcFingerprint[MAX_IPM_HASH_SIZE];
} IPM_FINGERPRINT_INFO, *PIPM_FINGERPRINT_INFO;

```

■ Description

This structure contains the information required to generate the fingerprint of the X.509 certificate that the IPM device will send to the remote endpoint during the DTLS handshake.

■ Field Descriptions

unVersion
Version number of the data structure.

HashType
Hash function to use when generating the fingerprint.

IPM_HASH_TYPE_SHA_256
Use the SHA-256 hash function to generate the fingerprint.

pcFingerprint
Fingerprint string returned by IPM device.

1.6.4 IPM_DTLS_SRTP_INFO

```

typedef struct ipm_dtls_srtp_tag {
    unsigned int    unVersion;
    eIPM_DTLS_SRTP_ROLE unLocalRole;
    unsigned char    *pRemoteFingerprint;
} IPM_DTLS_SRTP_INFO, *PIPM_DTLS_SRTP_INFO;

```

■ Description

This structure contains the information required to configure a DTLS-SRTP session. **INIT_IPM_DTLS_SRTP_INFO()** must be called to initialize the structure fields.

■ Field Descriptions

unVersion

Version number of the data structure. Used to ensure that an application is binary compatible with future changes to this data structure.

unLocalRole

- DTLS_SRTP_ROLE_CLIENT - The IPM device initiates the DTLS handshake, i.e. the IPM device sends the DTLS client hello messages.
- DTLS_SRTP_ROLE_SERVER - The remote endpoint initiates the DTLS handshake, i.e. the IPM device waits for the DTLS client hello messages.

pRemoteFingerprint

Set this pointer to the certificate fingerprint string received from the remote endpoint.

Typecast the string to an "unsigned char*". For example, "sha-256

FA:A8:B8:3E:F9:5E:F9:44:4B:C8:A0:5E:3C:5F:1B:9F:34:A8:2B:29:59:E7:AC:17:52:C
A:D8:F8:98:01:04:6E". Only "sha-256" is available.

1.7 STUN Message Support

The IPM device has been updated to allow an application to transmit STUN messages to a remote endpoint and receive STUN messages from a remote endpoint. The STUN message payload is provided by the application. STUN messages are defined in RFC 5389. This feature can be used to implement protocols such as ICE (RFC 5245).

1.7.1 Handling STUN Messages Received from a Remote Endpoint

Proceed as follows to handle STUN messages received from a remote endpoint:

1. Pass EVT_STUN_MESSAGE_RECEIVED to **ipm_EnableEvents()** to enable the IPMEV_STUN_MESSAGE_RECEIVED event.
2. The application will receive an IPMEV_STUN_MESSAGE_RECEIVED event for each STUN message received from the remote endpoint. The event contains the IPM_STUN_MESSAGE_INFO structure which includes the source and destination IPv4 or IPv6 address of the STUN message and the STUN message payload.
3. Pass EVT_STUN_MESSAGE_RECEIVED to **ipm_DisableEvents()** to disable the IPMEV_STUN_MESSAGE_RECEIVED event when STUN events are no longer required.

1.7.2 Transmitting STUN Messages to a Remote Endpoint

Proceed as follows to transmit STUN messages to a remote endpoint:

1. To enable STUN message transmission, set the remote IP address to "STUN" and the port ID to 0 in the REMOTE_RTP media infos. Refer to the following example.

```
MediaInfo.MediaData[index].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTP_INFO;
strcpy(MediaInfo.MediaData[index].mediaInfo.PortInfo.cIPAddress, "STUN");
MediaInfo.MediaData[index].mediaInfo.PortInfo.unPortId = 0;
index++;
MediaInfo.MediaData[index].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTCP_INFO;
strcpy(MediaInfo.MediaData[index].mediaInfo.PortInfo.cIPAddress, "STUN");
MediaInfo.MediaData[index].mediaInfo.PortInfo.unPortId = 0;
index++;
```

2. Call **ipm_StartMedia()** to start the media session in "STUN mode".
3. Set the fields in the IPM_STUN_MESSAGE_INFO structure. This includes source and destination IPv4 or IPv6 addresses and a user defined STUN payload.
4. Call **ipm_SendSTUNMessageEx()** to transmit the STUN message.

Note: STUN mode only applies to **ipm_StartMedia()**. **ipm_ModifyMedia()** will return an error if the address "STUN" is given.

1.7.3 ipm_SendSTUNMessageEx()

Name:	int ipm_SendSTUNMessageEx(nDeviceHandle, *pStunMessage)	
Inputs:	nDeviceHandle	• IP Media device handle
	IPM_STUN_MESSAGE_INFO* pStunMessage	• pointer to STUN information structure
Returns:	0 if success -1 if failure	
Includes:	srllib.h ipmlib.h	
Category:	Media Session	
Mode:	asynchronous	

■ Description

The **ipm_SendSTUNMessageEx()** function sends an RFC 5389 STUN message defined by the IPM_STUN_MESSAGE_INFO structure to a remote endpoint. The caller must create the STUN message payload.

The pMessage buffer is copied by **ipm_SendSTUNMessageEx()**, so the memory can be freed after calling this function.

Parameter	Description
nDeviceHandle	Handle of the IP Media device
pStunMessage	STUN message data structure; see IPM_STUN_MESSAGE_INFO for details.

■ Errors

If the function returns -1 to indicate failure, call **ATDV_LASTERR()** and **ATDV_ERRMSGP()** to return one of the following errors:

EIPM_BADPARAM
Invalid parameter.

EIPM_INTERNAL
Internal error.

EIPM_INV_MODE
Invalid mode.

EIPM_INV_STATE
Invalid state. Initial command did not complete before another function call was made.

EIPM_SYSTEM
System error.

1.7.4 IPM_STUN_MESSAGE_INFO

```
typedef struct IPM_STUN_MESSAGE_INFO_tag
{
    unsigned int      unVersion;
    IPM_PORT_INFO     PortInfoSrc;
    IPM_PORT_INFO     PortInfoDst;
    unsigned char*    pMessage;
    unsigned int      unMessageSize;
    IPM_PORT_INFO_V6  PortInfoSrc_V6;
    IPM_PORT_INFO_V6  PortInfoDst_V6;
    IPM_IP_ADDR_FAMILY IpAddrFamily;
} IPM_STUN_MESSAGE_INFO;
```

■ Description

This structure describes a STUN message to transmit or a STUN message that has been received. The inline function **INIT_IPM_STUN_MESSAGE_INFO()** must be called to initialize this structure.

■ Field Descriptions

unVersion
Version number of the data structure. Used to ensure that an application is binary compatible with future changes to this data structure.

IpAddrFamily

Address family. Set to IP_ADDR_FAMILY_IPV4 to use the IPv4 structures PortInfoSrc and PortInfoDst. Set to IP_ADDR_FAMILY_IPV6 to use the IPv6 structures PortInfoSrc_V6 and PortInfoDst_V6.

PortInfoSrc

Pointer to IPM_PORT_INFO structure for IPv4. The STUN message contained in pMessage was received from the endpoint identified by the IP address and UDP port in PortInfoSrc.

PortInfoDst

Pointer to IPM_PORT_INFO structure for IPv4. Contains the local IP address and local UDP port of the IPM device.

PortInfoSrc_V6

Pointer to IPM_PORT_INFO_V6 structure for IPv6. The STUN message contained in pMessage was received from the endpoint identified by the IP address and UDP port in PortInfoSrc.

PortInfoDst_V6

Pointer to IPM_PORT_INFO_V6 structure for IPv6. Contains the local IP address and local UDP port of the IPM device.

pMessage

Pointer to the STUN message to transmit or the STUN message received.

unMessageSize

Number of bytes in the STUN message payload pointed to by pMessage.

■ Caution

The caller must free pMessage by calling **free()** for received STUN messages.

1.8 Opus Audio Codec Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports the Opus codec as one of the supported audio codec options for media streaming or audio transcoding.

Note: The Opus codec is an open source, royalty free interactive speech audio codec provided by Google and designed to handle a wide range of interactive audio.

The Opus codec is composed of a layer based on Linear Prediction Coding (LPC) and a layer based on the Modified Discrete Cosine Transform (MDCT) that enables it to operate over a wider bandwidth range. It scales from low bitrate narrowband speech at 6 kbit/s to very high quality stereo music at 510 kbit/s. The Opus codec uses both Linear Prediction Coding (LPC) and the Modified Discrete Cosine Transform (MDCT) to achieve good compression of both speech and music.

The specification for the Opus codec can be found in RFC 6717. The Opus codec is currently supported by the WebRTC implementations of Firefox and Chrome browsers. Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports streaming Opus to SIP or WebRTC endpoints.

In Dialogic® PowerMedia™ HMP for Linux Release 4.1, the Opus implementation supports 20ms frames up to 48 kbit/s.

The **ipm_StartMedia()** function is used to configure an IPM device to transmit and receive Opus RTP streams. Set the eCoderType field of the IPM_AUDIO_CODER_INFO structure to CODER_TYPE_OPUS.

Refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for more information about **ipm_StartMedia()** and related structures.

Note: Depending on your OS distribution, the Opus codec RPM may not be installed by default. Dialogic® PowerMedia™ HMP for Linux Release 4.1 will start without the package installed; however, the codec will not be available when calling **ipm_StartMedia()**. Install the Opus codec package opus-1.0.2-6 or later to enable Opus support in PowerMedia HMP.

1.9 iLBC Audio Codec Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports the Internet Low Bitrate Codec (iLBC) as one of the supported audio codec options for media streaming or audio transcoding at 13.33 kbps and 15.2 kbps.

Note: The iLBC codec was originally developed by Global IP Solutions to support voice audio over unreliable IP connections. The iLBC codec has since been acquired by Google and now licensed as part of the WebRTC project (<http://www.webrtc.org/ilbc-freeware>).

The iLBC codec is a high-complexity speech codec that is suitable for robust voice communication over IP. The iLBC codec enables graceful speech quality degradation in the case of lost frames, which occurs in connection with lost or delayed IP packets. The specification for the iLBC codec can be found in RFC 3951. Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports streaming iLBC to SIP or WebRTC endpoints.

The **ipm_StartMedia()** function is used to configure an IPM device to transmit and receive iLBC RTP streams. Set the eCoderType field of the IPM_AUDIO_CODER_INFO structure to CODER_TYPE_ILBC_13_33K or CODER_TYPE_ILBC_15_2.

Refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for more information about **ipm_StartMedia()** and related structures.

Note: Depending on your OS distribution, the iLBC codec RPM may not be installed by default. Dialogic® PowerMedia™ HMP for Linux Release 4.1 will start without the package installed; however, the codec will not be available when calling **ipm_StartMedia()**. Install the iLBC codec package ilbc-1.1.1-4 or later to enable iLBC support in PowerMedia HMP.

1.10 VP8 Video Codec Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports the VP8 video codec as one of the supported video codec options for media streaming or video transcoding up to HD720p. VP8 is an open source, royalty free codec provided by Google for high quality video at varying bit rates. It uses the same concept of most modern video codecs (macroblocks, I-frame, P-frame) in addition to some unique enhancements that optimize the decode/encode process. VP8 is supported by the WebRTC implementations of Firefox and Chrome browsers. Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports streaming VP8 to SIP or WebRTC endpoints.

The **ipm_StartMedia()** function is used to configure an IPM device to transmit and receive VP8 RTP streams. Set the eCoderType field of the IPM_VIDEO_CODER_INFO structure to CODER_TYPE_VP8 and set up the IPM_VIDEO_CODER_INFO_EX structure.

When configuring MEDIATYPE_VIDEO_LOCAL_CODER_INFO, set the the elmageWidth and elmageHeight fields of the IPM_VIDEO_CODER_INFO_EX structure to VIDEO_IMAGE_WIDTH_UNKNOWN and VIDEO_IMAGE_HEIGHT_UNKNOWN.

Refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for more information about **ipm_StartMedia()** and related structures.

1.11 RTCP Feedback Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports several RTCP feedback mechanisms. The RTCP feedback mechanisms are used to improve video quality and adjust network bandwidth usage.

The following RTCP feedback mechanisms are supported. The RFC that defines each RTCP mechanism is given.

Supported RTCP Feedback Mechanisms	RFC
Generic NACK	RFC 4585
PLI	RFC 4585
FIR	RFC 5104
TMMBR	RFC 5104
REMB	draft-alvestrand-rmcat-remb-03

The IPM_VIDEO_RTCP_FEEDBACK_INFO structure has been added to the IPM_MEDIA structure. The IPM_VIDEO_RTCP_FEEDBACK_INFO structure is used to enable transmission of specific RTCP feedback messages. Incoming RTCP feedback messages are always processed independent of the transmit configuration.

The application negotiates the RTCP feedback types supported with the remote endpoint (e.g., using the mechanisms defined in RFC 4585 and RFC 5104).

■ Example

The following example shows how to enable transmission of Generic NACK, PLI, FIR, and REMB RTCP feedback messages.

```
void StartMedia()
{
    int index = 0;
    IPM_MEDIA_INFO MediaInfo;
    memset(&MediaInfo, 0, sizeof(IPM_MEDIA_INFO));

    ...

    MediaInfo.MediaData[index].eMediaType = MEDIATYPE_VIDEO_RTCP_FEEDBACK_INFO;
    INIT_IPM_VIDEO_RTCP_FEEDBACK_INFO(&MediaInfo.MediaData[index].mediaInfo.RtcpFeedbackInfo);
    MediaInfo.MediaData[index].mediaInfo.RtcpFeedbackInfo.unConfigBits1 = IPM_RTCP_FB_NACK |
        IPM_RTCP_FB_PLI |
        IPM_RTCP_FB_FIR |
        IPM_RTCP_FB_REMB; // alternatively use IPM_RTCP_FB_TMMBR to enable TMMBR

    index++;
    MediaInfo.unCount = index;

    if( ipm_StartMedia(handle, &MediaInfo, DATA_IP_TDM_BIDIRECTIONAL, EV_ASYNC) == -1 )
    {
        printf("ipm_StartMedia() failed, %s (%ld)\n", ATDV_ERRMSGP(handle), ATDV_LASTERR(handle));
    }
}
```

1.12 VGA 480x640 Aspect Ratio H.264 Video Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 adds support for VGA 480x640 aspect ratio H.264 video to the media engine.

1.13 3GP Multimedia Container for Play and Record Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports direct 3GPP file format (3GP) for both play and record operations. The 3GP file container, defined by the third generation partnership project (3GPP), is a standard format multimedia container for IMS based cellular telephony and VoIP networks as defined by 3GPP TS 26.44. The 3GP container is specified for a subset of container characteristics, including restricted audio and video codecs defined for these network use cases. 3GP is an ISO-based multimedia file format and a subset of the MP4 file container.

Dialogic® PowerMedia™ HMP for Linux Release 4.1 has been updated to support 3GP record and playback through remote API interfaces that support video. Some of the highlighted functionality provided for 3GP container includes:

- Support for play and record directly to/from .3gp
- Support for audio only, video only, and multimedia (A/V) files
- Supported video codecs: H.264 (up to HD720p resolution at 2Mbps)
- Supported audio codecs: AMR-NB and AMR-WB
- Supported DVR modes: skip forward, skip back, pause, resume (hint track required)
- Support for record with hint track to allow DVR modes on playback

Note: H.263 video codec is supported for play only; MPEG4 video codec is not supported in the initial release of this feature.

Note: AAC codec is not supported in 3GP container.

For 3GP file play and record, the Dialogic® Multimedia API library provides the EMM_FILE_FORMAT_3GP value to indicate that the file is in 3GP format. Refer to the MM_MEDIA_VIDEO and MM_MEDIA_AUDIO data structures for file formats.

When the szFileName in the MM_MEDIA_VIDEO and MM_MEDIA_AUDIO structures are set to the same filename, the MM device will play the tracks present in the 3GP file. For example, if the 3GP file contains an audio track and no video track, the audio track will be played. If the file contains an audio and a video track, both tracks will be played.

Refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for more information about **mm_Play()**, **mm_Record()**, **mm_Seek()**, **mm_Pause()**, **mm_Resume()**, and related structures.

1.14 G.729 Codec with 60ms Packet Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 adds support for G.729 codec with 60ms packets.

1.15 AMR2 Audio Codec Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports the AMR2 codec. The AMR2 codec is a restricted subset of AMR codec functionality provided for VoLTE compatibility with older UMTS networks. The use of AMR2 promotes Tandem Free Operation (TFO) and Transcoder Free Operation (TrFO) when a legacy network utilizes a restricted subset of AMR modes.

Support for AMR2 and AMR Mode Change Restrictions are specified as an optional, but recommended requirement by the IMS VoLTE specification IR.92, "IMS Profile for Voice and SMS". AMR2 provides compatibility with multiple AMR codec types, including FR AMR, HR AMR, UMTS AMR, and OHR AMR.

Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports mode-set and mode-change restrictions in offer/answer SDP for both AMR-NB and AMR-WB. RFC 4867 and 3GPP TS 26.114 describe AMR2 and how the AMR2 features are negotiated using SDP. The IPM device provides the media path processing required for the following SDP parameters:

- mode-set
- mode-change-capability
- mode-change-neighbor
- mode-change-period

For more information on using the AMR-NB and AMR-WB coders, refer to the “Using the AMR-NB and AMR-WB Audio Coder” chapter of the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

The following new macros are bitwise OR'd into the unCoderOptions field of the existing IPM_AUDIO_CODER_OPTIONS_INFO structure to configure AMR2.

- CODER_OPT_AMR_MODE_CHANGE_NEIGHBOR(neighbor) where "neighbor" is 0 or 1 as defined in RFC 4867.
- CODER_OPT_AMR_MODE_CHANGE_PERIOD(period) where "period" is 1 or 2 as defined in RFC 4867.
- CODER_OPT_AMR_MODE_SET(set) where each bit in "set" indicates a bitrate in the mode-set. The bit definitions are as follows.

Bit Definition for AMR

Bit	Bitrate (kbps)
0	4.75
1	5.15
2	5.90
3	6.70
4	7.40
5	7.95
6	10.2
7	12.2

Bit Definition for AMR-WB

Bit	Bitrate (kbps)
0	6.60
1	8.85
2	12.65
3	14.25
4	15.85
5	18.25
6	19.85
7	23.05
8	23.85

CMR rule and packing mode configuration are supported in previous releases.

The following macros have been added to this release.

CODER_OPT_AMR_CMR_RULE(rule) where "rule" is
CODER_OPT_AMR_CMR_TRACK or CODER_OPT_AMR_CMR_LIMIT

CODER_OPT_AMR_PACKING_MODE(mode) where "mode" is
CODER_OPT_AMR_OCTET or CODER_OPT_AMR EFFICIENT

■ Example

```
void StartMedia()
{
    int index = 0;
    IPM_MEDIA_INFO MediaInfo;
    info.MediaData[index].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
    info.MediaData[index].mediaInfo.AudioCoderInfo.eCoderType = CODER_TYPE_AMRNB_12_2k;
    info.MediaData[index].mediaInfo.AudioCoderInfo.eFrameSize = CODER_FRAME_SIZE_20;
    info.MediaData[index].mediaInfo.AudioCoderInfo.unFramesPerPkt = 1;
    info.MediaData[index].mediaInfo.AudioCoderInfo.eVadEnable = CODER_VAD_ENABLE;
    info.MediaData[index].mediaInfo.AudioCoderInfo.unCoderPayloadType = 96;
    info.MediaData[index].mediaInfo.AudioCoderInfo.unRedPayloadType = 0;
    index++;

    ...

    memset(&MediaInfo, 0, sizeof(IPM_MEDIA_INFO));
    MediaInfo.MediaData[index].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_OPTIONS_INFO;
    MediaInfo.MediaData[index].mediaInfo.AudioCoderOptionsInfo.unVersion =
        IPM_AUDIO_CODER_OPTIONS_INFO_VERSION;
    MediaInfo.MediaData[index].mediaInfo.AudioCoderOptionsInfo.unCoderOptions =
        CODER_OPT_AMR_CMR_RULE( CODER_OPT_AMR_CMR_TRACK ) |
        CODER_OPT_AMR_PACKING_MODE( CODER_OPT_AMR_OCTET ) |
        CODER_OPT_AMR_MODE_CHANGE_NEIGHBOR( 1 ) |
        CODER_OPT_AMR_MODE_CHANGE_PERIOD( 2 ) |
        CODER_OPT_AMR_MODE_SET( 0xFF ); // 0x1FF for AMR-WB when all bitrates are set

    index++;
    MediaInfo.unCount = index;
}
```



```

if( ipm_StartMedia(handle, &MediaInfo, DATA_IP_TDM_BIDIRECTIONAL, EV_ASYNC) == -1 )
{
    printf("ipm_StartMedia() failed, %s (%ld)\n",
        ATDV_ERRMSGP(handle), ATDV_LASTERR(handle));
}
}

```

1.16 Send/Receive RFC 2833/RFC 4733 Tone Events Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 supports send/receive of the full range of RFC 4733 tone events (0-255). This feature allows an application to transmit a sequence of both DTMF and non-DTMF telephony events over an IP network by calling **ipm_SendTelephonySignals()**. This can be used in generating all RFC 4733 tone event definitions (0-255) beyond the initial set of DTMF telephony events (0-15) used to represent digits 0-9, A-D, *, #. This can be used to generate a non-DTMF telephony event, such as a Hookflash event, and DTMF RFC 2833/RFC 4733 RTP telephony events based on WebRTC signaling events in a WebRTC Gateway application. The changes in this release also allow an application to support the modem and text tone event definitions specified in RFC 4734 (<https://tools.ietf.org/html/rfc4734>), or channel oriented signaling tone events specified in RFC 5244 (<https://tools.ietf.org/html/rfc5244>). The send/receive RFC 2833/RFC 4733 tone events feature is not supported in Dialogic® PowerMedia™ HMP for Windows Release 3.0.

The RFC 4733 (<https://tools.ietf.org/html/rfc4733>) recommendation specifies the "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals" and obsoletes the original RFC 2833 specification. The send/receive RFC 4733 tone event capability is integrated into the standard DTMF digit generation and detection API when RFC 2833/RFC 4733 mode is negotiated upon SDP media session establishment.

1.16.1 DTMF Transfer Modes

The DTMFXERMODE_RFC2833 DTMF transfer mode's capabilities are extended so that telephony events (e.g., DTMF events) are sent and received in the RTP stream as defined in RFC 2833 and RFC 4733. On the transmit side, inband tones are clamped and converted into telephony events. An application can optionally generate telephony events by calling **ipm_SendTelephonySignals()**.

The new DTMFXERMODE_RFC2833_APP transfer mode is the same as DTMFXERMODE_RFC2833 except that inband tones are not converted into DTMF events. The tones are still clamped. In this mode, telephony events are generated exclusively by the application when **ipm_SendTelephonySignals()** is called.

1.16.2 ipm_SendTelephonySignals()

Name:	int ipm_SendTelephonySignals(nDeviceHandle, *pInfo, usMode)	
Inputs:	int nDeviceHandle	• IP Media device handle
	IPM_TELEPHONY_SEQUENCE_INFO *pInfo	• pointer to telephony event sequence information structure
	unsigned short usMode	• async or sync mode setting
Returns:	0 if success -1 if failure	
Includes:	srllib.h ipmlib.h	
Category:	Media Session	
Mode:	asynchronous or synchronous	

■ Description

The **ipm_SendTelephonySignals()** function instructs the IPM device to generate a sequence of RFC 2833/RFC 4733 telephony events over an IP network. The on/off time and volume of each telephony event is configurable.

The transfer mode must be set to DTMFXERMODE_RFC2833 or DTMFXERMODE_RFC2833_APP for the telephony events to be transmitted on the network. Refer to the **ipm_SetParm()** in the *Dialogic® IP Media Library API Programming Guide and Library Reference* for more information.

Parameter	Description
nDeviceHandle	Handle of the IP Media device.
pInfo	Pointer to IPM_TELEPHONY_SEQUENCE_INFO structure.
usMode	Operation mode. Set to EV_ASYNC for asynchronous execution or to EV_SYNC for synchronous execution.

■ Termination Events

IPMEV_SEND_TELEPHONY_SIGNALS

Indicates successful completion. The given telephony event sequence has been transmitted to the remote endpoint. If **ipm_Stop()** is called while a sequence is being generated, generation is stopped immediately and the IPMEV_SEND_TELEPHONY_SIGNALS termination event is generated, followed by the IPMEV_STOP termination event.

IPMEV_SEND_TELEPHONY_SIGNALS_FAIL

Indicates that the function failed. See the "Errors" section below for a list of error codes.

■ Cautions

When the transfer mode is set to `DTMFXFERMODE_RFC2833`, inband tones that are converted to telephony events will conflict with telephony events that are generated by the application at the same time.

The `DTMFXFERMODE_RFC2833_APP` mode disables telephony event generation from inband tones on the transmit side. While in `DTMFXFERMODE_RFC2833_APP` mode, an application can detect inbound tones or telephony events using a DX device or detect inbound telephony events using IPM telephony event reporting. The detected tones/events can be regenerated using **`ipm_SendTelephonySignals()`**.

■ Errors

If the function returns -1 to indicate failure, call **`ATDV_LASTERR()`** and **`ATDV_ERRMSGP()`** to return one of the following errors:

`EIPM_BUSY`

Channel is busy.

`EIPM_INTERNAL`

Internal error.

`EIPM_INV_MODE`

Invalid mode.

`EIPM_INV_STATE`

Invalid state. Initial command did not complete before another function call was made.

`EIPM_SYSTEM`

System error.

If the `IPMEV_SEND_TELEPHONY_SIGNALS_FAIL` termination event is received, call **`ATDV_LASTERR()`** and **`ATDV_ERRMSGP()`** to return one of the following errors:

`EIPM_INVALID_EVENT_ID`

The event ID is not in the range 0 through 255.

`EIPM_INVALID_VOLUME`

The volume is not in the range 0 through 63.

`EIPM_INVALID_OPTIONS`

Options value is wrong.

`EIPM_INVALID_SIGNAL_TYPE`

Unrecognized signal type. The signal type must be "event".

EIPM_PAYLOAD_TYPE_NOT_IMPLEMENTED

Per sequence/per event RTP payload type is not implemented. The telephony event payload type must be set by the PARMCH_RFC2833EVT_TX_PLT to **ipm_SetParm()**.

EIPM_PTIME_NOT_IMPLEMENTED

Per sequence/per event ptime is not implemented. The frame time of the codec selected in **ipm_StartMedia()** is always used.

EIPM_CLOCK_RATE_NOT_IMPLEMENTED

Per sequence/per event clock rate is not implemented. The RTP clock rate of the codec selected in **ipm_StartMedia()** is always used.

EIPM_TONE_NOT_IMPLEMENTED

This function can only generate telephony events. Telephony tones can not be generated by this function.

EIPM_OUT_OF_RANGE

An event ID in the "telephony event ID string" is too large. The maximum value is 255.

EIPM_TOO_MANY_DIGITS

An event ID in the "telephony event ID string" contains too many digits. The maximum number of digits is 3.

EIPM_INVALID_CHARACTER

The "telephony event ID string" contains an invalid character. Valid characters are comma and decimal digits.

■ Example 1

In this example, the "telephony event ID string" is used to generate the sequence.

```
void SendTelephonySignalsExample1(int handle)
{
    IPM_TELEPHONY_SEQUENCE_INFO seq;
    // The second parameter to the INIT function is set to zero since it doesn't apply
    // to strEventIDs. It's only used for the signal array.
    INIT_IPM_TELEPHONY_SEQUENCE_INFO(&seq, 0);

    // These apply to each event in the event ID string.
    seq.sVolume = 7; // dBm0/sign dropped as defined in RFC 4733
    seq.usDuration = 200; // milliseconds

    // This is the duration of the gap between events in the event ID string.
    seq.unInterval = 100; // milliseconds

    // The event ID string is a comma separated list of telephony event IDs. The following
    // list includes DTMF 1,2,3 and hook flash.

    seq.strEventIDs = "1,2,3,16";
```

```

    if( ipm_SendTelephonySignals(handle, &seq, EV_ASYNC) == -1 )
    {
        printf("ipm_SendTelephonySignals() failed, %s (%ld)\n", ATDV_ERRMSGP(handle),
            ATDV_LASTERR(handle));
    }
    FREE_IPM_TELEPHONY_SEQUENCE_INFO(&seq); // this must be called
}

```

■ Example 2

In this example, the “telephony signal array” is used to generate the sequence. The volume and duration can be configured for each event.

```

void SendTelephonySignalsExample2(int handle)
{
    IPM_TELEPHONY_SEQUENCE_INFO seq;
    PIPM_TELEPHONY_EVENT_INFO pEventInfo;

    // The "count" parameter is set to 3 since there are 3 elements in the telephony
    // event array defined below.
    INIT_IPM_TELEPHONY_SEQUENCE_INFO(&seq, 3);

    // These are the default values that will be used for the event array elements.
    seq.sVolume      = 7;
    seq.usDuration    = 200;
    seq.unInterval    = 100;

    // The volume and duration aren't set in this element, so the default values
    // above are used.
    pEventInfo = INIT_IPM_TELEPHONY_EVENT_INFO(&seq);
    pEventInfo->eTelephonyEventID = SIGNAL_ID_EVENT_DTMF_0;

    // The volume and duration set on the next two events override the volume, duration
    // and interval values set above.
    pEventInfo = INIT_IPM_TELEPHONY_EVENT_INFO(&seq);
    pEventInfo->eTelephonyEventID = SIGNAL_ID_OFF; // pseudo event ID to insert a gap between
events
    pEventInfo->usDuration      = 150;
    pEventInfo = INIT_IPM_TELEPHONY_EVENT_INFO(&seq);
    pEventInfo->eTelephonyEventID = SIGNAL_ID_EVENT_DTMF_1;
    pEventInfo->sVolume          = 5;
    pEventInfo->usDuration      = 185; // this is rounded up to the next frame period, e.g. 200
milliseconds for 20 millisecond G.711

    if( ipm_SendTelephonySignals(handle_, &seq, EV_ASYNC) == -1 )
    {
        printf("ipm_SendTelephonySignals() failed, %s (%ld)\n", ATDV_ERRMSGP(handle_),
            ATDV_LASTERR(handle_));
    }
    FREE_IPM_TELEPHONY_SEQUENCE_INFO(&seq); // this must be called
}

```

1.16.3 IPM_TELEPHONY_SEQUENCE_INFO

```

typedef struct ipm_telephony_signal_sequence_info_tag
{
    unsigned int    unVersion;
    short           sPayloadType;
    short           sPTime;
    int             nClockRate;
    short           sVolume;
    unsigned short  usDuration;
    unsigned short  usDurationMultiplier;
}

```

```

    unsigned int    unInterval;
    unsigned int    unOptions;
    const char      *strEventIDs;
    unsigned short  unSignalInfoCount;
    unsigned short  unSignalInfoIndex;
    PIPM_TELEPHONY_SIGNAL_INFO pSignalInfo;
} IPM_TELEPHONY_SEQUENCE_INFO, *PIPM_TELEPHONY_SEQUENCE_INFO;

```

■ Description

The `INIT_IPM_TELEPHONY_SEQUENCE_INFO` inline function must be called to initialize the `IPM_TELEPHONY_SEQUENCE_INFO` structure. The `FREE_IPM_TELEPHONY_SEQUENCE_INFO` inline function must be called after **`ipm_SendTelephonySignals()`** returns.

The event sequence to generate is specified in a "telephony event ID string" and/or a "telephony signal array". The event string and the signal array can be used individually or together. If the event string and signal array are both given, the events for the event string are generated first.

`INIT_IPM_TELEPHONY_SEQUENCE_INFO(PIPM_TELEPHONY_SEQUENCE_INFO pStruct, unsigned int unSignalCount)`

This function allocates memory for the telephony signal array. `unSignalCount` is the number of elements in the signal array.

`FREE_IPM_TELEPHONY_SEQUENCE_INFO(PIPM_TELEPHONY_SEQUENCE_INFO pStruct)`

This function frees memory allocated by the `INIT_IPM_TELEPHONY_SEQUENCE_INFO` function.

■ Field Descriptions

`sVolume`

The volume of the telephony event in units of dBm0 with the sign dropped as defined in RFC 4733. The default value set by the `INIT` function is 7.

`usDuration`

The "on time" of the telephony events in the event string and signal array in milliseconds. The default value set by the `INIT` function is 100 milliseconds. Internally, the duration is rounded up to the closest codec frame period (e.g., for 20 millisecond G.711, a 50 millisecond duration is rounded up to 60 milliseconds). The maximum value depends on the codec timestamp resolution as defined in RFC 4733 (e.g., 8 seconds for G.711).

`unInterval`

The time between telephony events defined by the event string and signal array in milliseconds. The default value set by the `INIT` function is 50 milliseconds. Internally, the duration is rounded up to the closest codec frame period.

strEventIDs

A string of comma separated telephony event IDs (e.g., "1,2,3" represents DTMF 1, 2, and 3). The values set for sVolume, usDuration and unInterval are used. The default value set by the INIT function is NULL which means the event string is not used.

The following fields must not be set directly by the application. These fields are set by the INIT_IPM_TELEPHONY_SEQUENCE_INFO function.

unVersion

Version of the data structure. Used to ensure that an application is binary compatible with future changes to this data structure.

unSignalInfoCount

The number of element in the signal array. This field is set by the "unSignalCount" parameter passed to the INIT_IPM_TELEPHONY_SEQUENCE_INFO structure.

pSignalInfo

The signal array allocated by the INIT function. The number of elements is "unSignalCount" passed to the INIT_IPM_TELEPHONY_SEQUENCE_INFO function.

unSignalInfoIndex

For internal use.

unOptions

Reserved for future use.

sPayloadType

Reserved for future use.

sPTime

Reserved for future use.

nClockRate

Reserved for future use.

usDurationMultiplier

Reserved for future use.

1.16.4 IPM_TELEPHONY_SIGNAL_INFO

```
typedef struct ipm_telephony_signal_info_tag
{
    unsigned int    unVersion;
    short           sPayloadType;
    short           sPTime;
    int             nClockRate;
    unsigned short  usDurationMultiplier;
    unsigned int    unOptions;
    eIPM_TELEPHONY_INFO_TYPE eTelInfoType;
    union
```

```

    {
        IPM_TELEPHONY_EVENT_INFO TelEvtInfo;
        IPM_TELEPHONY_TONE_INFO TelToneInfo;
    }TelephonyInfo;
} IPM_TELEPHONY_SIGNAL_INFO, *PIPM_TELEPHONY_SIGNAL_INFO;

```

■ Description

This structure contains telephony information (such as RFC 2833 information) that is to be transmitted over an IP network.

The INIT_IPM_TELEPHONY_SIGNAL_INFO inline function must be called to initialize the IPM_TELEPHONY_SIGNAL_INFO structure.

INIT_IPM_TELEPHONY_SIGNAL_INFO(PIPM_TELEPHONY_SEQUENCE_INFO pSeq)
This function returns a pointer to the current IPM_TELEPHONY_SIGNAL_INFO array element.

■ Field Descriptions

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

eTelInfoType

- The information type (e.g., an RFC 2833/4733 named event).
- The eIPM_TELEPHONY_INFO_TYPE data type is an enumeration, which defines the following values:
 - TEL_INFOTYPE_EVENT indicates that the union in this structure is the IPM_TELEPHONY_EVENT_INFO structure.
 - TEL_INFOTYPE_TONE indicates that the union in this structure is the IPM_TELEPHONY_TONE_INFO structure. Reserved for future use.

TelephonyInfo.TelEvtInfo

Named event information (e.g., RFC 2833 DTMF digit). See IPM_TELEPHONY_EVENT_INFO for more information.

TelephonyInfo.TelToneInfo

Telephony tone information. Reserved for future use.

The following fields must not be set directly by the application. These fields are set by the INIT_IPM_TELEPHONY_SIGNAL_INFO function.

unVersion

Version of the data structure. Used to ensure that an application is binary compatible with future changes to this data structure.

unOptions

Reserved for future use.

sPayloadType
Reserved for future use.

sPTime
Reserved for future use.

nClockRate
Reserved for future use.

usDurationMultiplier
Reserved for future use.

1.16.5 IPM_TELEPHONY_EVENT_INFO

```
typedef struct ipm_telephony_event_info_tag
{
    unsigned int          unVersion;
    eIPM_TELEPHONY_EVENT_ID eTelephonyEventID;
    short                 sVolume;
    unsigned short         usDuration;
} IPM_TELEPHONY_EVENT_INFO, *PIPM_TELEPHONY_EVENT_INFO;
```

■ Description

The IPM_TELEPHONY_EVENT_INFO data structure contains information about a telephony event, for example a DTMF event. This structure is a child structure of the IPM_TELEPHONY_INFO and the IPM_TELEPHONY_SIGNAL_INFO data structures.

The IPM_TELEPHONY_EVENT_INFO structure is used to report received telephony events and is used to transmit telephony events over an IP network. See **ipm_EnableEvents()** and the EVT_TELEPHONY event type for more information about receiving telephony events and **ipm_SendTelephonySignals()** for more information about transmitting telephony events.

The INIT_IPM_TELEPHONY_EVENT_INFO inline function must be called to initialize the IPM_TELEPHONY_EVENT_INFO structure when used for **ipm_SendTelephonySignals()**.

INIT_IPM_TELEPHONY_EVENT_INFO(PIPM_TELEPHONY_SEQUENCE_INFO pSeq)
This function returns a pointer to the current IPM_TELEPHONY_EVENT_INFO array. An internal array index is incremented.

■ Field Descriptions

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

unVersion
Version of the IPM_TELEPHONY_EVENT_INFO structure. This field is used by the IP Media Library for checking the backward binary compatibility of future versions of the data structure.

eTelephonyEventID

A named event. The data type of the telephony_event field is an eIPM_TELEPHONY_EVENT_ID enumeration that lists all possible telephony event identifiers as described in RFC 2833. For RFC 4733 named events, use the equivalent RFC 2833 enumeration value or cast a value to the eTelephonyEventID type. The DTMF enumeration values are:

- SIGNAL_ID_EVENT_DTMF_0
- SIGNAL_ID_EVENT_DTMF_1
- SIGNAL_ID_EVENT_DTMF_2
- SIGNAL_ID_EVENT_DTMF_3
- SIGNAL_ID_EVENT_DTMF_4
- SIGNAL_ID_EVENT_DTMF_5
- SIGNAL_ID_EVENT_DTMF_6
- SIGNAL_ID_EVENT_DTMF_7
- SIGNAL_ID_EVENT_DTMF_8
- SIGNAL_ID_EVENT_DTMF_9
- SIGNAL_ID_EVENT_DTMF_STAR
- SIGNAL_ID_EVENT_DTMF_POUND
- SIGNAL_ID_EVENT_DTMF_A
- SIGNAL_ID_EVENT_DTMF_B
- SIGNAL_ID_EVENT_DTMF_C
- SIGNAL_ID_EVENT_DTMF_D

sVolume

The volume of the telephony event. The INIT function initializes this field to the value of sVolume in the IPM_TELEPHONY_SEQUENCE_INFO structure.

usDuration

The duration of the telephony event. For a received event, the duration is in timestamp units. For a transmitted event, the duration is in milliseconds. The INIT function initializes this field to the value of usDuration in the IPM_TELEPHONY_SEQUENCE_INFO structure.

1.17 Expanded Interface Identification Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 has been updated to allow the use of an interface that is not stored in the list reported by **ipm_GetLocalMediaInfo()**. The application can use the Linux glibc **getifaddrs()** API to identify available network interfaces programmatically. OA&M components will still be used to notify HMP of the primary interface. This is required in cases where the application does not pass a LOCAL RTP media info to **ipm_StartMedia()**. The **ipm_StartMedia()** API will report an error if an invalid IP address is passed in for a local media address.

1.18 Red Hat Enterprise Linux Release 7 Update 1 Support

Service Update 213 announces support for Red Hat Enterprise Linux Release 7 Update 1.

1.19 SUSE Linux Enterprise Server 11 Service Pack 3 64-bit Support

With Service Update 213, Dialogic® PowerMedia™ HMP for Linux Release 4.1 adds support for SUSE Linux Enterprise Server 11 Service Pack 3 64-bit. The following describes additional requirements.

1.19.1 Selecting Linux Kernel Source for Dialogic® HMP Interface Boards

If you use Dialogic® HMP Interface Boards in a system with SUSE Linux 11.3, source for the Linux kernel (kernel headers) must be installed on your system. This can be done by selecting these packages during the initial OS install or after the initial OS install by installing these source RPMs from your Linux OS distribution media.

On SUSE Linux, after the kernel source is installed, you must reconfigure the kernel source as follows:

- In the `/usr/src/linux` directory:
 - `run make cloneconfig`
 - `run make modules_prepare`
- Remove symbolic link `/lib/modules/'uname -r'/build`.
- Create a symbolic link `/lib/modules/'uname -r'/build` and direct it to `/lib/modules/'uname -r'/source`.

1.20 Oracle Enterprise Linux 6.2 64-bit Support

Since Service Update 141, Dialogic® PowerMedia™ HMP for Linux Release 4.1 has supported Oracle Enterprise Linux 6.2 64-bit with Unbreakable Linux Kernel (UEK).

1.21 Operating System Distributions No Longer Supported

With Service Update 165, Dialogic® PowerMedia™ HMP for Linux Release 4.1 will no longer support Red Hat Enterprise Linux Release 4 and SUSE Linux Enterprise Server 9 beyond Service Update 161.

Because Red Hat Enterprise Linux Release 4 and SUSE Linux Enterprise Server 9 are no longer supported, the GCC 3.2.3 and GCC 3.4.3 compiled packages from the software package have been removed with Service Update 213.

1.22 Support for Multiple NICs for Audio Media Sessions in 1PCC Mode

With Service Update 161, Dialogic® PowerMedia™ HMP for Linux Release 4.1 introduces support for using multiple local IP interfaces for audio sessions in Global Call SIP 1PCC mode.

1.22.1 Feature Description

Prior to this feature, the Global Call IP Technology library tied all RTP/RTCP ports used for media streaming into one local IP address, and thus only one Network Interface Card (NIC). With this feature, audio streaming RTP/RTCP ports are freely associated to one or more local IP address, independent of the call control IP address, and selectable on a Call or LineDevice basis at runtime in SIP 1PCC mode.

There are two main aspects of the feature; one is allowing for automatic OA&M HMP registration of all local, enabled, Network Interfaces in the system, making them available to the run-time media library. The second aspect involves the Global Call IP Technology library, which now provides the list of IP addresses provided by OA&M to the application.

The application is able to select any of the IP addresses on a IPT board device basis, LineDevice channel basis, and can change it during a particular Call; some limitations exist due to the underlying IP Media library.

This feature is built upon the previously available [Support for Multiple NICs](#). By means of it, HMP automatically retrieves the list of available local IP addresses enabled for IP traffic in a system, registers them, and then makes them available to the HMP IP Media SW component as a list. This list indicates which NIC is primary, with all remaining ones as secondary IP NICs. The primary NIC is used for all IP media streaming on all the IP Media channels by default, consequently is also used for all Global Call media sessions in the 1PCC mode.

1.22.2 Feature Limitations

- This feature is only available for SIP applications using the 1st Party Call Control (1PCC); however 3PCC applications can take advantage of the IP Media library ability to handle multiple local IP addresses for media sessions as indicated in [Support for Multiple NICs](#).

Note: Video is not supported in 1PCC mode, thus the feature doesn't apply to it.

- This feature is not applicable to T.38 Fax Server mode. T.38 Fax Server RTP/RTCP ports continue being tied to the HMP OA&M default local IP interface.
Note: Changing the default local IP interface at IPT board or channel level does not affect the local IP interface used by T.38 Fax.
- Overriding the media session IP address on a Call basis is only possible via the **gc_MakeCall()**; the **gc_AcceptXfer()** or the **gc_InvokeXfer()** do not allow the overriding of the media IP address.
- The scope of changing the default local IP interface at IPT board or its channel devices is limited to the application; it has no effect on the system-wide OA&M setting for the streaming interface.
- Likewise, the scope of changing the default local IP interface at IPT board or its channel devices is limited to the specific IPT board; other IPT boards, if so exist, are not affected.

1.22.3 Runtime Implementation

An application uses the Global Call library functions to control Audio Sessions within a dialog established in the 1PCC mode. The following functionality is available with this feature.

1.22.3.1 Retrieving Available RTP IP Addresses from the Application Using **gc_GetUserInfo()**

The application could use **gc_GetUserInfo()** to retrieve the available **RTP_ADDRESS** for a specific channel.

- The target type is **GCTGT_GCLIB_CHAN**
- The target id is an IPT LineDevice
- The target data is a **GC_PARM_BLK** with setId **IPSET_RTP_ADDRESS**, and parmId **IPPARM_LOCAL_ENUM**

The function will return an array of **RTP_ADDRESS** structures which contain the available RTP addresses. The first elements will be the IPv4 addresses, then the IPv6 addresses. The first one in each group is the default RTP address for that type of calls (IPv4 or IPv6 as applicable).

1.22.3.2 Example

```
int GetAvailableRTPAddresses(int nLineHandle)
{
    int          nError = GC_SUCCESS;
    long         nRequestID = 0;
    GC_PARM_BLK  parmblkpRTP = NULL;
    RTP_ADDR     rtpA = {0};
    RTP_ADDR*    pRTPAddress = NULL;

    GC_PARM_DATA_EXT parm_data_ext;

    INIT_GC_PARM_DATA_EXT(&parm_data_ext);

    /*insert into the GC_PARM_BLK*/
```

```

        gc_util_insert_parm_ref(&parmbldpRTP, IPSET_RTP_ADDRESS, IPPARM_LOCAL_ENUM,
sizeof(rtpA), &rtpA);

        nError = gc_GetUserInfo(GCTGT_GCLIB_CHAN,nBoardHandle, &parmbldpRTP, 0,
                                GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC) ;
        if (nError == GC_SUCCESS)
        {
            /*iterate through the array of RTP addresses*/
            while ( GC_SUCCESS == (gc_util_next_parm_ex(parmbldpRTP, &parm_dat_ext)) )
            {
                /* Process set_ID/parm_ID pairs */
                if ((parm_data_ext.set_ID == IPSET_RTP_ADDRESS)
                    && (parm_data_ext.parm_ID == IPPARM_LOCAL_ENUM))
                {
                    pRTPAddress = (RTP_ADDR *)parm_data_ext.pData;

                    char localAddress[1024];
                    size_t dwAddressSize = 1024;
                    if (pRTPAddress->ip_ver == IPVER4)
                        inet_ntop(AF_INET, (char*)&rtpA.u_ipaddr.ipv4,
localAddress, dwAddressSize);
                    else
                        inet_ntop(AF_INET6, (char*)&rtpA.u_ipaddr.ipv6,
localAddress, dwAddressSize);

                    /*save or print RTP address which is now in localAddress
buffer*/

                }
            }

            gc_util_delete_parm_blk(parmbldpRTP);

            return nError;
        }
}

```

Note: Since the RTP address list is global per system, the application could retrieve it only once, on the first open line device. It makes no difference if retrieving the list on every channel, because the returned values will be the same.

1.22.4 Setting Audio Sessions RTP/RTCP IP Address from the Application

For setting the **RTP_ADDRESS** for a Global Call IPT channel, the application has several methods:

- Use **gc_SetConfigData()** on a Global Call IPT board device
- Use **gc_SetUserInfo()** on a Global Call line device
- Insert a **RTP_ADDRESS** in a **GC_PARM_BLK** in **gc_MakeCall()**

Note: The application could set an IPv4 address and an IPv6 address. Depending on the call type, the corresponding one will be used.

1.22.4.1 Setting Audio Sessions Address Globally on All Channel Devices Using **gc_SetConfigData()**

The application could use **gc_SetConfigData()** as a convenience function to set the **RTP_ADDRESS** for all channels of the board and all calls subsequent to it:

- The target type is **GCTGT_CCLIB_NETIF**
- The target id is an IPT LineDevice
- The target data is a **GC_PARM_BLK** which contains a **RTP_ADDRESS** using setld **IPSET_RTP_ADDRESS**, and parmld **IPPARM_LOCAL**

1.22.4.2 Example

```
int SetBoardRTPAddress(int nBoardHandle, const char* sIPAddress)
{
    int                nError = GC_SUCCESS;
    long               nRequestID = 0;
    GC_PARM_BLK        parmblkpRTP = NULL;
    RTP_ADDR           rtpA = {0};

    rtpA.ip_ver = IPVER4; /*could be IPVER6 */

    inet_pton(AF_INET, sIPAddress, &rtpA.u_ipaddr.ipv4);

    /*insert into the GC_PARM_BLK*/
    gc_util_insert_parm_ref(&parmblkpRTP, IPSET_RTP_ADDRESS, IPPARM_LOCAL, sizeof(rtpA),
&rtpA);

    nError = gc_SetConfigData(GCTGT_CCLIB_NETIF,nBoardHandle, parmblkpRTP) ;
    gc_util_delete_parm_blk(parmblkpRTP);

    return nError;
}
```

1.22.4.3 Overriding Audio Sessions IP Address on a Channel Device Using gc_SetUserInfo()

The application could use **gc_SetUserInfo()** to override the **RTP_ADDRESS** for a specific channel, for the next call or for all calls.

- The target type is **GCTGT_GCLIB_CHAN**
- The target id is an IPT LineDevice
- The target data is a **GC_PARM_BLK** which contains an **RTP_ADDRESS** using setld **IPSET_RTP_ADDRESS**, and parmld **IPPARM_LOCAL**
- Duration must be set to **GC_ALLCALLS**

Note: GC_SINGLECALL duration is not supported and will have no effect; to set the media session address for one Call use the **gc_MakeCall()** method.

1.22.4.4 Example

```
int SetLineRTPAddress(int nLineHandle, const char* sIPAddress)
{
    int                nError = GC_SUCCESS;
    GC_PARM_BLK        parmblkpRTP = NULL;
    RTP_ADDR           rtpA = {0};
    rtpA.ip_ver = IPVER4; /*could be IPVER6 */

    inet_pton(AF_INET, sIPAddress, &rtpA.u_ipaddr.ipv4);

    /*insert into the GC_PARM_BLK*/
    gc_util_insert_parm_ref(&parmblkpRTP, IPSET_RTP_ADDRESS, IPPARM_LOCAL, sizeof(rtpA),
&rtpA);
}
```

```

nError = gc_SetUserInfo(GCTGT_GCLIB_CHAN,nLineHandle, parmblkpRTP, GC_ALLCALLS) ;
gc_util_delete_parm_blk(parmblkpRTP);

return nError;
}

```

In order that the setting is effective for inbound calls, the function must be called prior to any inbound call, after opening the line device.

The application could set one IPv4 **RTP_ADDRESS**, and one IPv6 **RTP_ADDRESS**. Depending on the call type (IPv4 or IPv6) the corresponding media session address is used.

1.22.4.5 Overriding Audio Sessions IP Address During a Call Using `gc_MakeCall()`

The application could take advantage of the **GCLIB_MAKECALL_BLK** structure within a `gc_MakeCall()` in order to override the **RTP_ADDRESS** for the next IP call.

1.22.4.6 Example

```

void Set_GCLIBMAKECALL_BLK_RTPAddress(GCLIB_MAKECALL_BLK pGCMKB, const char* sIPAddress)
{
    GC_PARM_BLK parmblkpRTP = pGCMKB->ext_datap;
    RTP_ADDR rtpA = {0};

    rtpA.ip_ver = IPVER4; /*could be IPVER6 on Linux*/

    inet_pton(AF_INET, sIPAddress, &rtpA.u_ipaddr.ipv4);

    /*insert into the GC_PARM_BLK*/
    gc_util_insert_parm_ref(&parmblkpRTP, IPSET_RTP_ADDRESS, IPPARM_LOCAL, sizeof(rtpA),
&rtpA);
}

```

Note: The behavior of a call that does not choose to override the **RTP_ADDRESS** depends on two factors:

- If no previous Call overrode the **RTP_ADDRESS** on the same channel device as indicated in this method:
 - If the **RTP_ADDRESS** was set on a channel device as specified in [Setting Audio Sessions Address Globally on All Channel Devices Using `gc_SetConfigData\(\)`](#) or [Overriding Audio Sessions IP Address on a Channel Device Using `gc_SetUserInfo\(\)`](#), it takes effect.
 - Otherwise the default system-wide **RTP_ADDRESS** takes effect; that is the first one in the group as per [Retrieving Available RTP IP Addresses from the Application Using `gc_GetUserInfo\(\)`](#).
- If a previous Call for the same channel device did override the **RTP_ADDRESS** as indicated in this method:
 - If the **RTP_ADDRESS** was set on a channel device as specified in [Setting Audio Sessions Address Globally on All Channel Devices Using `gc_SetConfigData\(\)`](#) or [Overriding Audio Sessions IP Address on a Channel Device Using `gc_SetUserInfo\(\)`](#), it takes effect.

- If no **RTP_ADDRESS** was set on the channel device, the behavior is as follows:

The **RTP_ADDRESS** from the prior Call with an override as described here, takes effect.

1.23 Configuring a Network Interface in Tristate or Line Monitor Modes

Service Update 161 provides the ability to configure an interface in either Tristate or Line Monitor modes. This feature is specific to the Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards.

1.23.1 Feature Description

Tristate or Line Monitor modes are used in cases where there is a need to tap on network interfaces carrying a signalling channel; for example when a line is configured with Combined SS7 functionality. Refer to the [Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards](#) section for more information.

By default these modes are disabled and must be enabled explicitly.

1.23.2 Feature Limitations

- The feature is applicable to ISDN, CAS, and R2MF protocol groups only, and is configurable on a network-interface basis using this particular method.
- This functionality is limited to the DNIxxxxTEPE2HMP Network Interface Boards.
- The functionality is limited to enabling or disabling it (default=Disable); configuring high-impedance and/or gain values is not possible.
- The feature is static in the sense that it remains in effect from the time the board is initialized until it is stopped; at which point it can be reconfigured (or removed).

1.23.3 Feature Configuration

The new Line Administration (LineAdmin) parameters, TristateMode and RxLineMonMode, allow setting a digital line interface in either high impedance, also known as HiZ, or else in Protected Monitoring Point (PMP) mode.

The TriStateMode parameter forces high impedance on the line interface unit (LIU) on transmit and receive paths. This allows it to disable the transmit path for all intents and purposes, and, on the receive path, avoid disrupting the data between two terminated LIUs that are being tapped into. This feature is configured through the LineAdmin parameter interface in the board's CONFIG file.

For the Protected Monitoring Point mode of operation, the line impedance is not altered; however the sensitivity of the line receiver is increased with the expectation that external equipment provides the high-impedance capability in order to avoid signals on the external LIUs being tapped, to be negatively impacted.

By default these parameters are disabled.

The parameters and their values are as shown below:

Parameter Name	Param Value	Settings	Description
LCON_TristateMode	0x1628	0 - Disable (default) 1 - Enable	High impedance mode of operation on the LIU.
LCON_RxLineMonMode	0x1629	0 - Disable (default) 1 - Enable	Protected Monitoring Point mode of operation on the LIU.

Note: The parameters are mutually exclusive; setting both of them on a line interface will yield unexpected results.

These parameters can be used for SS7 combined configurations with the DNlxxxTEPE2HMP; in this case a LIU configured in either mode would be configured with the MONITOR LINK command (MONITOR_LINK). Refer to the *Dialogic® Distributed Signaling Interface Components - Software Environment Programmer's Manual* for information on this command.

Note: Monitoring SS7 messages are passed from MTP2 directly to the User Module via the API_MSG_RX_IND message (0x8f01); however GC SS7 does not provide handling of SS7 messages directly from MTP2, thus GC SS7 will not process monitor-interface messages. Refer to the *Dialogic® SS7 Protocols MTP2 Programmer's Manual* for details, which is found in the “DSI Protocols Stack” page at:

<http://www.dialogic.com/products/signaling-and-ss7-components/download/dsi-interface-protocol-stacks.aspx>

To configure a line with either of the parameters, it must be added in any place after the LineType parameter in the [lineAdmin] section of each line as desired.

For example:

```
[lineAdmin.1]
...
SetParm=0x1628, 1          ! LCON_TristateMode (Default=0, Disable=0, Enable=1)

[lineAdmin.2]
...
SetParm=0x1629, 1          ! LCON_RxLineMonMode (Default=0, Disable=0, Enable=1)
```

Once the CONFIG file has been modified, a new FCD file must be generated prior to board initialization to match the new settings. Refer to the *Dialogic® Host Media Processing for Linux Configuration Guide* for details on how to do this. The parameter setting takes effect once the board is initialized and HMP is started.

Note: Setting either parameter to zero has the same effect as not having it configured in the first place (default=Disabled).

1.24 Support for GSM-FR and GSM-EFR Codecs

Service Update 161 adds support for GSM-FR and GSM-EFR codecs. GSM-FR and GSM-EFR are only supported through 3PCC mode. The following configurations are supported.

GSM-FR	1FPP	Vad Off
GSM-FR	1FPP	Vad On
GSM-FR	2FPP	Vad Off
GSM-FR	2FPP	Vad On
GSM-EFR	1FPP	Vad Off
GSM-EFR	1FPP	Vad On
GSM-EFR	2FPP	Vad Off
GSM-EFR	2FPP	Vad On

Payload types:

- The system shall support the GSM-EFR utilizing the dynamic payload type scheme during SDP negotiations, RTP packetization and processing (RFC 3551).
- The system shall support GSM-FR utilizing the static payload type (3) for SDP negotiations, RTP packetization and processing (RFC 3551).

1.25 Support for 256-bit Master Key Length in Secure RTP

With Service Update 151, the Secure RTP feature in the IP Media Library introduces support for 256-bit Master Key Length with the existing AES Counter-mode (AES-CM) Cipher.

1.25.1 Feature Implementation

The Dialogic® IP Media Library provides Secure RTP (SRTP) functionality as an enhancement to RTP that provides confidentiality, message authentication, and replay protection for the latter.

The SRTP feature had support for two Crypto Suites, AES in Counter Mode with 128-bit Master Key Length, 112-bit Salt Key Length, and either 32- or 80-bit Authentication Tag.

With this Service Update, Master Key Length of 256-bit is now introduced; the rest of the Crypto-suite parameters remain unchanged. Therefore with this new feature, two new Crypto Suites are introduced: 256-bit Master Key Length, 112-bit Salt Key Length, and either 32- or 80-bit Authentication Tag.

The below table complements “Table 7 - Crypto Suite Parameter Values” shown in the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

Characteristic	AES_CM_256_HMAC_SHA1_80	AES_CM_256_HMAC_SHA1_32
Master Key Length	256 bits	256 bits
Salt Value	112 bits	112 bits
Default Lifetime	231 packets	231 packets
Cipher	AES counter mode	AES counter mode
Encryption Key	256 bits	256 bits
MAC	HMAC-SHA1	HMAC-SHA1
Authentication Tag	80 bits	32 bits
SRTP Auth Key Length	160 bits	160 bits
SRTCP Auth Key Length	160 bits	160 bits

- Notes:**
1. This feature still uses the AES Counter Mode (AES-CM) for encryption, as previously supported; the key-derivation algorithm remains unchanged with the Master Key Length, thus its input "block size" remains 128 bits, irrespective of the Master Key Length. Refer to the details described in the *The Secure Real-time Transport Protocol (SRTP)* IETF publication, RFC 3711, available at <http://www.ietf.org/rfc/rfc3711.txt>.
 2. The Salt Key Length for SRTP does not change with this feature, i.e. it remains 112-bit length.
 3. The Authentication Transform for SRTP remains HMAC-SHA1 with configurable 32-bit or 80-bit length.

To take advantage of this new feature, the IP Media Library has been enhanced for secure RTP media sessions, as follows.

The SRTP IPM_SECURITY_KEY data structure, unMasterKeyLength now allows a setting of 256 as length (in bits), in addition to the previously 128 length value supported.

The SRTP IPM_SRTP_PARMS data structure, eIPM_CRYPTOSUITE, which is used for starting a secure RTP streaming session with the **ipm_StartMedia()** or for modifying a secure RTP streaming setting with the **ipm_ModifyMedia()** on an IP Media device, provides two new enumerations in support of this feature. The complete set is now augmented as follows:

```
typedef enum
{
    IPM_CRYPTOSUITE_AES_CM_128_HMAC_SHA1_80 = 1,
```

```
IPM_CRYPT0_AES_CM_128_HMAC_SHA1_32 = 2,
IPM_CRYPT0_AES_CM_256_HMAC_SHA1_80 = 3,
IPM_CRYPT0_AES_CM_256_HMAC_SHA1_32 = 4,
} eIPM_CRYPT0_SUITE;
```

When setting, the `eIPM_CRYPT0_SUITE` to either `IPM_CRYPT0_AES_CM_256_HMAC_SHA1_80` or `IPM_CRYPT0_AES_CM_256_HMAC_SHA1_32`, the `unMasterKeyLength` must be set to a value of 256 or an error will occur at the time of starting or modifying an SRTP session.

In addition, the `ipm_SecurityGenMasterKeys()` convenient function also supports the aforementioned `unMasterKeyLength` 256 value in the `IPM_SECURITY_KEY`.

1.26 Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards

With Service Update 151, the PCI Express half-size, full-profile, Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards, add SS7 protocol stack support comparable to the SS7LD boards, when used in conjunction with the Dialogic® Distributed Signaling Interface (DSI) SW; in addition its functionality is combined with full Dialogic® HMP media streaming.

For systems that already have Dialogic® Distributed Signaling Interface (DSI) operating with SS7LD Network Interface Board(s) within an HMP environment, they continue to operate as before (refer to the [Support for Dialogic® DSI SS7LD Network Interface Board](#) section for more information) with the added benefit that with this SW update, the same stack can run on both types of boards and both types of boards can operate at the same time under one DSI SS7 configuration umbrella.

1.26.1 Feature Implementation

The Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP Digital Network Interface Boards, when operating along the Dialogic® DSI Linux Development Package (DPK) version 6.6.1 or later, introduce support for the same SS7 SW stack that runs on the Dialogic® DSI SS7LD Network Interface Board.

Applications can take full advantage of the Dialogic® Global Call SS7 API which is available for ISUP and/or TUP SS7 protocols. In addition to that, the boards running the SS7 stack can also perform full HMP streaming of its Voice-Circuits, eliminating the need to route the circuits to a second board for that purpose.

To take advantage of this new feature, the Dialogic® DSI Development Package for Linux version 6.6.1 or later (DPK 6.6.1) must also be installed. Separate SW licenses for HMP media and SS7 signaling are required.

Automatic and Manual SS7 Startup mode operation are also possible. The former is still recommended and should be preferred over the Manual mode.

1.26.2 Feature Limitations

- Refer to the *Dialogic® DSI Development Package Release Notes* for up-to-date information regarding signaling support on the DNlxxxxTEPE2HMP Digital Network Interface Boards:
- <http://resource.dialogic.com/telecom/support/ss7/cd/GenericInfo/DevelopmentPackages/Linux/rn003dpk.pdf>
- The DPK installation procedure provides information for Building Device Drivers for DSI boards; the SS7LD (SS7LD_DRIVER) procedure must be bypassed to avoid board malfunctioning. Refer to the [Feature Configuration](#) section for more information.

Note: Installing the SS7LD device driver will cause DNlxxxxTEPE2HMP board(s) to malfunction in ways that might not be obvious.

- HMP IP Media streaming is not possible with the Dialogic® DSI SS7LD Network Interface Board. It continues to operate as it did prior to this SW update; that is, the SS7 voice circuits carried on the SS7LD are terminated at the board.
- While the DNlxxxxTEPE2HMP Digital Network Interface Boards support mixed protocol mode which include ISDN or CAS along with the SS7 stack, limitations exist. Contact your Dialogic sales support representative if this mixed-mode of operation is required.
- Manual mode of the SS7 Startup is allowed; however it should be limited to development and configuration times, while the Automatic startup mode should be used and is recommended for deployment. Refer to the [Board Configuration and Startup](#) section for more information.
- For proper Dialogic® DSI SW operation with the DNlxxxxTEPE2HMP configured with the SS7LD stack, HMP IP Media startup (or re-start) is required prior to DSI gctload startup (or re-start). In Automatic startup mode, this is accomplished within the HMP IP Media startup procedures.
- HMP product licenses do not provide SS7 protocol licensing. Separate SS7 protocol licenses are required; since the DNlxxxxTEPE2HMP is functionally equivalent to the SS7LD Network Interface board, product license are the same. The *Dialogic® DSI Development Package Release Notes* provides information about this.

Note: Only the user-part SS7 protocols supported by Global Call, such as ISUP and TUP, are integrated in HMP and are discussed in this document. Other transaction-based protocols such as SCCP, TCAP, MAP, etc. are outside of the scope of this document, as they require interfacing with the protocol layer using the message based interface as part of the DSI and have no HMP integration.

1.26.3 Feature Configuration

The HMP and DPK SW releases are installed separately and can be done in any particular order; however since both the DNlxxxxTEPE2HMP, and SS7LD Network Interface boards share the same set of device drivers, the user must make sure not to install the DSI SS7LD device driver as part of the DPK installation.

The DSI ssdl is the process that interfaces with either the DNlxxxxTEPE2HMP or the SS7LD device driver for passing messages to and from the board(s). Since the board(s) device driver is provided as part of the HMP run-time environment the user must refrain from installing it with the DPK release. The SS7LD (SS7LD_DRIVER) procedure must be bypassed to avoid board malfunctioning.

Refer to the “Building Device Drivers for DSI boards” section in *Dialogic® Distributed Signaling Interface Components - Software Environment Programmer's Manual*:

<http://www.dialogic.com/~media/manuals/ss7/cd/GenericInfo/GeneralDocumentation/U10SSS-SwEnv-PM.pdf>

For HMP SS7 setups with the DNlXXXXTEPE2HMP, it must be configured for Clear Channel, either T1CC or E1CC. Refer to the *Dialogic® Host Media Processing Software for Linux Configuration Guide* for more information.

The subsections below assume a proper Clear Channel configuration exists.

1.26.3.1 Board Configuration and Startup

Since the ssdl treats the DNlxxxxTEPE2HMP or the SS7LD boards as boards of the same family, and the device driver is common, the board enumeration follows the same mechanism and cannot be distinguished from each other; however the HMP SW does not enumerate SS7LD boards at all.

This means that while the ssdl assigns SS7_BOARD <board_id>(s) to both types of boards, the HMP IP Media Logical ID enumeration only applies to the DNlxxxxTEPE2HMP. The two methods don't necessarily follow the same scheme, and thus it might be difficult to identify the geographical location of boards and correlation with their corresponding SS7_BOARD <board_id>.

Both types of boards provide a rotary switch (thumbwheel) which is the recommended way of ssdl SS7_BOARD board addressing in the case of multiple board setup.

- Notes:**
1. The rotary switch mode of board addressing is not the default, thus the user must set it accordingly.
 2. Refer to the DNlxxxxTEPE2HMP and the SS7LD Quick Install Card (QIC) for information about rotary switch setting.

In particular the following ssdl cmd-line options apply:

-o<addressing mode> -o3 - Switch address mode based addressing, determined by a 16-position rotary switch (SW1) on the board.

-a<address> - The list of the switch settings for each logical board position (or board_id). Each entry in the list is separated by a comma. board_id's are sequentially assigned.

This is only a summary; however the *Dialogic® Distributed Signaling Interface Components - Software Environment Programmer's Manual* provides detailed information about ssdl board addressing modes.

Complete HMP/SS7 startup is a two-step process; first the HMP and DNIXXXTEPE2HMP board subsystems are started as it would be without this feature, and next the SS7 subsystem is started. This can be accomplished in two fashions: Automatic, and Manual. The following sections elaborate more on this.

Note: Whether using the automatic or manual mode for SS7 subsystem, the DSI software environment is still created and maintained using the gctload and s7_mgt utilities; in this particular case the s7_mgt utility will recognize a properly-configured Clear Channel, DNIXXXTEPE2HMP board as SS7 functionality equivalent to an SS7LD board.

If more than one DNlxxxTEPE2HMP are in the system, the boards must be cabled together via the SyncRoute cable and using the proper termination jumper setting; this is irrespective of whether or not the boards are or are not configured with the SS7LD stack. Since the DNlxxxTEPE2HMP perform HMP streaming, proper synchronization-timing source for the streams must be maintained at all times, irrespective of network interface clocking propagation. Streaming synchronization is achieved with the SyncRoute cable.

If one or more SS7LD boards are in the same system, the SyncRoute cable might or might not be required. HMP streaming on these boards is not possible, thus the SyncRoute cable is only required among them and/or with the DNlxxxTEPE2HMP, if network interface clock propagation is required. If so required, then proper setting for the termination jumpers is also required.

Refer to the QIC for each type of board for details on the SyncRoute and its termination jumper. The SS7LD QIC is also called *Dialogic® Distributed Signaling Interface SS7LDH4Q Network Interface Board Installation Guide* and is available at:

<http://www.dialogic.com/~media/manuals/ss7/cd/ProductSpecific/SS7LD/InstallationGuide/SS7LD-IG.pdf>

Whenever boards are cabled together via the SyncRoute bus, only one physical board within the bus can act as the TDM bus master agent; in a mixed system with both types of boards, the TDM clock synchronization setting is configured separately thus care must be taken in properly configuring the TDM clock and network-reference master agent(s). The *Dialogic® Host Media Processing Software for Linux Configuration Guide* and *Dialogic® Distributed Signaling Interface Components - Software Environment Programmer's Manual* provide the necessary information to achieve proper TDM clocking settings.

Automatic DSI SS7 Start and Stop within Dialogic System Service

The Dialogic System Service can be configured to start both subsystems at once. By SS7 subsystem, we refer to the combination of the DSI (DPK) and the GC SS7 SW components in the HMP environment.

This is done by means of the Global Call SS7 configuration (*gcss7.cfg*) file, which can be set up to start the SS7 stack automatically by means of several GC SS7 configuration file parameters; refer to the *Dialogic® Global Call SS7 Technology Guide* for more information about the following parameters.

In this case, the SS7 subsystem will also be stopped along the HMP IP Media stop.

In order to take advantage of this mode, the "Service.GCTLOAD_Control" entry must be set to "Yes" and the "SeptelCard.ConfigDir" entry must point to the proper DSI SS7 protocol configuration file path. In such a case, the GC SS7 *dlgcs7srv* will also use the setting of "Service.GCTLOAD_Path", if set accordingly, to invoke the DSI *gctload* executable. If the latter is not set, the former might be used instead as default.

Normally both of the above paths are set to the DPK installation directory.

For DNIXXXTEPE2HMP and SS7LD boards, the "System.Configuration" entry must be set to "Card"; other settings are also possible for Automatic startup on different protocols so long it's not set to "None".

With automatic SS7 startup mode setting, the HMP IP Media delivers Startup/Stop control for the GC SS7 *dlgcs7srv* and DSI subsystems, readying the system for operation without the need for further action, providing proper SS7 configuration was achieved.

The Automatic SS7 Startup mode is the recommended setting for systems in the field.

Note: Since in this case the *gctload* runs in the background without console output, it's important to set up its logging mechanism accordingly to allow it to troubleshoot it in case of startup procedure failure.

Manual DSI Start and Stop within HMP

In manual mode, the GC SS7 *dlgcs7srv* is not started automatically; which in turn results in the DSI subsystem also not to start automatically. The user is responsible for starting both subsystems separately upon HMP IP Media Start. To stop these subsystems the reverse order must be followed.

Manual Startup mode is achieved anytime the "Service.GCTLOAD_Control" is either commented out, or set to "No".

The user will start the DSI gctload manually upon HMP IP Media start, and then proceed to start the GC SS7 dlgs7srv. On stopping it the reverse order is expected. This setting is useful when configuring the DSI with the DNI Combined for the first time and for troubleshooting purposes.

Note: In this case restarting the dlgs7srv is possible without restarting HMP IP Media and DSI gctload.

The GC SS7 dlgs7srv can manually be invoked from the cmd-line by means of:

```
dlgs7d start
```

If successful "...OK" is printed in its standard output; if not an error is displayed.

The user should stop it first prior to DSI gctload and HMP IP Media shutdown:

```
dlgs7d stop
```

The Service daemon is located in the directory indicated by the INTEL_DIALOGIC_BIN environment variable.

1.26.4 SS7 Configuration

The SS7 SW stack supported on the DNXXXXTEPE2HMP interface boards provides the same SS7 functionality as currently provided on the SS7LD interface board; in particular MTP1 and MTP2 layers run on board and higher protocol levels on host SW.

Therefore, configuration of the SS7 protocols requires minimal changes to an equivalent configuration with the SS7LD.

Refer to the *Dialogic® Distributed Signaling Interface Components - Software Environment Programmer's Manual* for DSI-specific details:

<http://www.dialogic.com/~media/manuals/ss7/cd/GenericInfo/GeneralDocumentation/U10SSS-SwEnv-PM.pdf>

Refer to the *Dialogic® Global Call SS7 Technology Guide* for GC SS7-specific details:

http://www.dialogic.com/~media/manuals/docs/globalcall_for_ss7_v6.pdf

Note: For ISUP-based systems, run-time licenses which bundle ISUP, MTP3, and MTP2 functionality that run within the DSI ssdl component, are available.

1.26.4.1 DSI System Configuration File

These are the minimum System Configuration File settings required for proper SS7 operation; this does not provide a comprehensive list as other commands might also be used depending on the specific system configuration.

1. A LOCAL command is used to create a message queue for the necessary modules. At the very least, the ssdl and GCSS7 (mapping it to module_id=0x4d) modules must be present in this section, e.g.

```
LOCAL      0x20      * ssdl - Board interface task
LOCAL      0x4d      * GCSS7
```

2. LOCAL commands are required for MTP3 and the user-part chosen, message queues. For instance, if running voice circuits in the context of the ISUP protocol:

```
LOCAL      0x22      * MTP3 module (and SS7LD 'mtp' and 'isup'
run-mode)
LOCAL      0x23      * ISUP module (and SS7LD 'isup' run-mode)
```

3. A REDIRECT command is used to cause messages destined to the on-board SW to be redirected to the ssdl as it provides the board's interface. So, for instance:

```
REDIRECT   0x71      0x20      * MTP2 module (except SS7HD boards)
REDIRECT   0x8e      0x20      * On-board management module
```

4. A FORK_PROCESS command is used to start up the ssdl process that interfaces with the board. This example assumes an appropriate license and debug mode:

```
FORK_PROCESS    ./ssdl -d -Lp
```

Note: When configuring the board to run at ISUP run level, the ssdl process forks the run-level protocols that run on the host, such as MTP3 and ISUP, so separate commands for those aren't required.

5. The GC SS7 subsystem needs to receive a one-shot API_MSG_CNF_IND message from the s7_mgt tool indicating its completion status; using an optional text file for its debug output, therefore we would have it setup like this for the default GC SS7 dlgs7srv service module ID (0x4d); in this particular case in debug mode and with log output to a file:

```
FORK_PROCESS    ./s7_mgt -d -i0x4d -fs7mgt.log
```

1.26.4.2 DSI Protocol Configuration File

SS7 LIU configuration specified in the DSI protocol configuration file must not be set for the DNIXXXTEPE2HMP; this is already provisioned by the DNI Line Administration (LineAdmin) Parameters. LineAdmin configuration is normally performed at board and HMP configuration time using the board's .config/.fcd file pair.

Refer to the *Dialogic® Host Media Processing Software for Linux Configuration Guide* for LineAdmin Parameters, Clear Channel configuration, and board configuration in general.

These are the minimum DSI protocol settings; it this does not provide a comprehensive list and other settings might also be used depending on the specific system configuration.

1. The Physical Interface Configuration Command, SS7_BOARD, is used to configure the DNIXXXTEPE2HMP board in the system.

Syntax

SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>

The <board_id> is maintained by the DSI SW and is a logical identity of the board in the range from 0 to one less than the number of boards supported.

The <board_type> must be set to the first seven characters in the board's name (e.g., DNI2410).

The <code_file> must be set to the full path for the board's FW file, in this case the hmp2_mixed.bin which is located in the directory indicated by the INTEL_DIALOGIC_FWL environment variable (default /usr/dialogic/data).

For instance, here are two Octal boards configured for ISUP run-mode:

```
SS7_BOARD 0 DNI2410 0x0000 /usr/dialogic/data/hmp2_mixed.bin ISUP
SS7_BOARD 1 DNI2410 0x0000 /usr/dialogic/data/hmp2_mixed.bin ISUP
```

It is important noting that the relative order of DNIXXXTEPE2HMP board enumeration within the DSI SW is not related to the HMP board Logical ID, nor does it relate to the HMP CLI Device ID.

Two issues arise from this loose association of DSI SS7_BOARD <board_id> and the HMP SW board enumeration:

- How to determine which physical LIU corresponds to each SS7_BOARD <board_id> if more than one DNIXXXTEPE2HMP and/or SS7LD are in the system; in particular if a mix of SS7 and other protocols (e.g., ISDN) is required. To that end, the user should choose the ssdl Switch address mode.

The Switch address mode (ssdl option "-o3") allows forcing ssdl board enumeration based on the board's ADDR switch setting (thumbwheel).

With this mode the ADDR list order determines the order of SS7_BOARD <board_id>(s).

Note: By default ssdl uses the PCI address mode instead.

- How to determine the mapping of MTP links and/or Voice circuits map to GC R4 devices on DNIXXXTEPE2HMP. This is accomplished automatically by the GC SS7 dlgcs7srv service which maps MTP links and Voice circuits with the board's bearer channels based on the board(s) serial number; that is all SS7_BOARD <board_id> are obtained their S/N using both HMP and DSI SW; their MTP links, if any, and Voice circuits are properly identified and mapped to the GC SS7 devices (dtiBnTm) accordingly. Refer to the [GC SS7 Library Considerations](#) section for more information on this mapping.

2. MTP (LINK, LINKSET, and ROUTE) as well as ISUP and TUP (CONFIG and CFG_CCTGRP) commands are supported, but don't differ from the documented settings for the SS7LDH4 interface board.

The User-Part protocol configuration, either ISDN or Telephony (TUP or ISUP respectively) must set their <user_id> parameters to the GC SS7 value; this would match the Module ID as entered in the system configuration file as per LOCAL command on the GC SS7. By default its module ID is 0x4d (GC SS7 configuration file parameter "Service.ModuleID").

3. XXXX_CFG_CCTGRP command requires special attention; this applies to ISUP and TUP protocols. When configuring the user-part group mask, one indicates a range of voice circuits in a Circuit Identification Code (CIC), i.e. a circuit group. There is always a temptation to attempt to associate the circuits with the underlying LIU timeslots, but at the DSI level, they aren't related.

Looking at the ISUP circuit group command:

```
ISUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options>
<user_inst> <user_id> <opc> <ssf> <variant> <options2>
```

Similarly for a TUP circuit group command:

```
TUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options>
<user_inst> <user_id> <opc> <ssf>
```

The least significant bit in the <cic_mask> represents the base CIC <base_cic> for that group, and not any underlying TDM E1/T1 bearer-channel timeslot.

While this User-Part protocol command does not imply any correlation between TDM timeslots and voice circuit on its own, that association is made by the SW that implements the user part, in this case the GC SS7 configuring the various parameters require careful consideration; in particular with having an appropriate CIC mask that results in the proper number of circuits, and avoids using underlying timeslots that aren't available for voice circuits, as for example TDM timeslots used for MTP signaling links. Refer to the [GC SS7 Configuration File](#) section for more information about this.

4. LIU_CONFIG command is not used for the DNIXXXTEPE2HMP combined environment since all LIU settings are taken care by the DNI Line Administration settings in its configuration file.

5. The SCBUS_LISTEN Command is not supported either in this environment.

1.26.4.3 GC SS7 Configuration File

1. Be sure to have the Type of System Configuration as follows; the rest of the configuration for this subsection follows the guidelines in the standard documentation.

```
System.Configuration = "Card"
```

2. It is recommended leaving the Module ID unchanged; otherwise SS7 System Configuration needs changes accordingly.

```
#Service.ModuleID = 0x4d
```

3. The MtpLink source and parameters are not relevant and not used in this SS7 combined environment.

4. The ClearGrp parameter is neither used nor relevant in this SS7 combined environment since the board supports Clear Channel circuits within itself and full HMP streaming obviating the necessity of this parameter.

5. Global Call SS7 Circuit Group are specified for ISUP and TUP User-Part protocols and are used to make the association between the voice circuits with the DNI bearer-channel devices which represent the underlying T1/E1 TDM timeslots where the voice circuits and/or SS7 MTP links are carried (e.g., dtiBnTm).

Note: The <trunk_name> could be any of "dkBx" or "dtiBy"; however the former only allows it for pure ISUP/TUP call control, and doesn't provide any streaming capabilities for voice circuits, so it's not a recommended configuration

- The CGrp <gid> must match the group id in the SS7 Protocol Config file (config.txt). There needs to exist one per CCTGRP group that will run in the context of the GC SS7. The <"trunk_name">, when using the "dtiBn" name is expected to be a valid DNI line interface that has been configured for Clear Channel.
- The association between the CIC and the underlying LIU timeslots is made by the GC SS7 SW; the GC SS7 SW parses the XXX_CFG_CCTGRP command in the Protocol configuration file and associates the underlying LIU timeslots to their corresponding CIC voice circuits for each <gid> based on the CGrp mapping.

The <cic_mask> entry per <gid> in each XXX_CFG_CCTGRP command is obtained and associated with the GC SS7 device, <"trunk_name"> for the CGrp with same <gid>. The optional [<base_TS>] can also be used as documented.

1.26.5 GC SS7 Library Considerations

When using the **gc_OpenEx()** with the with the protocol_name field set to "SS7" string (":P_SS7") as part of the GC line device naming convention, the resulting GC SS7 line device handle is used to access the SS7 signaling capabilities of Global Call. The network_device_name field must be based on the <"trunk_name"> in the CGrp lines in the gcss7.cfg file.

The GC SS7 SW limits access to voice circuits to those specified in CCTGRP <cic_mask> / <base_cid> for each of the CGrp lines. Bearer channels are formed based on that ISUP_CFG_CCTGRP tuple and the <base_TS> in the CGrp line, and associated to the voice circuits. Consequently only those bearer channels will be available for a GC SS7 application.

HMP creates GC Line devices for the DNIxxxxTEPE2HMP Network Interface Board with the dtiBn prefix in their names; each one ties to a physical line interface, i.e. LIUs, and their channels dtiBnTm representing the bearer channels, tie to the underlying physical TDM TSlots; this association is explained below:

Each User-Part voice circuit (ISUP/TUP) has a Circuit Identification Code, or CIC. The GC SS7 SW is responsible for associating each CIC to a physical TDM timeslot on specific LIUs. There is always a 1-to-1 association between a CIC and a TDM timeslot in that the first CIC maps directly to the first TDM TSlot in a given line, 2nd to 2nd, and so on and so forth, up to a maximum of 32 CICs. Furthermore if a CIC is excluded via a <cic_mask>, its corresponding TDM TSlot will not be made available to the application; however this TSlot is still physically available, but it's just bypassed by the SW.

With the DNIxxxxTEPE2HMP Network Interface Boards, the above is still true; however those TDM TSlots are mapped to logical bearer channels.

When using GC line devices with the dtiBn prefix, the "Tm" in the device name is directly associated with the underlying board's TSC bearer channel, known as the BChanId in the [TSC] defineBSet Parameter configuration. This BChanId maps to physical TSlots known as SlotId, in the same parameter configuration.

This has implications in the E1 case, as the default defineBSet configuration for Clear Channel maps the physical SlotId for TS16 to bearer channel BChanId 31, and SlotId TS17 through TS31 to BChanId(s) 16 through 30. This is done since traditionally signaling in E1 is carried in the TSlot 16 (TS16).

Therefore an application using the Global Call dtiBnTm channel devices should not make the assumption that there is a 1-to-1 association between a "Tm" in the dtiBnTm device and the CIC. The [TSC] defineBSet parameter can be re-configured as "1-to-1 for 31" if such E1 configuration is desired.

For instance, this is a possible setting in the case where LIUs one (1) and two (2) are configured to carry MTP links on TS16 each one. The default mapping is left untouched for those two lines; however the rest of the lines are used exclusively for voice circuits and in such case it might be desirable to have a "1-to-1 for 31" mapping instead. For instance to do so on network interfaces three to eight on a DNI2410TEPE2HMP board, one could configure it like this:

```
defineBSet=10,1,1,30,0,0,0,1,20,1, 1,1,3,15, 16,17,3,15, 0
defineBSet=20,2,1,30,0,0,0,1,20,1, 1,1,3,15, 16,17,3,15, 0
defineBSet=30,3,1,31,0,0,0,1,20,1, 1,1,3,31,0
defineBSet=40,4,1,31,0,0,0,1,20,1, 1,1,3,31,0
defineBSet=50,5,1,31,0,0,0,1,20,1, 1,1,3,31,0
defineBSet=60,6,1,31,0,0,0,1,20,1, 1,1,3,31,0
```

```
defineBSet=70,7,1,31,0,0,0,1,20,1, 1,1,3,31,0
defineBSet=80,8,1,31,0,0,0,1,20,1, 1,1,3,31,0
defineBSet=90,1,31,1, 0,0,0,1,21,1, 31, 16,3,1, 0,
defineBSet=100,2,31,1, 0,0,0,1,21,1, 31, 16,3,1, 0,
```

- Notes:**
1. Refer to the *Dialogic® Host Media Processing Software for Linux Configuration Guide* for more information on the [TSC] defineBSet parameter, and how to change the default BChanId mapping to SlotId.
 2. For this product, the association bearer channel to underlying TDM TSlot might not necessarily be the same as used with other GC SS7 network interface boards.

Since GC line devices using the dkBn prefix do not represent actual underlying board's TSC instances, they don't correlate with its BChanId, thus the above mapping is not applicable; the "dkBnTm" channels are enumerated and mapped sequentially to available circuits in the CCTGRP, as per its <cic_mask>.

To avoid having voice circuits stepping on a physical TSlot that carries an MTP link, the user should set the ISUP/TUP CCTGRP mask accordingly. For instance, it is typical to carry an MTP link on TS16 in the underlying E1 LIU; in which case a <cic_mask>=0x7fff7fff would exclude the TS16; when using dtiBn convention for <"trunk_name">, this timeslot normally (default defineBSet) maps to bearer-channel 31 in the GC SS7 SW, i.e. dtiBnT31.

When using dtiBn GC line devices, the GC SS7 library provides an additional level of protection against an improper <cic_mask> setting for LIU's that carry MTP links. The GC SS7 SW makes an attempt to locate all MTP links as set up in the DSI protocol configuration file with the MTP_LINK Command:

- The <stream> and <timeslot> parameters are obtained for each <board_id>.
- The <board_id> is matched with its corresponding physical DNI board in the HMP SW; next, its corresponding association between the board's line interface and the <stream> and the underlying physical timeslot with the <timeslot> last.
- Finally this bearer channel associated with the <board_id>, <stream>, and <timeslot> parameter tuple is checked against the defined <cic_mask>; if that channel was not excluded by it, the GC SS7 SW forces its exclusion by adding it to an internally maintained CIC mask in order to avoid accidental voice circuit collision with an MTP link.

Full HMP media streaming is supported via the standard **xx_Listen()** APIs; to take advantage of HMP media streaming for SS7 voice circuits the application will use the dtiBnTm channel devices that correspond to the voice circuit (bearer channel).

Note: dkBnTm doesn't provide any routing capabilities thus HMP streaming is not possible.

The standard **dt_listen()**, **dt_unlisten()**, and **dt_getxmitslot()**, as well as GC **gc_Listen()**, **gc_Unlisten()**, and **gc_GetXmitslot()** on dtiBnTm channel devices are supported for the purposes of routing of voice circuits.

1.26.6 Sample Configurations

Three sample configurations are provided as examples. In the three cases, the DSI SW is installed under /opt/DSI.

1.26.6.1 Sample Configuration 1

In this case two DNI2410TEPE2HMP are configured in E1 Clear Channel with each LIU carrying either MTP links or ISUP Voice circuits, thus maximizing board's utilization in an SS7 distributed environment.

Each LIU is connected to a remote SS7 destination.

```
*****
* Example System Configuration File (example_system.txt)
*****
* Essential modules running on host:
*
LOCAL          0x20          * ssds/ssdh/ssdm - Board interface task
LOCAL          0x00          * tim - Timer task
*
* Optional modules running on the host:
*
LOCAL          0xcf          * s7_mgt - Management/config task
LOCAL          0xef          * s7_log - Display and logging utility
LOCAL          0x3d          * Disstat
LOCAL          0x4d          * GCSS7
*
* Modules that optionally run on the host:
*
LOCAL          0x22          * MTP3 module (and SS7LD 'mtp' and 'isup' run-mode)
LOCAL          0x23          * ISUP module (and SS7LD 'isup' run-mode)
*
* Essential modules running on the board (all redirected via ssd):
*
REDIRECT       0x71      0x20  * MTP2 module (except SS7HD boards)
REDIRECT       0x8e      0x20  * On-board management module
*
* Modules that optionally run on the board (all redirected via ssd):
*
* Redirection of status indications:
*
REDIRECT       0xdf      0xef  * LIU/MTP2 status messages -> s7_log
*
DEFAULT_MODULE 0xef          * Redirect messages by default to module 0xef
*

* Dimensioning the Message Passing Environment:
*
* Now start-up all local tasks:
*   for SS7LD boards use ssdl
*
FORK_PROCESS   ./ssdl -d -Lp
FORK_PROCESS   ./tim
FORK_PROCESS   ./tick
FORK_PROCESS   ./s7_mgt -d
FORK_PROCESS   ./s7_log -flog7.txt
*****
```

In the Protocol configuration there is only one Linkset in this case for all four MTP Links configured. The MTP Links are carried on TS16, LIUs one and two (<stream>) of each of the DNI boards, thus for a total of 4 links.

Four of the ISUP circuit groups have their CIC masks to exclude one timeslot so that they are used to carry the MTP links.

The gcss7 configuration file: CGrp <gid> <"trunk_name"> [<base_TS> [<"Pref_SIU">]] creates the association between the user-part CFG_CCTGRP group lds <gid> in the Protocol configuration file. Except for the optional [<"Pref_SIU">], the rest of the fields are applicable to this product.

In this particular case since we know that TS16 on LIUs one and two of each of the boards carry MTP links, the bit that corresponds to a CIC that falls in the TS16 must be zeroed, in order to avoid using a voice circuit over an MTP Link.

In this case we will have the following mask for <gid>=0:

```
ISUP_CFG_CCTGRP 0 111 0x01 0x01 0x7fff7fff 0x001c 0 0x4d 222 0x8
0 0x00
```

Which excludes CIC=16 by means of the corresponding CGrp entry in the gcss7 config file for that <gid>:

```
CGrp 0 dtiB1

*****
* Example Protocol Configuration File                                     *
*****
* Configure individual boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
*
SS7_BOARD 0 DNI2410 0x0000 /usr/dialogic/data/hmp2_mixed.bin ISUP
SS7_BOARD 1 DNI2410 0x0000 /usr/dialogic/data/hmp2_mixed.bin ISUP

*
* MTP_CONFIG <reserved> <reserved> <options>
*
MTP_CONFIG 0 0 0x00040000
*
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc> <ssf>
*
MTP_LINKSET 0 111 4 0x0800 222 0x0008

*
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink> <stream> <timeslot>
<flags>
*
MTP_LINK 0x0 0 0 0 0 0x0 0 16 0x0006
MTP_LINK 0x1 0 1 1 0 0x1 1 16 0x0006
MTP_LINK 0x2 0 2 2 1 0x2 0 16 0x0006
MTP_LINK 0x3 0 3 3 1 0x3 1 16 0x0006

*
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
*
MTP_ROUTE 111 0 0x0020

*
```

```

* ISUP_CONFIG <reserved> <reserved> <user_id> <options> <num_grps> <num_cts>
ISUP_CONFIG 0 0 0x4d 0x0475 16 512

*
* ISUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options> <user_inst> <user_id>
<opc> <ssf> <variant> <options2>
ISUP_CFG_CCTGRP 0 111 0x01 0x01 0x7fff7fff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 1 111 0x21 0x21 0x7fff7fff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 2 111 0x41 0x41 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 3 111 0x61 0x61 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 4 111 0x81 0x81 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 5 111 0xa1 0xa1 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 6 111 0xc1 0xc1 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 7 111 0xe1 0xe1 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 8 111 0x101 0x101 0x7fff7fff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 9 111 0x121 0x121 0x7fff7fff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 10 111 0x141 0x141 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 11 111 0x161 0x161 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 12 111 0x181 0x181 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 13 111 0x1a1 0x1a1 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 14 111 0x1c1 0x1c1 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
ISUP_CFG_CCTGRP 15 111 0x1e1 0x1e1 0x7fffffff 0x001c 0 0x4d 222 0x8 0 0x00
*****

#####
#
# Global Call SS7 Configuration File (gcsc7.cfg)
#

# NOTE: all the entries and parameters are CASE SENSITIVE.

#####
# Type of System Configuration #
#####

# Leave commented out or set to "None" when not using Dialogic SS7.
# Depending on the value of this parameter, the sections below, that
# are specific to some configurations (SeptelCard, SIU, SIU.Dual, UserPart)
# will be used or not. The "UserPart" configuration is used for ISUP/TUP
# only configuration where lower layers are not of concern e.g. SIGTRAN
# configuration.
# Format: String - ["None", "Card", "SIU", "DualSIU", "UserPart"]
System.Configuration = "Card"

#####
# Parameters for the GlobalCall SS7 Call Control Library #
#####

# If defined, this parameter will cause the library logging to be
# activated at the first gc_Open() of an SS7 circuit and the trace
# file will have the specified name.
# Format: String
Library.LogFile = "ss7.log"

# Logging Level for the library
# Format: String - ["None", "Errors", "All"]
# Default: "Errors" (and Warnings)
Library.LogLevels = "All"

# Maximum size of the library log in kilobytes
# Format: Integer, Default: 200
Library.LogMaxSize = 2000

#####
# Parameters for the Dialogic SS7 service/daemon #
#####

```

```

# Logging Level for the service (DlgsS7.log)
# Format: String - ["None", "Errors", "All"]
# Default: "Errors" (and Warnings)
Service.LogLevels = "All"

# Maximum size of the service log in kilobytes
# Format: Integer, Default: 200
#Service.LogMaxSize = 2000

# Does the service need to start GCTLOAD automatically?
# Format: String - ["Yes", "No"]
Service.GCTLOAD_Control = "No" # "Yes"

# Path to GCTLOAD (Used only if GCTLOAD_Control is set to 'Yes')
# For Setpel Cards, the parameter defaults to the same path as ConfigDir
# Format: String
Service.GCTLOAD_Path = "c:\DSI"

# GCT-environment module id used by the service
# Format: Integer, Default: 0x4d
#Service.ModuleID = 0x4d

# Maximum timeout (in seconds) for server-application keep-alive mechanism
# Format: Integer; Default: 7; 0 means the mechanism is off (recommended for Windows)
#Service.WatchDogMaxTime = 0

#####
# Configuration for Septel Card Systems #
#####

# Path to the config.txt file
# Format: String
SeptelCard.ConfigDir = "c:\DSI"

# Should MTP links be automatically activated ?
# Format: String - ["None", "All"]
#SeptelCard.Auto_Links_Activation = "All"

#####
# Configuration for SIU Systems #
#####

# ID of this host - Use 0 if only one host accessing the SIU(s)
# Format: Integer
SIU.HostID = 0

# Type of File Transfer Protocol to use.
# Currently only FTP and SSH FTP (SFTP) is supported. The default is FTP.
# To use SSH FTP (SFTP), set this parameter to "SFTP"
# Format: String - ["SFTP", "FTP"]
#SIU.FTP_Type = "FTP"

# SIU A - IP Address
# Format: String

#SIU.A.IP_Address = "192.168.0.21"

# SIU A - Account to use to connect to SIU when using FTP
# Format: String
#SIU.A.FTP_Account = "siuftp"

# SIU A - Password for the FTP account
# Format: String
#SIU.A.FTP_Password = "siuftp"

# SIU A - Directory to which to change (in FTP session) in order to get config.txt
# Format: String

```

```

#SIU.A.RemoteConfigDir = "."

# Maximum time (in seconds) to wait at startup for an SIU to come on-line before
# considering it as being down.
# Format: Integer, Default: 10
#SIU.InitTimeout = 10

#####
# Parameters specific to Dual-Resilient SIU Configurations #
#####

# SIU B Parameters - See the same parameters for SIU.A
#SIU.B.IP_Address = "192.168.0.22"

#SIU.B.FTP_Account = "siuftp"

#SIU.B.FTP_Password = "siuftp"

#SIU.B.RemoteConfigDir = "."

# Maximum timeout (in seconds) for how long the service will keep calls in speech after
# control of a circuit group was transferred to other unit due to SIU/RSI/etc. failure
# Format: Integer; Default: 600 seconds; 0 means the feature is disabled
#SIU.Dual.TolerateCallTime = 20

#####
# Parameters that are related to config.txt #
#####

# MTP Link source - this can be specified in System Configuration = "Card" mode only
# "link_id" must match the values in config.txt
# "link_source" parameter must be one of the valid dti interfaces carrying SS7 signalling, for
ex.: "dtiB1T31"
# MtpLink <link_id> <"link_source">
#MtpLink 0 dtiB1T31

# Circuit Group configuration, Group ID must match the values in config.txt
# "trunk_name" could be any of "dkBx", "dtiBy" or "dumBz"
# "base_TS" optional parameter defaults to 1 if not set, it must be set if "Pref_SIU" is to be
specified
# "Pref_SIU" optional parameter can have "SIUA" or "SIUB" values only
#CGrp <gid> <"trunk_name"> [<base_TS> [<"Pref_SIU">]]
CGrp 0 dtiB1
CGrp 1 dtiB2
CGrp 2 dtiB3
CGrp 3 dtiB4
CGrp 4 dtiB5
CGrp 5 dtiB6
CGrp 6 dtiB7
CGrp 7 dtiB8
CGrp 8 dtiB9
CGrp 9 dtiB10
CGrp 10 dtiB11
CGrp 11 dtiB12
CGrp 12 dtiB13
CGrp 13 dtiB14
CGrp 14 dtiB15
CGrp 15 dtiB16

# Clear Channel Group configuration.
# The following fields must be specified for the new "ClearGrp" parameter:
# "trunk_name" could be any of "dkBx", "dtiBy" or "dumBz"
# <ts_mask> - this is the timeslot mask that represents all the timeslots to be used on this
trunk
# ClearGrp <"trunk_name"> <ts_mask>
# e.g ClearGrp dkB1 0x7fffffff
#ClearGrp dtiB1 0x7fffffff

```

```
# For the detailed description of available options and
# other parameters refer to Global Call for SS7 Technology Guide.

#
# End of gcss7.cfg
#
```

1.26.6.2 Sample Configuration 2

In this case one DNI2410TEPE2HMP is configured in E1 Clear Channel, back to back, with each LIU carrying either MTP links or TUP Voice circuits; only four TUP circuit groups are configured in this example. The GC SS7 configuration file Circuit Group configuration, CGrp, bypasses the first bearer channel of each line device, which is done with the <base_TS>=2.

Also, note that even though in this example the <cic_mask> does not have a TS16 exclusion for the MTP links in TS16 in LIU one and, the GC SS7 library is capable of discovering this fact and excluding its corresponding devices (dtiB1T31, and dtiB2T31).

The TUP protocol does not run embedded within the ssdl binary and needs to be run separately and requires separate license. In this case ssdl run mode is set to MTP2; in this particular case MTP3 and TUP are running with the trial-mode license (-t) as a matter of an example, though proper separate licenses would normally be required.

```
*****
* Example System Configuration File
*****

* Essential modules running on host:
*
LOCAL          0x20          * ssds/ssdh/ssdm - Board interface task
LOCAL          0x00          * tim - Timer task
*
* Optional modules running on the host:
*
LOCAL          0xcf          * s7_mgt - Management/config task
LOCAL          0xef          * s7_log - Display and logging utility
LOCAL          0x4d          * GCSS7
*
* Modules that optionally run on the host:
*
LOCAL          0x22          * MTP3 module
LOCAL          0x4a          * TUP module
*
* Essential modules running on the board (all redirected via ssd):
*
REDIRECT       0x71 0x20      * MTP2 module (except SS7HD boards)
REDIRECT       0x8e 0x20      * On-board management module
*
* Modules that optionally run on the board (all redirected via ssd):
*
* Redirection of status indications:
*
REDIRECT       0xdf 0xef      * LIU/MTP2 status messages -> s7_log
*
DEFAULT_MODULE 0xef          * Redirect messages by default to module 0xef
*
* Dimensioning the Message Passing Environment:
*
* Now start-up all local tasks:
```

```

*   for SS7LD boards use ssdl
*
FORK_PROCESS    ./ssdl -d -Lp
FORK_PROCESS    ./tim
FORK_PROCESS    ./tick
FORK_PROCESS    ./s7_mgt -d
FORK_PROCESS    ./s7_log -flog7.txt
FORK_PROCESS    /opt/DSI/HSTBIN/mtp3 -t
FORK_PROCESS    /opt/DSI/HSTBIN/tup -t -m0x4a

*****
* Example Protocol Configuration File
*****
* Configure individual boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
*
SS7_BOARD 0 DNI2410 0x0000 /usr/dialogic/data/hmp2_mixed.bin MTP2

*
* MTP Parameters:
* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG 0 0 0x00040000
*
* Two linksets are defined. Each has a unique ID. Point
* codes given for the local and adjacent linkset point at one
* another.
*
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags>
* <local_spc> <ssf>
MTP_LINKSET 0 900 1 0x0800 959 0x0008
MTP_LINKSET 1 959 1 0x0800 900 0x0008

*
* Two signaling links are defined, one in each linkset.
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink>
* <stream> <timeslot> <flags>
*
MTP_LINK 0x0 0 0 0 0 0x0 0 16 0x0006
MTP_LINK 0x1 1 0 0 0 0x1 1 16 0x0006

*
* Define a route for each remote signaling point. For linksets
* 900/959, send to SIGTRAN message queue (ID 0x20)
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
*
MTP_ROUTE 900 0 0x0020
MTP_ROUTE 959 1 0x0020
*
* TUP parameters:
* Configure TUP module:
* TUP_CONFIG <reserved> <reserved> <user_id> <options> <num_grps> <num_cts>
TUP_CONFIG 0 0 0x4d 0x8166 4 128
*
* Define TUP circuit groups:
* TUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options>
* <user_inst> <user_id> <opc> <ssf>
TUP_CFG_CCTGRP 0 900 0x01 0x00 0x7fffffff 0x0030 0 0x4d 959 0x8
TUP_CFG_CCTGRP 2 900 0x21 0x20 0x7fffffff 0x0030 0 0x4d 959 0x8

TUP_CFG_CCTGRP 1 959 0x01 0x40 0x7fffffff 0x0030 0 0x4d 900 0x8
TUP_CFG_CCTGRP 3 959 0x21 0x60 0x7fffffff 0x0030 0 0x4d 900 0x8

```

The GC SS7 configuration file is similar to the previous sample; the only change is in the CGrp parameters as follows; again, note the <base_TS> of 2:

```
# Circuit Group configuration, Group ID must match the values in config.txt
# "trunk_name" could be any of "dkBx", "dtiBy" or "dumBz"
# "base_TS" optional parameter defaults to 1 if not set, it must be set if "Pref_SIU" is to be
specified
# "Pref_SIU" optional parameter can have "SIUA" or "SIUB" values only
# CGrp <gid> <"trunk_name"> [<base_TS> [<"Pref_SIU">]]
CGrp 0 dtiB1 2 CGrp 1 dtiB2 2 CGrp 2 dtiB3 2 CGrp 3 dtiB4 2
```

1.26.6.3 Sample Configuration 3

In this example one DNI2410TEPE2HMP is configured in E1 Clear Channel, back to back, with each LIU carrying MTP links and/or ISUP Voice circuits. An SS7LD is also configured in the same system and under the same configuration back-to-back as well.

The SS7LD board is not cabled to the DNI board with the SyncRoute cable since the former terminates its LIUs at the board and there is no need to propagate the TDM clock reference from one to another board.

```
*****
* Example System Configuration File                                     *
*****
* The following lines identify the various modules, running on this
* system, used in this SIGTRAN configuration.
*
LOCAL 0x20 * ssdl - Board interface task
LOCAL 0x00 * Timer Task
LOCAL 0xcf * s7_mgt - Management/config task
LOCAL 0xef * s7_log - Display and logging utility
*LOCAL 0xc2 * mbm - Management task
*LOCAL 0xd0 * SCTPD (daemon) module
*LOCAL 0xd1 * SCTP module
*LOCAL 0xc1 * M2PA module

*
* Modules that optionally run on the host:
*
LOCAL 0x22 * MTP3 module (and SS7LD 'mtp' and 'isup' run-mode)
LOCAL 0x23 * ISUP module (and SS7LD 'isup' run-mode)
*LOCAL 0x4a * TUP module
LOCAL 0x4d * GCSS7 (GlobalCall interface) module

*
* Essential modules running on the board (all redirected via ssd):
*
REDIRECT 0x71 0x20 * MTP2 module (except SS7HD boards)
REDIRECT 0x8e 0x20 * On-board management module
REDIRECT 0x10 0x20 * CT bus/Clocking control module
* Redirection of status indications:
*
REDIRECT 0xdf 0xef * LIU/MTP2 status messages -> s7_log

DEFAULT_MODULE 0xef * Redirect messages by default to module 0xef
*
* Dimensioning the Message Passing Environment:
*
NUM_MSGS      5000      * Number of standard size
*                      * messages in the environment
*NUM_LMSGs     200      * Number of 'long' messages
*                      * (used for certain TCAP based applications)
*
* Start-up commands for all local tasks. Note that
* LINUXDEV_DIR will be replaced by the installation script
* by the full path needed to reach the executables - for example
```



```

* "/root/Desktop/SS7/linux_dev"
*
* Processes started with "-t" are running in trial mode - they will
* automatically halt after 10 hours and must be restarted
*
FORK_PROCESS /opt/DSI/tim_lnx
FORK_PROCESS /opt/DSI/tick_lnx
FORK_PROCESS /opt/DSI/ssdl -d -Lp
*FORK_PROCESS /opt/DSI/ssdl -d -t -o3 -a1
*FORK_PROCESS /opt/DSI/ssdl -d -t
*FORK_PROCESS /opt/DSI/sctpd
*FORK_PROCESS /opt/DSI/sctp
*FORK_PROCESS /opt/DSI/m2pa_lnx6 -t
*FORK_PROCESS /opt/DSI/mtp_lnx6 -t
*FORK_PROCESS /opt/DSI/mbm -d
*FORK_PROCESS /opt/DSI/isp_lnx6 -t
FORK_PROCESS /opt/DSI/s7_mgt -d -i0x4d
*FORK_PROCESS /opt/DSI/s7_mgt -i0x4d
FORK_PROCESS /opt/DSI/s7_log -fss7.log -m0xef
*FORK_PROCESS /opt/DSI/HSTBIN/mtp3 -t
*FORK_PROCESS /opt/DSI/HSTBIN/tup -t -m0x4a

```

Because the SS7LD is initialized by the gctload, the LIU_CONFIG parameters are required in the protocol configuration file. Both boards are configured for ISUP user-part protocol.

```

*****
* Example Protocol Configuration File *
*****
* Configure individual boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
SS7_BOARD 0 SS7LD 0x0001 ./DC/ss7.dc7 ISUP
SS7_BOARD 1 DNI2410 0x0000 /usr/dialogic/data/hmp2_mixed.bin ISUP
*
* MTP Parameters:
* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG 0 0 0x00040000

TRACE_MOD_ID 0xef * Set default trace module to 0xef.
*
* Configure individual T1/E1 interfaces:
* LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format>
* <crc_mode> [<build_out>]
* For Opal only as Gemini LIU config is done by TSP
LIU_CONFIG 0 0 5 1 1 1
LIU_CONFIG 0 1 5 1 1 1
LIU_CONFIG 0 2 5 1 1 1
LIU_CONFIG 0 3 5 1 1 1
*
* Two linksets are defined. Each has a unique ID. Point
* codes given for the local and adjacent linkset point at one
* another.
*
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags>
* <local_spc> <ssf>
* combined
MTP_LINKSET 0 900 2 0x0800 959 0x0008
MTP_LINKSET 1 959 2 0x0800 900 0x0008
*
* Two signaling links are defined, one in each linkset.
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink>
* <stream> <timeslot> <flags>
*

```

```

MTP_LINK 0x0 0 0 0 0 0x0 0 16 0x0006
MTP_LINK 0x1 1 0 0 0 0x1 1 16 0x0006

MTP_LINK 0x2 0 1 1 1 0x0 0 16 0x0006
MTP_LINK 0x3 1 1 1 1 0x1 1 16 0x0006
*
* Define a route for each remote signaling point. For linksets
* 900/959, send to SIGTRAN message queue (ID 0x20)
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
* combined
MTP_ROUTE 900 0 0x0020
MTP_ROUTE 959 1 0x0020

*
* ISUP parameters:
* Configure ISUP module. Since SS7 is run under GlobalCall,
* 0x4d is always reserved for GC SS7. (msg queue address for GC)
* ISUP_CONFIG <reserved> <reserved> <user_id> <options> <num_grps>
* <num_ccts> [<partner_id>]
*
ISUP_CONFIG 0 0 0x4d 0x0475 12 384

*
* Configure ISUP circuit groups. Number of groups same as number of
* links. 900/959 relates back to linksets/links.
* Number of circuits is arbitrary, we only use 2 to test.
* Destination Point Code (DPC) for 1st group is 900, Origination
* Point Code (OPC) is 959.
* Base Circuit Identification Code (CIC) - starting circuit number for
* that group. CIC_mask - 7fff masked with circuits in the circuit
* group - if 0, circuit used for signaling, if 1 clear channel for
* bearer.
* ISUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask>
* <options> <user_inst> <user_id> <opc> <ssf> <variant> <options2>
*
ISUP_CFG_CCTGRP 0 900 0x01 0x00 0x7fff7fff 0x001c 0 0x4d 959 0x8 0 0x00
ISUP_CFG_CCTGRP 2 900 0x21 0x20 0x7fffffff 0x001c 0 0x4d 959 0x8 0 0x00

ISUP_CFG_CCTGRP 4 900 0x41 0x40 0x7fff7fff 0x001c 0 0x4d 959 0x8 0 0x00
ISUP_CFG_CCTGRP 6 900 0x61 0x60 0x7fffffff 0x001c 0 0x4d 959 0x8 0 0x00
ISUP_CFG_CCTGRP 8 900 0x81 0x80 0x7fffffff 0x001c 0 0x4d 959 0x8 0 0x00
ISUP_CFG_CCTGRP 10 900 0xa1 0xa0 0x7fffffff 0x001c 0 0x4d 959 0x8 0 0x00

ISUP_CFG_CCTGRP 1 959 0x01 0xc0 0x7fff7fff 0x001c 0 0x4d 900 0x8 0 0x00
ISUP_CFG_CCTGRP 3 959 0x21 0xe0 0x7fffffff 0x001c 0 0x4d 900 0x8 0 0x00

ISUP_CFG_CCTGRP 5 959 0x41 0x100 0x7fff7fff 0x001c 0 0x4d 900 0x8 0 0x00
ISUP_CFG_CCTGRP 7 959 0x61 0x120 0x7fffffff 0x001c 0 0x4d 900 0x8 0 0x00
ISUP_CFG_CCTGRP 9 959 0x81 0x140 0x7fffffff 0x001c 0 0x4d 900 0x8 0 0x00
ISUP_CFG_CCTGRP 11 959 0xa1 0x160 0x7fffffff 0x001c 0 0x4d 900 0x8 0 0x00

```

The SS7LD GC SS7 devices are of the dkBn type, while the ones that correspond to the DNI board are given the dtiBn name type.

```

#####@@@SOFT@@@WARE@@@COPY@@@RIGHT@@@#####
#
# Global Call SS7 Configuration File (gcss7.cfg)
#

# NOTE: all the entries and parameters are CASE SENSITIVE.

#####
# Type of System Configuration #
#####
# Leave commented out or set to "None" when not using Dialogic SS7.
# Depending on the value of this parameter, the sections below, that

```

```

# are specific to some configurations (SeptelCard, SIU, SIU.Dual, UserPart)
# will be used or not. The "UserPart" configuration is used for ISUP/TUP
# only configuration where lower layers are not of concern e.g. SIGTRAN
# configuration.
# Format: String - ["None", "Card", "SIU", "DualSIU", "UserPart"]
System.Configuration = "Card"

#####
# Parameters for the GlobalCall SS7 Call Control Library #
#####

# If defined, this parameter will cause the library logging to be
# activated at the first gc_Open() of an SS7 circuit and the trace
# file will have the specified name.
# Format: String
Library.LogFile = "ss7.log"

# Logging Level for the library
# Format: String - ["None", "Errors", "All"]
# Default: "Errors" (and Warnings)
Library.LogLevels = "All"

# Maximum size of the library log in kilobytes
# Format: Integer, Default: 200
#Library.LogMaxSize = 2000

#####
# Parameters for the Dialogic SS7 service/daemon #
#####

# Logging Level for the service (Dlgs7.log)
# Format: String - ["None", "Errors", "All"]
# Default: "Errors" (and Warnings)
Service.LogLevels = "All"

# Maximum size of the service log in kilobytes
# Format: Integer, Default: 200
#Service.LogMaxSize = 2000

# Does the service need to start GCTLOAD automatically?
# Format: String - ["Yes", "No"]
Service.GCTLOAD_Control = "Yes"

# Path to GCTLOAD (Used only if GCTLOAD_Control is set to 'Yes')
# For Setpel Cards, the parameter defaults to the same path as ConfigDir
# Format: String
Service.GCTLOAD_Path = "/opt/DSI"

# GCT-environment module id used by the service
# Format: Integer, Default: 0x4d
#Service.ModuleID = 0x4d

# Maximum timeout (in seconds) for server-application keep-alive mechanism
# Format: Integer; Default: 7; 0 means the mechanism is off (recommended for Windows)
#Service.WatchDogMaxTime = 0

#####
# Configuration for Septel Card Systems #
#####

# Path to the config.txt file
# Format: String
SeptelCard.ConfigDir = "/opt/DSI"

# Should MTP links be automatically activated ?
# Format: String - ["None", "All"]
#SeptelCard.Auto_Links_Activation = "All"

```

```
#####
# Parameters that are related to config.txt #
#####

# MTP Link source - this can be specified in System Configuration = "Card" mode only
# "link_id" must match the values in config.txt
# "link_source" parameter must be one of the valid dti interfaces carrying SS7 signalling, for
ex.: "dtiB1T31"
# MtpLink <link_id> <"link_source">
#MtpLink 0 dtiB1T31
#MtpLink 1 dtiB2T31
#MtpLink 2 dtiB3T31
#MtpLink 3 dtiB4T31

# Circuit Group configuration, Group ID must match the values in config.txt
# "trunk_name" could be any of "dkBx", "dtiBy" or "dumBz"
# "base_TS" optional parameter defaults to 1 if not set, it must be set if "Pref_SIU" is to be
specified
# "Pref_SIU" optional parameter can have "SIUA" or "SIUB" values only
#CGrp <gid> <"trunk_name"> [<base_TS> [<"Pref_SIU">]]
CGrp 0 dkB1
CGrp 1 dkB2
CGrp 2 dkB3
CGrp 3 dkB4
CGrp 4 dtiB1
CGrp 5 dtiB2
CGrp 6 dtiB3
CGrp 7 dtiB4
CGrp 8 dtiB5
CGrp 9 dtiB6
CGrp 10 dtiB7
CGrp 11 dtiB8

# Clear Channel Group configuration.
# The following fields must be specified for the new "ClearGrp" parameter:
# "trunk_name" could be any of "dkBx", "dtiBy" or "dumBz"
# <ts_mask> - this is the timeslot mask that represents all the timeslots to be used on this
trunk
# ClearGrp <"trunk_name"> <ts_mask>
# e.g ClearGrp dkB1 0x7fffffff
#ClearGrp dtiB1 0x7fffffff

# For the detailed description of available options and
# other parameters please refer to Global Call for SS7 Technology Guide.

#
# End of gcss7.cfg
#
```

1.27 SIP Session Timer Modifications

The MSML Media Server has been updated to resolve several issues related to handling of SIP Session timers. The previous version of the media server includes the Supported and Session-Expires header fields in the 2xx response, but did not refresh the call session using re-INVITE messages when "refresher=uas" is specified.

With Service Update 151, if the caller / AS doesn't specify the support of SIP session timers by including the Supported header field with the option tag "timer", the media server will assume responsibility of session refreshes and send a re-INVITE for sessions that are connected for longer than the default session expiry of 1 hour. Refer to the <http://www.ietf.org/rfc/rfc4028.txt> for additional information regarding use of Session timers in SIP.

1.28 Line Loopback Control on DNI2410AMCTEHMP Board

Service Update 141 provides the ability to control Transceiver Local and Line Interface Unit (LIU) Local, and Remote loopback modes using the standard Digital Network Software API.

An application can set and get transceiver local, and line interface unit local/remote loopback mode using the **dt_setparm()** and **dt_getparm()** respectively. The APIs must pass a board device handle as their first argument, obtained from the logical board device, i.e. **dtiBn**, which represents a network interface. The device parameter define must be set to DTG_SETBDMD; its value argument must point to the address of an integer containing one of the value defines in this table:

#Define	Value	Description
DTG_SETBDMD	Set/get mode value.	
DTMD_NORMAL	DTMD_NORMAL	Normal mode (default)
DTMD_XCVRLB	DTMD_XCVRLB	Transceiver unit local loopback mode (used for digital network interface testing). Also known as Payload Loopback mode. The Network receiving line and transmitting line are connected without termination.
DTMD_LIULLB	DTMD_LIULLB	Line interface unit local loopback mode (used for digital network interface testing). Also known as Line Loopback mode. The Network receiving line and transmitting line are connected without termination.
DTMD_LIURLB	DTMD_LIURLB	Line interface unit remote loopback mode (used by network for network testing). Also known as Diagnostic Digital Loopback mode where signals generated by the network are looped back as they had been received from the network.

The **dt_setparm()** and **dt_getparm()** do not have a mode parameter; however since the operation takes some finite time to complete, the application should not assume it completed successfully until the loopback mode is confirmed via an SRL event. When using the Digital Network Interface API, use the standard run-time (SRL) event interfaces which will signal operation completion by overloading the **DTEV_RETDIAG** (Diagnostics Complete event), as follows:

Event Management Function	Digital Network Interface-Specific Input	Value
sr_getevtdev() Get event device	Event device	Logical board device handle obtained from the logical board device
sr_getevttype() Get event type	Event type	DTEV_RETDIAG
sr_getevtdatap() Get pointer to event data	Event data pointer should be cast to unsigned long	DTMD_NORMAL DTMD_XCVRLB DTMD_LIULLB DTMD_LIURLB

- Notes:**
1. The feature does not apply on a logical channel handle.
 2. Refer to the *Dialogic® Digital Network Interface Software Reference* for more information.
 3. While a network interface logical board device is in any of the loopback modes (other than Normal mode), the line is considered to be Out of Service (OOS). While the network interface is OOS its logical channel devices may not make or receive calls.

Support for Global Call Alarm Management System (GCAMS) is also provided as a means for signaling operation completion via the following GCAMS alarm numbers:

DTT1_LOOPBACK_CFA

Diagnostic mode on the (T1) line trunk

DTT1_LOOPBACK_CFAOK

Diagnostic mode on the (T1) line trunk recovered

DTE1_LOOPBACK_CFA

Diagnostic mode on the (E1) line trunk

DTE1_LOOPBACK_CFAOK

Diagnostic mode on the (E1) line trunk recovered

- Notes:**
1. Refer to the *Dialogic® Global Call API Library Reference* for GCAMS details.
 2. The GCAMS alarm number will be reported on the Global Call Line device obtained from a **gc_OpenEx()** on the logical board device **dtiBn**.

Example

```
{
    ...
    LINEDEV ldev;
    if (gc_OpenEx(&ldev, ":N_dtiB1", EV_SYNC, 0) != GC_SUCCESS)
    {
        printf("Error in gc_OpenEx for board dtiB1\n");
        return -1;
    }

    int ndev;
    if (gc_GetResourceH(ldev, &ndev, GC_NETWORKDEVICE) != GC_SUCCESS)
    {
        printf("Error in gc_GetResourceH for board dtiB1\n");
        return -1;
    }

    //
    // GCAMS
    //
    if (gc_SetAlarmNotifyAll(ldev, ALARM_SOURCE_ID_NETWORK_ID, ALARM_NOTIFY) != GC_SUCCESS)
    {
        printf("Error in gc_SetAlarmNotifyAll on the dtiB1\n");
        return -1;
    }

    int value = DTMD_LIURLB;
    if (dt_setparm(ndev, DTG_SETBDMD, (void *) &value) == -1)
    {
        printf("Error in dt_setparm(0x%x) on %s: %d - %s\n", value, ATDV_NAMEP(ndev),
            ATDV_LASTERR(ndev),
            ATDV_ERRMSGP(ndev));

        return -1;
    }

    //
    // Set logical board device (network interface) LIU to remote loopback mode
    //
    printf("dt_setparm(DTG_SETBDMD, 0x%x) on %s issued\n", value, ATDV_NAMEP(ndev));

    METAEVENT metaevent;
    //
    // Wait for completion event
    //
    for (;;)
    {
        if (sr_waitevt(500) < 0) break;
        if (gc_GetMetaEvent(&metaevent) != GC_SUCCESS)
        {
            printf("Error in gc_GetMetaEvent\n");
            return 1;
        }

        if (metaevent.evtttype == DTEV_RETDIAG && !(metaevent.flags & GCME_GC_EVENT))
        {
            unsigned long * mode = (unsigned long *) metaevent.evtdatap;
            printf("Received DTEV_RETDIAG on %s with value %d\n", ATDV_NAMEP(sr_getevtdev()),
                *mode);
        }

        if (metaevent.evtttype == GCEV_ALARM && (metaevent.flags & GCME_GC_EVENT))
        {
            long alarm_number = 0;
            char* alarm_name;
        }
    }
}
```

```

        if (gc_AlarmNumber(&metaevent, &alarm_number) != GC_SUCCESS)
        {
            printf("Error in gc_AlarmNumber, device %s\n", ATDV_NAMEP(sr_getevtddev()) );
            return -1;
        }

        if (gc_AlarmName(&metaevent, &alarm_name) != GC_SUCCESS)
        {
            printf("Error in gc_AlarmNumber, device %s\n", ATDV_NAMEP(sr_getevtddev()) );
            return -1;
        }

        if (alarm_number == DTE1_LOOPBACK_CFA || alarm_number == DTE1_LOOPBACK_CFAOK)
        {
            printf("Diagnostic mode on %s: %s, 0x%x\n", ATDV_NAMEP(sr_getevtddev()),
                alarm_name, alarm_number);
        }
    }
}

value = -1;
if (dt_getparm(ndev, DTG_SETBDMD, (void *) &value) == -1)
{
    printf("Error in dt_getparm() on %s: %d - %s\n", ATDV_NAMEP(ndev), ATDV_LASTERR(ndev),
        ATDV_ERRMSGP(ndev));

    return -1;
}

printf("dt_getparm(DTG_SETBDMD) on %s is 0x%x\n", ATDV_NAMEP(ndev), value);

//
// Perform Network Test
//
...
//
// Restore logical board device (network interface) LIU to normal mode
//
int value = DTMD_NORMAL;
if (dt_setparm(ndev, DTG_SETBDMD, (void *) &value) == -1)
{
    printf("Error in dt_setparm(0x%x) on %s: %d - %s\n", value, ATDV_NAMEP(ndev),
        ATDV_LASTERR(ndev),
        ATDV_ERRMSGP(ndev));

    return -1;
}

//
// Wait for completion event
//
...
gc_Close(ldev);
return 0;
}

```


1.29 Updated Operating System Distributions

Service Update 129 announces support for the following Red Hat Enterprise Linux (RHEL) and Community ENTERprise Operating System (CentOS) versions in both 32-bit and 64-bit:

- Red Hat 5 update 7 or greater
- CentOS 5 update 7 or greater
- Red Hat 6 update 2 or greater
- CentOS 6 update 2 or greater

1.30 Standalone Licensing Server

Service Update 129 introduces the Dialogic® Standalone License Server (SLS). This server licensing model, or concurrent/floating licensing, is the first step in the licensing migration process from the current node-locked certificate model to a concurrent model that supports resource sharing. This initial release is based on a Client/Server model where the license server normally resides on a separate platform from Client license machines.

The SLS changes the existing licensing model such that licenses are held and maintained centrally by a License Server. HMP clients are granted rights to access and use resources from a designated resource pool controlled by the License Server. HMP connects across a TCP/IP network and sends requests to checkout any number of features that are available from the SLS.

For information about installing and using the SLS, refer to the *Dialogic® Standalone License Server User's Guide* which has been added to the documentation bookshelf.

1.31 Enhanced Nb UP Transcode Support

Service Update 129 allows an IP media streaming (IPM) device to stream AMR-NB audio, AMR-WB audio and G.711 audio over Nb UP. This data can be streamed to and from a 3G network. A multimedia (MM) device or a voice device (DX) can be connected to the IPM device for play and record operations.

Setting up AMR-NB, AMR_WB or G.711 over Nb UP media sessions is similar to setting up a 3G-324M over Nb UP session; however, the 3G-324M (M3G) component is not used.

For information and guidelines for streaming AMR-NB audio, AMR-WB audio or G.711 audio over Nb UP, refer to the *Dialogic® IP Media Guide API Programming Guide and Library Reference*. For information on the Dialogic® 3G-324M API, refer to the *Dialogic® 3G-324M API Programming Guide and Library Reference*.

Note: Using the AMR-NB or the AMR-WB resource in connection with one or more Dialogic products mentioned herein does not grant the right to practice the AMR-NB or the AMR-WB standard. To seek a patent license agreement to practice the standard, contact the VoiceAge Corporation at www.voiceage.com/licensing.php.

1.32 MSML Native Hairpinning Support

Service Update 129 adds support for MSML native hairpinning. This feature adds the ability to create a low latency, low overhead, native hair-pin connection between two network connection (conn) objects by instructing the media server to <join> the two conn objects via a <stream> using compressed media. Such a connection is formed by supporting the new <stream> element attribute, compressed.

- compressed
This attribute specifies whether the stream uses compressed media. It supports both audio and video streams.
Valid values:
 - true
 - false (default)

This enhancement allows MSML applications to take advantage of HMP media engine capabilities that already exist and are accessible to applications via the R4/GC API.

1.32.1 Compressed Attribute Guidelines

The attribute compressed is a <stream> attribute used when joining streams between conn objects. The following guidelines apply to its use:

- Upon specifying the compressed attribute with its value as “true” when forming such a stream, both conn objects must use the same media format or the media server will return the MSML error response code of 450, objects have incompatible media formats.
- Do not use the compressed attribute when streams are joined to conference (conf) objects. If the attribute compressed is used when forming such a stream, the media server will return a MSML error response code of 407, attribute not supported.
- An existing stream between two conn objects that is currently using transcoding (uncompressed media) can be modified to become a native connection by using the <modifystream> element with the attribute compressed set to “true”.
- An existing stream between two conn objects that is currently a native hair-pin connection can be modified to use transcoding (uncompressed media) by using the <modifystream> element with the attribute compressed set to “false”.

1.32.2 Use Cases

The following use case section describes the various operations and behaviors expected when using the <stream> element along with the attribute compressed.

Use Case 1: Forming a full duplex native audio hair-pin connection between two network connection objects conn:caller_1 and conn:caller_2

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" compressed="true"/>
  </join>
</msml>
```

Use Case 2: Forming a half duplex native audio hair-pin connection from network connection object conn:caller_1 to network connection object conn:caller_2

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" dir="from-id1" compressed="true"/>
  </join>
</msml>
```

Use Case 3: Forming a half duplex native audio hair-pin connection from network connection object conn:caller_1 to network connection object conn:caller_2 while network connection object conn:caller_1 also has a full duplex connection to a conference, conf:1234

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conf:1234"/>
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" dir="from-id1" compressed="true"/>
  </join>
</msml>
```

Use Case 4: Forming a full duplex native multimedia hair-pin connection between between network connection objects conn:caller_1 and conn:caller_2

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" compressed="true"/>
    <stream media="video" compressed="true"/>
  </join>
</msml>
```

Use Case 5: Forming a half duplex native multimedia hair-pin connection from network connection object conn:caller_1 to network connection object conn:caller_2

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" dir="from-id1" compressed="true"/>
    <stream media="video" dir="from-id1" compressed="true"/>
  </join>
</msml>
```

Use Case 6: Forming a half duplex native audio hair-pin connection from network connection object conn:caller_1 to network connection object conn:caller_2 while network connection object conn:caller_1 also has a full duplex connection to a multimedia conference, conf:1234

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conf:1234">
    <stream media="audio"/>
    <stream media="video" dir="from-id1" display="region1"/>
    <stream media="video" dir="to-id1"/>
  </join>
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" dir="from-id1" compressed="true"/>
  </join>
</msml>
```

Use Case 7: Forming a half duplex native multimedia hair-pin connection from network connection object conn:caller_1 to network connection object conn:caller_2 while network connection object conn:caller_1 also has a full duplex connection to a multimedia conference, conf:1234

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <join id1="conn:caller_1" id2=" conf:1234">
    <stream media="audio"/>
    <stream media="video" dir="from-id1" display="region1"/>
    <stream media="video" dir="to-id1"/>
  </join>
  <join id1="conn:caller_1" id2=" conn:caller_2">
    <stream media="audio" dir="from-id1" compressed="true"/>
    <stream media="video" dir="from-id1" compressed="true"/>
  </join>
</msml>
```

For more information about MSML, refer to the *Dialogic® MSML Media Server Software User's Guide* located on the documentation bookshelf.

1.33 720p Resolution Support

Service Update 129 adds support for 720p HD video.

Note: Dialogic® PowerMedia™ HMP for Linux Release 4.1 currently outputs 720p at up to 15 frames per second. Setting the resolution to a higher value has no effect on the output since it is limited to 15 fps.

1.34 MSML <var> Element Support

Service Update 118 supports the MSML <var> element. The <var> element specifies the generation of audio using prerecorded audio segments that are selected and dynamically played in sequence based upon a specified variable.

For a complete description of the <var> element and its variables, refer to the *Dialogic® MSML Media Server Software User's Guide*.

1.35 Improved Audio Conferencing Support

Service Update 118 improves audio conferencing support for playing audio into a conference. This improvement supports use cases where playing music or a sound track into a conference is required.

1.35.1 Feature Description

The application can now specify the conferencing party attributes ECNF_PARTY_ATTR_PRIVILEGE (R4/Global Call) or preferred (MSML) with a new value that will result in the audio of that party to always be summed independent of the speech detection algorithm.

The value for the MSML attribute preferred is "true_enhanced". For detailed information, refer to the *Dialogic® MSML Media Server Software User's Guide*.

This MSML attribute is supported by setting the CNF API party attribute ECNF_PARTY_ATTR_PRIVILEGE using the **cnf_SetAttributes()** function. The new value for ECNF_PARTY_ATTR_PRIVILEGE is

ECNF_ATTR_STATE_ENABLED_ENHANCED. The following table shows the CNF party attribute mappings to support the MSML preferred attribute.

MSML preferred attribute values	CNF Party Attribute ECNF_PARTY_ATTR_PRIVILEGE
true	CNF_ATTR_STATE_ENABLED
false (default)	CNF_ATTR_STATE_DISABLED
true_enhanced	CNF_ATTR_STATE_ENABLED_ENHANCED

Note: The feature is provided via the Conferencing (CNF) API library and applies when using the MCX board device only. For more information about this feature, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.35.2 Examples

Setting <stream> attribute preferred = true_enhanced using <join>:

```
<msml version="1.1">
  <join id1="conn:jd87dfg4h" id2="conf:exampleConf">
    <stream media="audio" dir="from-id1" preferred="true_enhanced"/>
    <stream media="audio" dir="to-id1"/>
  </join>
</msml>
```

Setting <stream> attribute preferred = true_enhanced using <modifystream>:

```
<msml version="1.1">
  <modifystream id1="conn:jd87dfg4h" id2="conf:exampleConf">
    <stream media="audio" dir="from-id1" preferred="true_enhanced"/>
  </modifystream>
</msml>
```

1.36 Echo Cancellation Support for MSML

Service Update 118 adds the ability to enable or disable echo cancellation for participants that are joined in a conference.

1.36.1 Feature Description

The feature is provided via the MSML Media Server to assist in removing echo.

A new MSML <stream> attribute, dlgc:echo_cancel, is added to support this capability. This MSML attribute will be supported by setting the Conferencing API conference attribute, ECNF_CONF_ATTR_ECHO_CANCEL using the **cnf_SetAttributes()** function. For information about this attribute, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

For detailed information about MSML support, refer to the *Dialogic® MSML Media Server Software User's Guide*.

1.36.2 Use Cases

The following Use Cases show how to enable/disable echo cancellation for participants that are joined in a conference.

Use Case 1: Setting <stream> attribute dlgc:echo_cancel = enable using <join>:

```
<msml version="1.1">
  <join id1="conn:jd87dfg4h" id2="conf:exampleConf">
    <stream media="audio" dir="from-id1" dlgc:echo_cancel = "enable" />
    <stream media="audio" dir="to-id1" />
  </join>
</msml>
```

Use Case 2: Setting <stream> attribute dlgc:echo_cancel = enable using <modifystream>:

```
<msml version="1.1">
  <modifystream id1="conn:jd87dfg4h" id2="conf:exampleConf">
    <stream media="audio" dir="from-id1" dlgc:echo_cancel = "enable" />
  </modifystream>
</msml>
```

Use Case 3: Setting <stream> attribute dlgc:echo_cancel = disable using <modifystream>:

```
<msml version="1.1">
  <modifystream id1="conn:jd87dfg4h" id2="conf:exampleConf">
    <stream media="audio" dir="from-id1" dlgc:echo_cancel = "disable" />
  </modifystream>
</msml>
```

1.37 Multiple NIC Support using MSML for RTP

Service Update 118 adds support for multiple RTP interfaces for redundancy and load balancing with MSML.

HMP currently dedicates a single NIC for RTP traffic using the CLI command “conf system hmp-rtp-address”. With this feature, more than one NIC for RTP can be dedicated through system configuration. If one NIC is down, one of the remaining NICs will be used.

For details related to this feature, refer to the “media_server.xml Configuration File” section in *Dialogic® MSML Media Server Software User's Guide*.

1.38 Additional Single and Dual DNI Support

Service Update 118 announces support for the Dialogic® DNI310TEPE2HMP and the Dialogic® DNI610TEPE2HMP Digital Network Interface boards. Support was added in Service Update 108. Refer to the [Support for Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP Boards](#) section.

1.39 64-bit SSP Support

With Service Update 115, the Dialogic® PowerMedia™ HMP for Linux Release 4.1 media engine is a native 64-bit process when running on a 64-bit platform. This change results in a more efficient use of system memory as the media engine is no longer constrained by the memory limitations of a 32-bit process. Performance is further enhanced with the elimination of 32-bit to 64-bit system call conversions, as well as the utilization of 64-bit instructions and registers for media processing.

Note: Customer application requirements have not changed. Customer applications must continue to be 32-bit processes since 64-bit customer applications are not yet supported.

1.40 Red Hat Enterprise Linux Release 5 Update 6 Support

Service Update 108 announces support for Red Hat Enterprise Linux Release 5 Update 6, both 32-bit and 64-bit versions.

1.41 Support for Party Type in MSML Conferencing

Service Update 108 adds coach and pupil support to the MSML Media Server product to enhance the product capabilities with HMP software.

1.41.1 New MSML Attributes

This feature adds the ability to specify a “conference party type” using MSML for participants that are joined to a conference. Two new MSML attributes are added to allow a conference party type of standard, coach, pupil, or privileged to be specified.

These attributes, “dlgc:conf_party_type” and “preferred”, are **<stream>** attributes specifically for audio streams that are formed when joining participants to a conference. These attributes *may* be used for an audio stream that is an input to a conference and cannot be used for other streams.

This enhancement allows MSML applications to take advantage of HMP media engine capabilities that already exist and are accessible to applications via the R4/Global Call API.

dlgc:conf_party_type

Specifies the conference party type. Two selected parties can establish a private communication link within the overall conference. The coach is a private member of the conference and is only heard by the pupil. However, the pupil cannot speak privately with the coach. Valid values are “coach”, “pupil”, or “standard” (default).

preferred

Specifies if the stream will always be mixed and audible to conference participants or whether it will need to contend for N-loudest mixing. Valid values are “true” or “false” (default).

true

The stream **MUST** always be mixed. This means that party’s input, providing its speech level is greater than zero, is always included in the output summation process along with the loudest remaining “Standard/Pupil” parties within the active talker limit defined for the conference.

false

The stream **MAY** contend for mixing into a conference when N-loudest mixing is enabled.

For details about these attributes, refer to the *Dialogic® MSML Media Server User’s Guide*.

1.42 Early MSML Connection Identifier Support

Service Update 108 modifies the creation time for the MSML Connection Identifier for a call. This allows the Connection Identifier to exist when the media server sends the SIP 200 OK message for that call.

1.42.1 Feature Implementation

MSML supports creation of an early connection by supporting the **<join>** command on the SIP dialog of the existing call leg (inbound call), or on any other existing SIP dialog. Dialogic® HMP MSML Media Server product does not currently support this capability. In order for the **<join>** command to be successfully accepted and executed, the connection identifier for the new call leg must exist at the time the **<join>** command is asserted. Currently, that connection identifier is only created when the SIP ACK message is received. Therefore, the media server currently rejects the command with a 480 “Object does not exist” response.

MSML connection objects are created when media sessions get established through SIP. The connection object associated with a SIP call **MUST** have an identifier equivalent to the local tag assigned by the media server to identify the SIP dialog. This is the tag the media server adds to the SIP “To” header in its response to an initial INVITE transaction. RFC 3261 requires the tag values be globally unique. For example:

```
conn:74jgd63956ts
```

where the value “74jgd63956ts” is the tag from the SIP “To” header.

For an incoming SIP call, SIP header information, including the “To” line and “From” line, is accessible to an R4/Global Call application via the Global Call APIs when the GCEV_OFFERED event is received by the application. This event corresponds to the arrival of the SIP INVITE. The function [SIP “To tag” for Inbound Calls](#) adds the capability for the application to obtain the “To tag” closely following the reception of the GCEV_ACCEPT event following the assertion of the **gc_AcceptCall()** function and sending a SIP 180 Ringing (or 183 Session Progress) message to the remote end.

For more details about this feature, refer to the *Dialogic® MSML Media Server Software User's Guide*.

1.43 Loudest Talker Attribute Support for MSML

Service Update 108, adds the ability to query the current <n-loudest> setting for a conference as specified by the MSML Audit Conference Package.

1.43.1 Feature Description

The feature is provided via the MSML Media Server and applies to using the MCX board device for conferencing.

The <n-loudest> element specifies the maximum number of conference participants that will be summed as part of the audio mix at any time. Participants to be mixed are determined by audio energy levels.

The <n-loudest> element is a child of the <audiomix> element. Its attribute, “n”, is a mandatory attribute that specifies the number of participants as mentioned above. Supported values are integers from 2 to 10.

When the <n-loudest> element is included when creating or modifying a conference, the maximum number of conference participants that will be summed as part of the audio mix at any time will be equivalent to the default for the media server. This default may be set via configuration options provided for the media server.

This MSML attribute will be supported by setting the Conferencing API conference attribute, ECNF_CONF_ATTR_MAX_ACTIVE_TALKERS using the **cnf_SetAttributes()** function. For information about this attribute, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

For detailed information about the MSML Audit Conference Package and the <n-loudest> element, refer to the *Dialogic® MSML Media Server Software User's Guide*.

1.43.2 Use Cases

Use Case 1: Creating conferences specifying a different number “n” for each conference. For this example use case, two conferences are created as follows:

```

<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <createconference name="conf_1234" >
    <audiomix>
      <n-loudest n="3" />
      <asn ri="1s" />
    </audiomix>
  </createconference>
</msml>

<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <createconference name="conf_5678" >
    <audiomix>
      <n-loudest n="2" />
      <asn ri="1s" />
    </audiomix>
  </createconference>
</msml>

```

With a minimum of four parties joined to each conference, and with all parties actively speaking at the same time; only three parties will be summed for conference “conf_1234” and only two parties will be summed for conference “conf_5678”.

Use Case 2: Creating a conference with one value for <n-loudest> and then modifying a conference to use a different value. For this example use case, a conference is created as follows:

```

<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <createconference name="conf_1234" >
    <audiomix>
      <n-loudest n="3" />
      <asn ri="1s" />
    </audiomix>
  </createconference>
</msml>

```

The conference is then modified to use a different value for <n-loudest>:

```

<?xml version="1.0" encoding="UTF-8"?>
<msml version="1.1">
  <modifyconference name="conf_1234" >
    <audiomix>
      <n-loudest n="2" />
      <asn ri="1s" />
    </audiomix>
  </modifyconference>
</msml>

```

With a minimum of four parties joined to the conference, and with all parties actively speaking at the same time; only three parties will be summed for the original conference created and only two parties will be summed after the conference is modified.

1.44 Mute/Un-mute Audio Support for MSML

Service Update 108 adds mute audio attribute support to the MSML Media Server product to enhance the product capabilities with HMP software.

1.44.1 Audio Stream Attribute

This feature allows the application to mute or un-mute an audio stream flowing into a conference or out of a conference. This capability is provided the **amt** attribute of the **<gain>** element which is a child of the **<stream>** element. The **<gain>** element specifies the gain characteristics applied to an audio stream, including the ability to mute and un-mute the stream. In addition to supporting RFC 5707, the **amt** attribute values specified in dB and the values “mute” and “un-mute” are also supported. The indicated streams may be muted or un-muted as part of **<modifystream>** and/or **<join>** operations.

For more information about the **<gain>** element, refer to the *Dialogic® MSML Media Server User's Guide*.

1.44.2 HMP Media Processing Engine Enhancements

This feature is also implemented at the HMP R4/Global Call API level, providing a programmatic interface to allow applications to mute and un-mute the audio stream transmitted from a conference to a conferencing party so that when muted, that party will not hear the other conferencing participants. When muted, silence will be transmitted from the conference for that party.

Note: The feature only applies when using the MCX board device. There is no mute/un-mute support for the CNF board device at this time.

A new party attribute, **ECNF_PARTY_ATTR_TXMUTE**, provides the ability to mute and un-mute a party. The values are **ECNF_ATTR_STATE_ENABLED** or **ECNF_ATTR_STATE_DISABLED**.

When **ECNF_PARTY_ATTR_TXMUTE** is set to **ECNF_ATTR_STATE_ENABLED** for party X, party X will no longer be capable of hearing other conference participants. When a muted party is un-muted by setting **ECNF_PARTY_ATTR_TXMUTE** for that party to **ECNF_ATTR_STATE_DISABLED**, that party is once again capable of hearing other participants in the conference.

For more information about the Conferencing APIs, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.45 Wildcard Identifiers Support for MSML

Service Update 108 adds wildcard identifiers support to the MSML Media Server product to enhance the product capabilities with HMP software.

1.45.1 Identifiers and Wildcards

Identifiers allow every object in a media server to be uniquely addressed. They can also be used to refer to multiple objects. An identifier can reference multiple objects when a wildcard is used as an instance name. Wildcards are used as an identifier or as part of the instance names referenced by the identifier attributes id1 or id2 of the **<unjoin>** element. Adding wildcards as identifiers for the **<unjoin>** element allows media streams related to a specific instance of an object (connection or conference) to be terminated without specifying the other object or objects at the other end of the streams.

Note: Streams terminated using the **<unjoin>** element are only streams between objects that can be formed using the **<join>** element. Streams created using the **<dialogstart>** element cannot be terminated using the **<unjoin>** element.

MSML reserves the instance name composed of a single asterisk (*) to mean all objects have the same identifier root and class. Instance names containing an asterisk cannot be created. Wildcards **MUST** only be used as the right-most term of an identifier and **MUST NOT** be used as part of the root for dialog identifiers.

Examples of valid identifiers using wildcards:

```
conf:abc/dialog:*  
conn:*
```

Example of invalid wildcard usage:

```
conf:*/dialog:73849
```

For more information about wildcard identifiers, refer to the *Dialogic® MSML Media Server Software User's Guide*.

1.46 VMware ESXi 5.0 Support

Service Update 108 adds ESXi 5.0 support to the list of VMware products supported by Dialogic® PowerMedia™ HMP for Linux Release 4.1, both 32-bit and 64-bit versions, and IP-only support with Red Hat Enterprise Linux Release 5.0 with Update 6. Refer to the [Virtualization Support](#) section for HMP configuration and other relevant information details which are still applicable to this product. Also, refer to the VMware ESXi 5.0 documentation at <http://www.vmware.com/support/pubs>.

1.47 SIP “To tag” for Inbound Calls

Service Update 108 provides a SIP “To tag” for inbound calls for both first party call control (1PCC) and third party call control (3PCC) operating modes.

Note: For outbound call support, all SIP header information can be obtained from the event data when the GCEV_ALERTING event is received by the application.

1.47.1 Feature Implementation

Currently, HMP does not allow an application to obtain the “To tag” until the call is connected and the GCEV_ANSWERED event occurs. With this feature, the application can retrieve a SIP “To tag” using the **gc_GetCallInfo()** function shortly after the GCEV_ACCEPT event is received, and prior to the call being connected. If the application answers a call immediately after receiving the GCEV_OFFERED, the SIP “To tag” will be available after the application calls the **gc_AnswerCall()** function and send a SIP 200 OK message to the remote end.

A new parmID IPEXTID_GETCALLINFOUPDATE in the existing setID IPSET_CONFIG is introduced for the support of this feature at the HMP board level. The feature is enabled using the **gc_SetConfigData()** function and the following GC_PARM_BLK:

IPSET_CONFIG setID
IPPARM_GETCALLINFOUPDATE parmID
And one of the following (int) values:

- GCPV_ENABLE
- GCPV_DISABLE

Because a small time window exists between when the GCEV_ACCEPT event is returned and when the updated call information is available, a separate event is provided. For this reason, as well as supporting the case when the application answers the call immediately, when the Global Call API library’s SIP stack is ready to provide the “To tag” information, a GCEV_EXTENSION of type IPEXTID_GETCALLINFOUPDATE will be sent to the application. This extension ID is used to notify the application that call information retrievable using the **gc_GetCallInfo()** function has been updated.

Note: Currently, enabling “To tag” retrieval is only supported at the HMP board level. Once enabled, all channels will be enabled.

The application can use the **gc_GetCallInfo()** API to retrieve the DESTINATION_ADDRESS_SIP, the ORIGINATION_ADDRESS_SIP, and the IP_CALLID (if desired). The DESTINATION_ADDRESS_SIP at this time will also include the “To tag”.

For example, when the application accepts the call using **gc_AcceptCall()**, the following information is obtained by calling the **gc_GetCallInfo()** function after receipt of the GCEV_EXTENSION event:

- DESTINATION_ADDRESS_SIP
(sip:ms-ivr@64.52.111.192;tag=969b6d0-0-13c4-50022-2826d-2f0e8b34-2826d)
- ORIGINATION_ADDRESS_SIP
(sip:60000027@64.52.111.182;tag=a0016671)
- IP_CALLID

The same information is obtained using the **gc_GetCallInfo()** function after receipt of the GCEV_EXTENSION event in the case the application answers the call immediately using the **gc_AnswerCall()** function after it receives the GCEV_OFFERED event.

Refer to the [Code Examples](#) section below for details. For more information about **gc_GetCallInfo()** variances for IP, refer to the *Dialogic® Global Call IP Technology Guide*.

1.47.2 Code Examples

The following code example shows how to enable “To tag” retrieval:

```
parmblkp = NULL;

gc_util_insert_parm_val(&parmblkp,
                        IPSET_CONFIG,
                        IPPARM_GETCALLINFOUPDATE,
                        sizeof(int),
                        GCPV_ENABLE);

if (gc_SetConfigData( GCTGT_CCLIB_NETIF,
                    bdev,
                    parmblkp,
                    0,
                    GCUUPDATE_IMMEDIATE,
                    &request_id,
                    EV_ASYNC) != GC_SUCCESS)
{
    sprintf(str, "Enabling send_totag has failed\n");
    printandlog(index, GC_APICALL, NULL, str, 0);
}
else
{
    sprintf(str, "gc_SetConfigData(linedev=%ld) Success ", bdev);
    printandlog(index, GC_APICALL, NULL, str, 0);
}
```

The following code example shows how to retrieve “To tag” information using the **gc_GetCallInfo()** function once an extension event is received:

```
// process GCEV_EXTENSION event to check if extID is IPEXTID_GETCALLINFOUPDATE
// get SIP Msg and SIP Msg Info
int OnExtensionEvent(METAEVENT *metaeventp)
{
    EXTENSIONEVTBLK *pExtensionBlock = NULL;
    unsigned char extID;
    pExtensionBlock = (EXTENSIONEVTBLK*)(metaeventp->extevtdatap);
    extID = pExtensionBlock->ext_id;

    printf("extID : %x\n", extID);

    if(extID == IPEXTID_GETCALLINFOUPDATE)
        return 1;
    else
        return 0;
}

if(evttype == GCEV_EXTENSION)
{
    ToTagQueryable = OnExtensionEvent(metaeventp);
    if(ToTagQueryable)
    {
        /* Call gc_getcallinfo here */

        sprintf(str, "!!!! Received GCEV_extension in offered state");
        printandlog(index, GC_APICALL, NULL, str, 0);
        /* get to party number + to tag */
        if (gc_GetCallInfo(pline->call[callindex].crn, DESTINATION_ADDRESS_SIP, ToAddr) ==
```

```

        GC_SUCCESS) {
            sprintf(str, "!!!!!! gc_GetCallInfo(crn=0x%lx) Success in OFFERED state = %s",
                pline->call[callindex].crn, ToAddr);
        } else {
            sprintf(str, "!!!!!! gc_GetCallInfo(crn=0x%lx) Failure - pline
                ->call[callindex].crn);
        }
        printandlog(pline->index, GC_APICALL, NULL, str, 0);
    }
}

```

1.48 IPv6 Call Control Support

Service Update 103 adds Internet Protocol Version 6 (IPv6) call control support for open media connections. This represents the second phase of IPv6 support. The initial phase is described in [IPv6 Support](#).

1.48.1 Data Structure Updates

To implement this feature, updates were made to the Global Call IP API library data structures IP_ADDR and IP_VIRTBOARD. These updates are provided in the Documentation Updates chapter, [Section 3.4.11, “Dialogic® Global Call IP Technology Guide”](#), on page 257.

1.48.2 SIP IPv6 SDP Configuration Parameters

The following table shows how the SIP protocol can be set to support IPv6 addressing in the SDP offer/answer model. The default value is IPv4 addressing for backward compatibility. The **gc_SetUserInfo()** function can be used to specify arguments for single call [GC_SINGLECALL] or all calls [GC_ALLCALLS] related to a line device. The **gc_SetConfigData()** function is not used to set those parameters.

Table 1-1. Global Call IPv6 SDP Configuration

Set	Set ID	Parameter ID	Value	Description
gc_SetUserInfo()	IPSET_SDP	IPPARM_SDP_IP_TYPE	USE_IPv4	(Default value) Only IPv4 addressing will be accepted in incoming/outgoing SDP.
gc_SetUserInfo()	IPSET_SDP	IPPARM_SDP_IP_TYPE	USE_IPv6	Only IPv6 addressing will be accepted in incoming SDP.
gc_SetUserInfo()	IPSET_SDP	IPPARM_SDP_IP_TYPE	PREFER_IPv6	In this mode IPv6 addressing will be used when sending an SDP offer. However when receiving an SDP offer based on IPv4, the SIP stack will adapt itself and use IPv4 SDP for that connection.

Note: Both the USE_IPv6 and PREFER_IPv6 values require a valid local media IPv6 address to be configured using the “conf system HMP” command (refer to the example below).

Example

```
telnet localhost
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
+-----+
|
| .....
| :: Dialogic(R) Host Media Processing Software Release 4.1 LIN ::
| `.....'
|
|
| For HELP:
|   Use '?' at command prompt
|   Use '-h' within commands
|
+-----+

Login :admin
Password :*****
CLI> conf system HMP 2001:db8:0:f101::2
updated
updated
CLI> show system
UNKNOWN RMS System
Name:          System Name Not Defined
Location:      System Location Not Defined
Contact:       System Contact Not Defined

Operational Status:    System is ready
Run Status:            Active
Media Last Request:    Start
Media Start Mode:      Manual
Software Media Support: Refer to license details

Uptime:                2 week(s) 1 day(s) 21 hour(s) 30 minute(s) 54 second(s)
Last Boot Reason:      Power on
Services:              7
Form Factor:           RMSModel:          Dialogic(R) Host Media Processing Software
Release 4.1 LIN
Vendor:                Dialogic Research Inc.
Serial Number:         Undefined
HMP RTP IPv4 Address:  10.130.4.141
HMP RTP IPv6 Address:  2001:db8:0:f101::2
```

1.49 Higher Resolution Video Support

Service Update 100 adds Video Graphics Array (VGA) 640x480 resolution support to the Dialogic® PowerMedia™ HMP for Linux Release 4.1 media engine.

1.49.1 Feature Description

This feature allows applications to support higher resolution video in SIP Video scenarios at equivalent VGA and/or 4CIF (Common Interchange Format) resolutions. It also introduces the concept of “Video Levels of Experience” to support the next group of video experience beyond CIF resolution. VGA and 4CIF resolution video fall into Experience Level 2, where resolutions up to CIF are covered by Experience Level 1. Prior to this feature, HMP supported only Experience Level 1.

The system functionality covered by Experience Level 2 includes video scenarios such as video streaming, native play and record, transcoded play and record, video transcoding, transrating, transizing, and video mixing as provided through the video conferencing (mcx) resource. Refer to the [Feature Scenarios](#) section for feature behavior within these scenarios.

1.49.2 Video Levels of Experience

In order to promote licensing and system level accounting of resources required per channel for VGA/4CIF on Dialogic® HMP software, Dialogic uses different Levels of Experience used per channel. The Level of Experience will be licensed beyond a standard video transcoding channel to enable the customer to achieve the next level of higher resolution video on the video channel.

These Levels of Experience are designed to provide a boundary to the capabilities of an HMP real-time video channel, while providing reasonable expectation for customers. The Levels of Experience concept is intended to correspond to an approximate standard H.264 Baseline Profile Levels given an HMP bit rate constraint. The goal is to provide industry standard resolutions at a maximum of 30 frames per second and to provide a designated channel maximum bit rate. In the event that a non-standard resolution is requested by a customer, each Level of Experience is approximated in the average number of 16x16 macroblocks that describe a single image frame. This field should be used to determine if the target resolution fits into a particular experience level.

The following table provides a breakdown of HMP Levels of Experience. This feature covers VGA/4CIF resolutions described by HMP Experience Level 2. The current HMP capability is covered by HMP Experience Level 1. HMP Experience Levels 3 and 4, beyond the VGA/4CIF resolution experience, are here as a placeholder for future reference.

Table 1-2. HMP Levels of Experience

HMP Level of Experience	Video Channel LevelFormat	Approx max Macroblocks (16x16) /image	Corresponding H.264 Baseline Level	Channel Max Bit Rate	Max Resolutions covered
Level 1	QCIF/CIF	Up to 400 MB	Up to Level 1.3	Up to 768 Kbps	SQCIF(128x96) @ 30 fps QCIF (176x144) @ 30 fps QVGA (320x240) @ 30 fps CIF (352x288) @ 30 fps
Level 2	VGA/4CIF	Up to 1600 MB	Up to Level 3.1	Up to 2 Mbps	WQVGA @ 30 fps SQVGA[iPhone] (400x300) @ 30 fps W288p (512x288) @ 30 fps nHD[iPhone/ipad](640x360) @ 30 fps 448p (576x448) @ 30 fps VGA (640x480) @ 30 fps W448p (768x448) @ 30 fps SD (720x480) @ 30 fps 4CIF (704x576) @ 30 fps
Level 3 (Future)	HD720p	Up to 4000 MB	Up to Level 4.0	Up to 4 Mbps	HD720p @ 30 fps
Level 4 (Future)	HD1080p	Up to 9000 MB	Up to Level 4.0	Up to 10 Mbps	HD1080p @ 30 fps

1.49.3 Licensing

Three new licenses have been added to support the VGA/4CIF resolution experience level. These licenses show capabilities and do not result in additional created devices.

VLE2-MM: # = channels supported by multimedia (MM) at Level 2.

- Requires multimedia licenses \geq the number of VLE2-MM licenses. Multimedia licenses support multimedia play and record which includes support for video. VLE2-MM permits the multimedia device to support play and record operation for files containing video resolutions up to Level 2. The VLE2-MM licenses are shared across multimedia devices, consumed when used, and then released. When released, they are available to be reused by any multimedia device.
- This license allows native play and record at Level 2.
- When coupled with VLE2-XCode, plus appropriate video coder licenses, such as H.264, allows play and record at Level 2 with transcoding. This only applies for video coders for which HMP supports up to this level of resolution.

VLE2-XCode: # = channels of transcoding at Level 2.

- Requires video transcoding licenses \geq the number of VLE2-XCode licenses. This is for video coders for which we support up to this level of resolution.
- Supports RTP to RTP and RTP to M3G transcoding up to Level 2.
- For play and record at Level 2 with transcoding description, see VLE2-MM description.
- For video conferencing resolutions up to level 2, see VLE2-Conf description.

VLE2-Conf: # = number of conferences supporting up to a Level 2 conferencing root frame size.

- This license is at the conference level, not the party level.
- Requires video conferencing licenses per video party.
- Requires VLE2-XCode for conference parties at Level 2 resolution along with the appropriate video transcoding licenses.

1.49.4 Supported Video Codec

This feature covers resolutions up to VGA/4CIF for the H.264 Baseline profile codec only. The specific supported resolutions for H.264 encoding are covered by the supported resolution sizes defined in the *videodefs.h* file.

1.49.5 Feature Scenarios

The following scenarios are provided to demonstrate feature behavior:

Video Play (Multimedia device and IP Media device)

An application can play video files that conform to Experience Level 2 (VGA/4CIF) using a Multimedia device connected to an IP Media device. This can be done both natively and with transcoding using the H.264 codec. An application can also play still images up to VGA/4CIF resolutions. It is recommended that applications fill in the proper video level, resolution (height and width) and file bit rate in the mm play structure to allocate internal video buffers accordingly. File types played by the Multimedia device are covered in the supported file format section.

The application can instruct HMP to transmit an H.264 video RTP stream using the IP Media device up to the VGA/4CIF resolutions as described in the Experience level 2. In order to encode an H.264 output RTP stream at Experience Level 2 (VGA/4CIF) resolution, the application must specify the appropriate H264 profile, level and output bit rate, along with the desired video resolution for width and height in the ipm structure LOCAL codec settings. The supported resolutions for H.264 encoding are covered by the supported resolution sizes defined in the *videodefs.h* file. For more information about the IP Media API, refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

Video Record (Multimedia device and IP Media device)

An application can record video that conforms to Experience Level 2 (VGA/4CIF) to files using a Multimedia device connected to an IP Media device. This can be done both natively and with transcoding using the H.264 codec. An application can also capture still images up to VGA/4CIF resolutions. Files recorded by the Multimedia device are in recorded in dmf (.vid/.aud) format and the default max compressed framesize will be two seconds worth of bit rate at the resolution set by the multimedia record structure. A utility, such as the hmp3gp offline utility, will be required to convert from a recorded file to .3gp or .mp4 format if that file storage format is desired.

The application can instruct HMP to receive an H.264 video RTP stream that conforms to Experience Level 2 (VGA/4CIF) from an RTP video endpoint using an IP Media device. In order to decode an H.264 input RTP stream up to Experience Level 2 (VGA/4CIF) resolution, the application must set the ipm structure REMOTE codec setting to the maximum H.264 Level supported by Experience level 2 along with the expected resolution width and height. The system decoder will prepare to decode based on the REMOTE codec settings of the IP Media device, but will try to handle any video resolution up to the maximum supported Experience level 2. The H.264 in-band DCI, when available in the incoming video stream, will supersede the REMOTE codec settings and will be used to configure the decoder for the incoming video stream. Selecting the correct H.264 level for the REMOTE codec settings is critical to insure that adequate resource allocation occurs to handle the incoming stream.

Video Streaming (IP Media device)

This feature allows HMP to receive an H.264 RTP video stream that conforms to Experience Level 2 (VGA/4CIF) natively or with transcoding. Two or more IP Media devices can be connected together allowing a video stream received on one IPM device to be streamed out one or more other IP Media devices. Refer to the above scenarios (video play and video record) for transmitting and receiving video streams at Experience Level 2 (VGA/4CIF).

Video Conferencing (Conferencing devices and IP Media devices)

This feature allows multimedia conferencing to be supported with video conforming up to Experience Level 2 (VGA/4CIF). Conference layouts up to VGA/4CIF are supported. Mixing video from multiple parties using multiple codecs (H.264, H.263, and MPEG-4) at different resolutions are supported. Parties using Experience Level 2 (VGA/4CIF) must be H.264 parties. Refer to the above scenarios (video play and video record) for transmitting and receiving video streams at Experience Level 2 (VGA/4CIF).

Note: Conferencing requires transcoding.

1.49.6 Feature Limitations

The following limitations apply to this feature:

- This feature is limited to video resolutions covered by Experience Level 2 (VGA/4CIF) only.

- The user application may choose to ignore the Experience Levels and provide a video file or video stream that does not conform to H.264 profile, level, bit rate, or resolution requirements. Dialogic® HMP software may attempt to deal with video that does not conform to the specified characteristics the best it can or it may give up transcoding or streaming and report an error to the application. In both cases, the application will receive enough information through errors and/or warnings to understand how the capability of the channel has been exceeded.

1.50 Support for Dialogic® DSI SS7LD Network Interface Board

Service Update 100 provides support for the Dialogic® DSI SS7LDH4Q Network Interface Board with Global Call SS7.

1.50.1 Configuration Restrictions

Although the Dialogic® DSI SS7LDH4Q board configuration is similar to other SS7 boards, the following restrictions apply:

- The "GCTLOAD_Control" parameter in the *gcss7.cfg* file must be set to "Yes". Manual mode for the Dialogic® DSI SS7LDH4Q board is currently not supported. In addition, the "GCTLOAD_PATH" parameter must be set to the location of the SS7 DSI software. For example:

```
# Does the service need to start GCTLOAD automatically?
# Format: String - ["Yes", "No"]
Service.GCTLOAD_Control = "Yes"

# Path to GCTLOAD (Used only if GCTLOAD_Control is set to 'Yes')
# For Setpel Cards, the parameter defaults to the same path as
ConfigDir
# Format: String
Service.GCTLOAD_Path = "/usr/septel"
```

- The SW1 board instance ID number should be passed to the "-a" option for the *ssdl* process in the *system.txt* file. For example, if the ID on the Dialogic® DNI board is 0 and the Dialogic® DSI SS7LDH4Q board was set to 1, then the following options should be passed to *ssdl*:

```
FORK_PROCESS    ./ssdl -d -t -o3 -a1
```

- The Dialogic® DSI SS7LDH4Q board takes about 50 to 60 seconds to complete the download (i.e., a "Boot Complete" occurs). The following is displayed on the screen when *dlstart* completes:

```
...
Starting CLI Agent Service :      [ OK ]

STARTING DIALOGIC MEDIA
. . . . .
. . . . .
Dialogic Media is ACTIVE
```

```
Continue GC/SS7 Service: ... OK
STARTING DIALOGIC SERVICES COMPLETE.
```

Note: Even though “Continue GC/SS7 Service: ... OK” is displayed, GCTLOAD is still actively downloading the board. To be sure the GCTLOAD download completes, wait for approximately one minute before starting the application. To verify completion, check the *DlgcS7.log* file to see if the following message, API_MSG_CNF_IND, which indicates that GCTLOAD is finished downloading, is shown:

```
11:10 57.716      DkRuntime::WaitGCT()
11:11 53.665    <-- DkRuntime::ProcessMsg() API_MSG_CNF_IND Configuration status indication
                  received
11:11 53.665      DkRuntime::WaitGCT() returns true
```

1.50.2 System CT Bus (TDM) Clocking with Mixed Configurations

In most mixed configurations, the Dialogic® DSI SS7LDH4Q board is electrically isolated from the other boards using the CT Bus. This is because the Dialogic® DSI SS7LDH4Q board terminates signaling only from its own line interfaces, and has no capability for streaming or hairpinning, thus network clocking propagation with other boards is not required. For this configuration, the board’s SyncRoute cable is not used. In this case, the DNI board’s TDM bus is configured with the HMP clocking daemon set to Active, and the Dialogic® DSI SS7LDH4Q board is configured to recover clock from the network with CT Bus clocking mode set to disabled.

For other CT Bus clocking configurations that require the interconnection of the Dialogic® DSI SS7LDH4Q board with other boards in the system through the SyncRoute connector, only one board in the system can act as a primary clock master at any given time; please refer to the respective guides on how to configure each TDM Bus manually according to the specific TDM system clocking requirements.

Refer to the *Dialogic® Distributed Signaling Interface SS7LDH4Q Network Interface Board Installation Guide* for more information about the board and the SyncRoute cable, and to *Dialogic® Distributed Signaling Interface SS7LD Network Interface Boards Programmer’s Manual*, *Dialogic® Host Media Processing for Linux Configuration Guide*, *Dialogic® Global Call SS7 Technology Guide*, and *Dialogic® Distributed Signaling Interface SS7HD Network Interface Boards Programmer’s Manual* for other information.

1.51 Enhanced Direct 3GP File Play Capabilities

With Service Update 94, Dialogic® HMP software supports 3GP files with H.264 video and AMR-WB audio. In addition, the software supports a 3GP play operation regardless of whether or not hint tracks are present in the file. Previously, only a hinted 3GP file with MPEG-4/H.263 video plus AMR-NB audio was supported.

1.51.1 Feature Implementation

A complete description of the enhanced 3GP direct play capabilities can be found in the “3GP File Format Direct Playback” in *Dialogic® Multimedia API Programming Guide and Library Reference*. The guide had been revised for this feature.

New API Set and Demo Program

A new API set, Media Parsing (MP), is added to Dialogic® Media Toolkit API library. Media Parsing functions are used to retrieve file information from multimedia files. This file information can be used by the application to:

- decide whether video transcoding/transrating/transsizing is necessary.
- decide whether audio transcoding/transrating is necessary.
- retrieve information necessary for call/media session setup purposes.
- decide which media track to play

Note: Track selection is currently not supported with the Multimedia API.

The Media Parsing API set consists of the following functions:

mp_CloseFile()

closes a previously-opened multimedia file

mp_GetFileInfo()

retrieves detailed file information from a multimedia file

mp_GetFileInfoSize()

optional function that returns the minimum buffer size needed to pass to the **mp_GetFileInfo()** function

mp_OpenFile()

opens a multimedia file for reading

A demo program, *mptestdemo*, is added to demonstrate Media Parsing API functionality. Its source code, along with a readme file containing usage and sample output, is located in the */usr/dialogic/demos/mptestdemo* directory. In addition, a precompiled executable, *mpctest*, is located in the */usr/dialogic/bin* directory.

Refer to the *Dialogic® Media Toolkit API Library Reference* on the documentation bookshelf for more information about Media Parsing APIs and the *mptestdemo*. The guide had been revised for this feature.

1.52 Enhanced Video Active Talker Design

Service Update 94 introduces a new approach for the Video Active talker feature. This enhancement was made in response to IPY00093585. Refer to the [Release Issues](#) chapter for more information.

1.52.1 Conferencing API Changes

A new attribute, `ECNF_CONF_ATTR_VIDEO_SWITCHING_INTERVAL`, is added to the `ECNF_CONF_ATTR` data type. This attribute specifies the video active talker switching interval for the conference.

The `ECNF_CONF_ATTR` enumeration is:

```
typedef enum tagECNF_CONF_ATTR
{
    ECNF_CONF_ATTR_TONE_CLAMPING    = 101,
    ECNF_CONF_ATTR_DTMF_MASK        = 102,
    ECNF_CONF_ATTR_NOTIFY           = 103,
    ECNF_CONF_ATTR_MAX_ACTIVE_TALKERS = 104,
    ECNF_CONF_ATTR_VIDEO_SWITCHING_INTERVAL = 105,
    ECNF_CONF_ATTR_END_OF_LIST
} ECNF_CONF_ATTR;
```

Setting this parameter decides how often firmware determines video active talker. For example, if this parameter is set to one second, firmware will determine every one second which party is active talker and displays that party (active talker mode) for the next one second. After one second expires, firmware will again make a decision about which party is active talker. If the party being displayed is the loudest participant again, then firmware continues to display that party, otherwise it switches the new loudest party to active talker mode.

The `ECNF_CONF_ATTR_VIDEO_SWITCHING_INTERVAL` attribute is set using the **`cnf_SetAttributes()`** function. Valid values are 100 ms to 5000 ms in increments of 10 ms. The default value is 500 ms.

The value of `ECNF_CONF_ATTR_VIDEO_SWITCHING_INTERVAL` is retrieved using the **`cnf_GetAttributes()`** function.

For more information about the Conferencing APIs, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.53 CentOS 5 Update 6 Support

Service Update 87 introduces CentOS 5 Update 6 64-bit operating system support. CentOS, a Community ENTERprise Operating System, is based on Red Hat Enterprise Linux (RHEL).

1.54 Increased Density for G.711, Low Bit Rate Coders, and High Density Conferencing Placeholder

Service Update 87 supports up to 2000 channels of G.711, AMR-NB, G.726, as well as 2000 channels of audio conferencing when using G.711.

1.54.1 Configuration and Performance Recommendations

System requirements for densities of approximately 1500 channels and greater have proven to be more demanding with respect to maintaining real-time performance, consistent high audio quality, and reliable signal detection. It was found that high density systems were often limited by the system I/O performance, including the hard drive subsystem throughput, more so than the system CPU, specifically for applications requiring heavy file I/O access. As a result of these findings, for higher density systems, Dialogic **encourages** adherence to recommended performance tuning settings plus selection of hardware platforms meeting the following **minimum** system configuration:

- Intel Dual Processor X5650 or better
- 8 GB 1333MHz UDIMM in a memory optimized configuration for the Nehalem architecture
- RAID Controller, 1GB NV Cache, 6 Gbps SAS
- Two 300 GB 15K RPM Serial-Attach SCSI 6 Gbps configured for RAID 0
- Intel Gigabit ET NIC, Dual Port, Copper, PCIe-4
- Red Hat Enterprise Linux or CentOS 5 Update 6, 64 bit

The maximum new licensable densities are as follows:

Feature	Prior to SU 87	SU 87 and later
Basic RTP Streaming	1500	2000
G.711, G.726 Coder	1500	2000
AMR-NB Coder	750	2000
G.723 Coder	750	1900
G.729 Coder	750	1200
Voice	1500	2000
HD Voice Conferencing	500	2000
G.722	--	2000
AMR-WB	--	600

Note: The same configuration cannot be used to run all of these features simultaneously.

1.54.2 Performance Recommendation

The following two new performance tuning settings have been identified in addition to those already specified in the *Dialogic® Host Media Processing Software for Linux Configuration Guide*.

- To achieve optimal performance, the CPU speed service must be disabled. This service reduces the clock speed of the CPU when the service detects that demand for the CPU has decreased.
- Maximum shared memory should be increased to support the high densities specified above. Currently, the *Dialogic® Host Media Processing Software for Linux Configuration Guide* recommends 65536000 (64 MB), however, in order to download

some of these licenses, memory should be increased to 134217728 (128 MB). To increase the size to 128 MB, use the following command:

```
sysctl -w kernel.shmmax=134217728
```

To make this setting persistent across reboots, add the `kernel.shmmax=134217728` variable to the `/etc/sysctl.conf` file.

1.55 Support for Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP Boards

Service Update 86 introduces support for the Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP boards. Both boards are high-density, high-performance, digital network interface boards. The DNI1210TEPE2HMP board has four T1/E1 network interfaces, while the DNI2410TEPE2HMP board has eight T1/E1 network interfaces in a half-length by 1 (x1) PCI Express form factor.

These boards allow HMP applications to connect to PSTN networks through a T1 or E1 interface. The digital network interface boards can be used for converged IP-PSTN solutions that require both IP and PSTN connectivity.

1.55.1 Installing the Board

For board installation instructions, refer to the Installation Guide (*Dialogic® Quick Install Card*) that comes with each board. The Installation Guide explains how to set the jumpers on the board, install the board in the computer, and connect to external PSTN equipment.

Notes: 1. If the board is not configured and installed properly, it will not be detected in the system.

2. The Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP boards require a system (chipset) that supports Message Signaling Interrupts (MSI). If your system does not support MSI, the board will not work. Be sure to check your system's specification to confirm that MSI is supported. You can see if MSI is supported by entering the following commands on the command line:

```
dmesg | grep -i msi
```

```
cat /proc/interrupts | grep -i msi
```

MSI will be shown if it is supported by the system.

1.55.2 Installing the Dialogic® HMP Software

The Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP boards require the use of a Dialogic® PowerMedia™ HMP for Linux Release 4.1 Service Update 86 or later.

Note: If the HMP software has already been installed without the T1/E1 package, reinstall it and select the correct package.

To check which packages are installed, enter:

```
rpm -qa ls-dialogic-hmp-\*
```

The names of the installed packages are listed. The name of the Dialogic® PowerMedia™ HMP for Linux Release 4.1 package with support for the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board is `ls-dialogic-hmp-t1e1`.

For detailed information about installing the Dialogic® PowerMedia™ HMP for Linux Release 4.1, refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide*.

1.55.3 Licensing

An HMP license is a file containing authorization for a combination of call control and media processing features. You can obtain a license file either before or after you install the HMP software and the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board, but you need to obtain a license file before you can proceed using the HMP software and these boards.

If you have been using the HMP software without digital network interface boards, you have a host-based license, which is associated with the host machine via its host ID (MAC address). You cannot use a host-based license with the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board. Instead, you will need a board-based license, which is associated with one of the boards in the system via its board serial number.

If you are adding either a Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board to a system that already has a board-based license, you can use the existing license but you may have to upgrade it if, for instance, you require additional media feature densities to account for the additional PSTN interfaces.

For detailed information about obtaining and upgrading a license, refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide*.

1.55.4 Configuring the Boards

After the Dialogic® HMP software is installed and the appropriate licensing is obtained, you can begin the configuration process. Two tools are available to perform configuration tasks:

- Command Line Interface (CLI)
- Simple Network Management Protocol (SNMP)

Both tools have access to the same configuration data.

The *Dialogic® Host Media Processing Software for Linux Configuration Guide* provides detailed instructions for using these tools to configure HMP software and digital network interface boards in the system. This section provides supplementary information that is applicable to the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board:

- [Trunk Configuration Using CLI](#)
- [Trunk Configuration Using SNMP](#)
- [FCD/PCD File Names](#)
- [Bridge Configuration](#)

1.55.4.1 Trunk Configuration Using CLI

The “Configuration Procedures Using CLI” chapter in *Dialogic® Host Media Processing Software for Linux Configuration Guide* includes a procedure for configuring PSTN network interfaces. You can follow that procedure for the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board. The procedure is repeated here, with supplementary information applicable to these boards.

1. Verify that the media services and boards are stopped. Refer to the “Stopping Media Services” procedure in *Dialogic® Host Media Processing Software for Linux Configuration Guide*.

Note: You can configure PSTN network interface settings while media services are active. The configuration is **updated**; however, the configuration is only **applied** (1) after media services are stopped, (2) the **conf system pstn apply** command is issued (see [Step 6](#)), and (3) media services are restarted. Thus, it is recommended that you stop media services before performing the procedure.

2. Retrieve the device ID for the hardware in your system.

```
CLI> show hardware
```

All Dialogic® devices are displayed including device ID, technology, device name, slot ID, and more.

3. Retrieve the current settings for the PSTN network interfaces.

```
CLI> show hardware pstn
```

Information about device ID, trunk ID, interface ID, protocol, and more is displayed.

4. Modify a current PSTN network interface setting for a specific device and trunk; for example, modify protocol.

Note: This step must be repeated for each trunk; entering a range of trunk IDs is not allowed.

```
CLI> conf hardware pstn <device ID> <trunk ID> protocol <protocol value>
```

You can leave out the <protocol value> to display all supported protocols. The following command and output are for the DNI2410TEPE2HMP board:

```
CLI> conf hardware pstn 2 1 protocol
missing option - expected
```

```
<integer> 1 - 4ESS
<integer> 2 - 5ESS
<integer> 3 - CAS
<integer> 4 - DMS
<integer> 5 - E1CC
<integer> 6 - NET5
<integer> 7 - NI2
<integer> 8 - NTT
<integer> 9 - QSIG1
<integer> 10 - QSIGT1
<integer> 11 - R2MF
<integer> 12 - T1CC
<integer> 13 - DASS2
<integer> 14 - DPNSS
<integer> 15 - E1CAS
```

Note: R2MF is not supported on the DNI1210TEPE2HMP or DNI2410TEPE2HMP board. You may assign the same protocol or different protocols to each trunk on the board, with some limitations. Refer to the [Guidelines for Assigning Protocols](#) section.

The following example sets the protocol for device 2 trunk 1 to QSIG1 (9):

```
CLI> conf hardware pstn 2 1 protocol 9
updated
```

The updated message is displayed.

5. Verify the new settings using the **show** command.

```
CLI> show hardware pstn
```

Information about device ID, trunk ID, interface ID, protocol, and more is displayed in two columns: User Defined Table and Resolved Table. The protocol value will show QSIG1 (the value "9" is translated) in the User Defined Table and the previous protocol setting for that trunk in the Resolved Table. The user defined value takes effect after the setting has been applied ([Step 6](#)),

6. Apply the new settings using the **conf** command. This step validates that the protocol is supported on this board. The setting takes effect on the next media services restart.

```
CLI> conf system pstn apply
updated
```

7. Verify that the User Defined Table and Resolved Table values match, indicating that the new value has been applied.

```
CLI> show hardware pstn
```

Information about device ID, trunk ID, interface ID, protocol, and more is displayed in two columns: User Defined Table and Resolved Table. In this example, after you have restarted media services, the protocol value will show QSIGE1 (the value “9” is translated) in the User Defined Table and in the Resolved Table. Any errors are displayed in the Error Description for each device.

8. Repeat steps 4 through 7 for each PSTN network interface setting to be modified.

Continue with any additional configuration procedures that are applicable to your system. When you are satisfied with all configuration information, restart the media services. Refer to the “Restarting Media Services” procedure in *Dialogic® Host Media Processing Software for Linux Configuration Guide*.

Guidelines for Assigning Protocols

The protocols supported on the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board are as follows (the protocols are shown as “Group 1” and “Group 2”, which is explained below):

Group 1 Protocol Values	Group 2 Protocol Values
4ESS (T1)	DPNSS (E1)
5ESS (T1)	DASS2 (E1)
CAS (T1)	
DMS (T1)	
NI2 (T1)	
NTT (T1)	
QSIGT1 (T1)	
T1CC (T1)	
E1CC (E1)	
NET5 (E1)	
QSIGE1 (E1)	
E1CAS (E1)	

You may assign the same protocol or different protocols to each trunk on the board, but all of the protocols on either board must belong to the same group. In addition, the DNI1210TEPE2HMP board and trunks 1 - 4 on the DNI2410TEPE2HMP board must have the same line type (T1 or E1), and all of the protocols on trunks 5 - 8 on the DNI2410TEPE2HMP board must have the same line type.

The FCD and PCD file names are updated appropriately after you configure the protocols for the PSTN network interface. Refer to the [FCD/PCD File Names](#) section.

1.55.4.2 Trunk Configuration Using SNMP

The “Configuration Procedures Using SNMP” chapter in *Dialogic® Host Media Processing Software for Linux Configuration Guide* includes a procedure for configuring PSTN network interfaces. You can follow that procedure for the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board. The procedure is repeated here, with supplementary information for the DNI2410TEPE2HMP board.

1. Verify that the media services and boards are stopped. Refer to the “Stopping Media Services” procedure in *Dialogic® Host Media Processing Software for Linux Configuration Guide*.
2. In the **ipmsHwDM3Trunk** MIB, modify a current PSTN network interface setting for a specific device and network interface (trunk). For example, modify the protocol using **ipmsctHwDM3TrunkProtocol** object. The change takes effect on the next media services restart.

Notes: 1. R2MF is not supported on the DNI1210TEPE2HMP and DNI2410TEPE2HMP boards.

2. You may assign the same protocol or different protocols to each trunk on the boards, with some limitations. Refer to the [Guidelines for Assigning Protocols](#) section.

Continue with any additional configuration procedures that are applicable to your system. When you are satisfied with all configuration information, restart the media services. Refer to the “Restarting Media Services” procedure in *Dialogic® Host Media Processing Software for Linux Configuration Guide*.

Note: Configuration files must be updated *before* starting media services.

1.55.4.3 FCD/PCD File Names

The FCD and PCD files make up the configuration file set that is downloaded to the board. The default FCD and PCD files for the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board are *hmposbe_default.fcd* and *hmposbe_default.pcd*.

After you select the protocols by using either CLI or SNMP as shown above, new FCD and PCD files are automatically generated to reflect the protocols that were selected. FCD and PCD file names that begin with “g” are files that have been generated. For example, *gnetworkonly_hmposbe_1_net5_7_qsige1.pcd* is a generated PCD file for a DNI2410TEPE2HMP board using NET5 protocol on one trunk and QSIG1 protocol on the other seven trunks. The generated PCD file, *gnetworkonly_hmposbe_1_net5_3_qsige1.pcd*, would be for a DNI1210TEPE2HMP board using NET5 protocol on one trunk and QSIG1 protocol on the other three trunks.

When using SNMP, you can see the FCD/PCD file names in the **ipmsHwDM3Dev** MIB.

When using CLI, you can see the FCD/PCD file names by entering **show hardware dm3**. In the following example, Device ID 1 is the HMP software virtual device and Device ID 2 is the DNI2410TEPE2HMP board. Sample command and sample output are:


```

CLI> show hardware dm3
Device ID 1
    Logical ID:      1
    Model Number:    32809
    Media Load:      HMP Load
    PCD File Name:    0r360v0e0c0s0f0i0m0a0u0n0g_hib_pur.pcd
    FCD File Name:    0r360v0e0c0s0f0i0m0a0u0n0g_hib_pur.fcd
    PCM Encoding:     Mu-Law
    Driver State:     DOWNLOADED
    CFG ID:           21

Device ID 2
    Logical ID:      2
    Model Number:    7440
    Media Load:      Network Only
    PCD File Name:    gnetworkonly_hmposbe_8_net5.pcd
    FCD File Name:    gnetworkonly_hmposbe_8_net5.fcd
    PCM Encoding:     A-Law
    Driver State:     DOWNLOADED
    CFG ID:           2

```

Note: It is recommended that you do **not** modify the FCD/PCD file names via the CLI **conf hardware dm3** command or via SNMP. The FCD/PCD file name values are updated when you configure the protocol for the network interface.

For detailed information about FCD and PCD files, refer to the *Dialogic® Host Media Processing Software for Linux Configuration Guide*.

1.55.4.4 Bridge Configuration

The Dialogic® DNI1210TEPE2HMP and DNI2410TEPE2HMP boards provide a dedicated bridge. A dedicated bridge does not contain a switching handler; thus every Host Streaming Interface (HSI) stream is mapped on a one-to-one basis to a network port (channel).

The bridge is configured as if it was only voice, but the stream mode setting, voice or data (transparent mode), can be controlled using the **gc_SetConfigData()** function on a channel (stream) basis at run time.

The user may query the bridge parameters via the CLI. In the example below, values shown represent what is always shown for the DNI2410TEPE2HMP board, while values not shown may vary depending on system configuration.

```

CLI> show hardware bridge
Bridging and Streaming Device Level Attribute Settings

    BH Stream Capacity: 255
    HB Stream Capacity: 255
    Max BH Data Streams: 0
    Max HB Data Streams: 0
    Max BH Voice Streams: 255
    Max HB Voice Streams: 255

```

Note: It is neither required or recommended to use the **conf hardware bridge** command on the either the DNI1210TEPE2HMP or DNI2410TEPE2HMP board.

1.55.5 Transparent Mode Settings

The **gc_SetConfigData()** function is used to configure a network channel (dtiBxTy) for transparent mode on a channel-by-channel basis. The function uses a GC_PARM_BLK structure that contains the configuration element. The GC_PARM_BLK is populated using the gc_util_insert_parm_val() function.

To enable/disable transparent mode on a channel, use the following setID/parmID pair:

- CCSET_DM3FW_PARM is the setID
- CCPARM_TRANSPARENTMODE is the parmID, with values:
 - CCDM3FW_TRANSPARENTMODE_ENABLE Enable transparent mode
 - CCDM3FW_TRANSPARENTMODE_DISABLE Disable transparent mode

The size of the parameter is a char or UINT8.

The **gc_SetConfigData()** function is issued to set the configuration once the GC_PARM_BLK has been populated with the desired values. Use the target type GCTGT_CCLIB_CHAN.

Once a network interface channel is configured for transparent mode, law conversion is removed and TDM DS0 64 Kbps bit stream is streamed to and from HMP as raw 8-bit, 64Kbps data. This feature can be used for [Playing and Recording Raw E1/T1 DS0 64 Kbps Bit Streams](#). For additional information along with a code sample, refer to the [Playing and Recording Raw E1/T1 DS0 64 Kbps Bit Streams](#) section. Refer to the *Dialogic® Global Call API Library Reference* for more information about the **gc_SetConfigData()** function.

1.55.6 Configuration for 3G-324 Operation

There is no bridge configuration required in order to configure the Dialogic® DNI1210TEPE2HMP or DNI2410TEPE2HMP board for 3G-324M operation. Instead, the application will set individual network interface channels in transparent mode as indicated above in the [Transparent Mode Settings](#) section. Once a network interface channel is set to transparent mode, it is suitable for 3G-324M operation.

1.56 Mute Audio Attribute Support

With Service Update 86, the application has the ability to mute and un-mute parties of a multimedia conference so that the muted parties will not be heard by other members of the conference.

1.56.1 Feature Description

This feature provides a programmatic interface allowing the application to mute and un-mute the audio stream received from a conference party. This feature applies when using the MCX board device only.

Note: This feature does not support a CNF board device at this time.

For more information about the Conferencing APIs, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.57 Privilege Talker Attribute Support

With Service Update 85, the application can explicitly delegate which conference participants are always included in the conference summation output process.

1.57.1 Feature Description

Prior to this feature, HMP determined the conference parties included in each summation cycle based on their speech level, resulting in the “loudest” active talkers for a given sample cycle being selected for conference summation. Now, when the new “Privilege Party” attribute is assigned to a conference party, that party’s input, providing its speech level is greater than zero, is always included in the output summation process along with the loudest remaining “Normal/Pupil” parties within the active talker limit defined for the conference.

For more information about the Conferencing APIs, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.58 Support for RFC 3311 UPDATE Message

With Service Update 85, the application can support SIP UPDATE requests and responses in third-party call control (3PCC) mode.

1.58.1 Feature Implementation

This feature will allow user applications to send and receive SIP UPDATE requests and responses to modify the state of a pending or pre-established session. Prior to this Service Update, the UPDATE message was only supported in conjunction with SIP Session Timer refreshing and did not address the updating of session parameters. This feature will utilize the Global Call Extension API mechanism for application notification and message construction.

1.58.2 Data Structure Changes

The following new member in the IP_VIRTBOARD data structure allows the application to send and receive UPDATE requests and responses:

```
pVIRTBOARD
    EnumSIP_Enabled
    E_SIP_UPDATE_Access;
```

The default value of E_SIP_UPDATE_Access is ENUM_Disabled.

1.58.3 New Parameter Values

The existing ParmID, IPPARM_MSGTYPE, supports this feature. The following new IPPARM_MSGTYPE parameter values are used for sending and receiving UPDATE requests and responses:

ParmID	Value	Description
IPPARM_MSGTYPE	IPPARM_MSG_SIP_RESPONSE_CODE	Inserted or extracted from the parameter block. Used by the application to set or get the SIP-specific response code. Associated SIP Response Codes: any
	IP_MSGTYPE_SIP_UPDATE	Contained in the parameter block when calling the gc_Extension() function or when receiving a GCEV_EXTENSION event. Used by the application to set or determine the event type. Associated SIP Response Codes: not applicable

ParmID	Value	Description
	IP_MSGTYPE_SIP_UPDATE_OK	Contained in the parameter block when calling the gc_Extension() function or when receiving a GCEV_EXTENSION event. Used by the application to set or determine the event type. The SIP-specific response code can be inserted or extracted from the IP_MSGTYPE_SIP_UPDATE event.
		Associated SIP Response Codes: 200-299
	IP_MSGTYPE_SIP_UPDATE_FAILED	Contained in the parameter block when calling the gc_Extension() function or when receiving a GCEV_EXTENSION event. Used by the application to set or determine the event type. The SIP-specific response code can be inserted or extracted from the IP_MSGTYPE_SIP_UPDATE_FAILED event.
		Associated SIP Response Codes: 300+

1.58.4 Example

The following example demonstrates enabling this feature at the board level to support sending and receiving a SIP UPDATE.

```
.
.
INIT_IPCCLIB_START_DATA(&ipcclibstart, 2, ip_virtboard);
INIT_IP_VIRTBOARD(&ip_virtboard[0]);
INIT_IP_VIRTBOARD(&ip_virtboard[1]);
Ip_virtboard[0].E_SIP_UPDATE_Access = ENUM_Enabled;
Ip_virtboard[1].E_SIP_UPDATE_Access = ENUM_Enabled;
Ip_virtboard[0].sip_msginfo_mask = IP_SIP_MSGINFO_ENABLE;
Ip_virtboard[1].sip_msginfo_mask = IP_SIP_MSGINFO_ENABLE;
.
.
```

1.58.5 Usage Scenarios

The following sections provide different scenarios for using this feature. Sample code is provided after each scenario.

Note: All examples shown in this section are pre-connection only.

Send an UPDATE Request

When SIP UPDATE access is enabled, applications use the **gc_Extension()** function to send the message after assembling the appropriate header fields and any SDP parts. To build an UPDATE request, the application uses the parameter set ID IPSET_MSG_SIP, the parameter ID IPPARM_MSGTYPE and the parameter value IP_MSGTYPE_SIP_UPDATE.

The application can send an UPDATE request within a dialog by using the line device handle in the **gc_Extension()** function call: **gc_Extension(GCTGT_GCLIB_CHAN, linedevhandle, IPEXTID_SENDMSG, parmbldp, &retbldp, EV_ASYNC)**.

If the application requires SDP information, the information must be explicitly inserted using the **gc_SetUserInfo()** function and the IPSET_SDP/IPPARM_SDP_OFFER/IPPARM_SDP_ANSWER parameter.

Once the header fields are set up, the application can send the message within a dialog using: **gc_Extension(GCTGT_GCLIB_CRN, crn, IPEXTID_SENDMSG, parmbldp, &retbldp, EV_ASYNC)**

Sample Code

```
// Set the SDP for 3PCC only
printf("\n Setting SDP \n");
GC_PARM_BLK libBlock = NULL;
libBlock = set3PCCSDPInfo1(libBlock, 0, true);

if (gc_SetUserInfo(GCTGT_GCLIB_CRN,
                  pline->crn,
                  libBlock,
                  GC_SINGLECALL) != GC_SUCCESS)
{
    printf("\n gc_SetUserInfo failed for setting SDP \n");
    exit(1);
}

if (libBlock) gc_util_delete_parm_blk(libBlock);

// Send the UPDATE message
pline->crn = callindex;
printf("Sending UPDATE ..... \n");
sendUpdate(pline->crn);
..
..
..

GC_PARM_BLK set3PCCSDPInfo1(GC_PARM_BLK libBlock,int index, bool isAnswer)
{
    if (isAnswer) {} // For compilation

    char sdp_buf[1024];
    int ip_parm = IPPARM_SDP_ANSWER;

    sprintf(sdp_buf,
            "v=0%c%c" \
            "o=- 25678 753849 IN IP4 192.168.0.12%c%c" \
            "s=xxxSession %d %sStartxxx%c%c" \
            "c=IN IP4 192.168.0.12%c%c" \
            "t=0 0%c%c" \
            "m=audio 3456 RTP/AVP 0%c%c" \
            "\n");
```

```

        "a=ptime:20%c%c",
        0xd, 0xa,
        0xd, 0xa,
        index, "Slow", 0xd, 0xa,
        0xd, 0xa,
        0xd, 0xa,
        0xd, 0xa,
        0xd, 0xa);

if ((gc_util_insert_parm_ref_ex(&libBlock, IPSET_SDP, ip_parm, strlen(sdp_buf)+1, sdp_buf))
    != GC_SUCCESS) {
    printf("gc_util_insert_parm_ref_ex(IPSET_SDP, ip_parm=0x%x) failed: ", ip_parm);
} else {
    printf("GC_APP : [%d] 3PCC SDP: body inserted (IPPARM=0x%x):\n%s\n\n", index, ip_parm,
        sdp_buf);
}
return libBlock;
}

static void sendUpdate(int callindex)
{
    printf("sendUpdate:: callindex = %d \n", port[0].crn);

    GC_PARM_BLK    parmblkp=NULL;
    GC_PARM_BLK    retblkp=NULL;

    gc_util_insert_parm_val(&parmblkp,
                            IPSET_MSG_SIP,
                            IPPARM_MSGTYPE,
                            sizeof(int),

    IP_MSGTYPE_SIP_UPDATE);
    if (gc_Extension(GCTGT_GCLIB_CRN, port[0].crn, IPEXTID_SENDSMSG, parmblkp, &retblkp, EV_ASYNC)
        != GC_SUCCESS) {
        printf("\n <---- gc_GetCallInfo(crn=0x%x) Failure - calling party not available \n",
            callindex);
        exit(1);
    } else {
        printf("Sent UPDATE successfully \n");
    }
}
}

```

Receive an UPDATE Response

When the Global Call API library's SIP stack receives a response to a SIP UPDATE request, it generates a GCEV_EXTENSION event of type IPEXTID_RECEIVMSG.

The GC_PARM_BLK associated with the GCEV_EXTENSION event will contain a parameter element as follows:

IPSET_MSG_SIP
 IPPARM_MSGTYPE parameter ID
 And one of the following values:

- IP_MSGTYPE_SIP_UPDATE_OK
- IP_MSGTYPE_SIP_UPDATE_FAILED

The application may also retrieve the specific SIP response code from the event's parameter block using the IPSET_MSG_SIP set ID and the IPPARM_MSG_SIP_RESPONSE_CODE parameter ID.

The SDP information from the responses will be passed to the application using the IPSET_SDP set ID and the IPPARM_SDP_UPDATE_OFFER/IPPARM_SDP_UPDATE_ANSWER parameter ID.

If an IP_MSGTYPE_SIP_UPDATE_FAILED response is received, the application can retrieve the SIP header fields from the meta event. The application processes this using the **gc_GetMetaEvent()** function, and then processes GC_PARM_BLK using the Global Call utility functions to retrieve the message type information and individual SIP header fields.

Sample Code

```
case GCEV_EXTENSION:
{
    printf("\n ----> GCEV_EXTENSION \n");
    getExtensionInfo(metaeventp, 0);
}
break;

int getExtensionInfo(METAEVENT * metaeventp, int index)
{
    int retCode = 0; // for success case.

    GC_PARM_BLK gcParmBlkp = NULL;
    GC_PARM_DATAP t_gcParmDatap = NULL;
    EXTENSIONEVTBLK *ext_evtblkp = NULL;
    GC_IE_BLK * t_gcIEBlk = NULL;
    char str[500];

    ext_evtblkp = (EXTENSIONEVTBLK *)metaeventp->extevtdatap;
    gcParmBlkp = &ext_evtblkp->parmbkp;

    sprintf(str, "Received GCEV_EXTENSION event with ExtID = 0x%x", ext_evtblkp->ext_id);
    printf("%s \n", str);
    while(t_gcParmDatap = gc_util_next_parm(gcParmBlkp, t_gcParmDatap))
    {
        switch (t_gcParmDatap->set_ID)
        {
            case IPSET_SDP:
                printf("IPSET_SDP \n");

                switch(t_gcParmDatap->parm_ID)
                {
                    case IPPARM_SDP_OFFER:
                    case IPPARM_SDP_ANSWER:
                        printf("GC_APP : [%d] SDP recieved (IPPARM=0x%x):\n \n\n", index,
                            t_gcParmDatap->parm_ID);
                        break;

                    default:
                        printf("setID is IPSET_SDP \n");
                        break;
                }
            }
        }
    }
    break;

    case IPSET_MSG_SIP:
    if (t_gcParmDatap->parm_ID == IPPARM_MSGTYPE)
    {
        int l_mtype = (int)(t_gcParmDatap->value_buf);
        if (l_mtype == IP_MSGTYPE_SIP_UPDATE_OK)
        {
            printf("GC_APP : Received 200OK for UPDATE \n");
        }
    }
}
```



```

        } else if (l_mtype == IP_MSGTYPE_SIP_UPDATE_FAILED)
        {
            printf("GC_APP : Received 3XX ~ 5XX response for UPDATE \n");
        } else {
            printf("GC_APP : This is the default for IPPARM_MSGTYPE \n");
        }
    }
    break;
    default:
        printf("This is default \n");
        break;
    }
}
}
}

```

Receive an UPDATE Request

When the Global Call API library is started with the IP_VIRTBOARD>E_SIP_UPATE>Access field set to ENUM_Enabled, the library will generate a GCEV_CALLUPDATE event to the application when the SIP stack receives an incoming SIP UPDATE message. The application can extract standard message header fields from the parameter block associated with the GCEV_CALLUPDATE event.

The SDP information from the responses is passed to the application using the IPSET_SDP set ID and the IPPARM_SDP_UPDATE_OFFER/IPPARM_SDP_UPDATE_ANSWER parameter ID.

Sample Code

```

case GCEV_CALLUPDATE:
    printf("\n ---> GCEV_CALLUPDATE \n");
    printf("\n *** GCST_ACCEPTED State *** \n");
    pline->call_state = GCST_ACCEPTED;

    printf("\n Extract the SDP from GCEV_CALLUPDATE \n");
    if (get3PCCSDPInfo((GC_PARM_BLK *)metaeventp->extevtdatap, 0))
    {
        printf("ModifyMedia Already done ..... \n");
    }
    // Send 200 OK for UPDATE
    printf("Responding to the UPDATE message with 200 OK.... \n");
    sendUpdate(pline->crn);

break;
int get3PCCSDPInfo(GC_PARM_BLK* pParmBlk, int index)
{
    int retCode = 0; // for success case.
    printf("GC_APP : [%d] Looking for SDP...\n",index);

    if (pParmBlk) {
        GC_PARM_DATA_EXT ParmDataExt;

        //Initialize the structure to start from the 1st parm in the GC_PARM_BLK
        INIT_GC_PARM_DATA_EXT(&ParmDataExt);
        int UtilRet = gc_util_next_parm_ex(pParmBlk, &ParmDataExt);
        while (GC_SUCCESS == UtilRet) {

            if (ParmDataExt.set_ID == IPSET_SDP) {
                switch (ParmDataExt.parm_ID)
                {
                    case IPPARM_SDP_OFFER:
                    case IPPARM_SDP_ANSWER:

```

```

        // What type of SDP do we expect?
        retCode = 1;
        printf("GC_APP : [%d] SDP recieved (IPPARM=0x%x):\n%s\n\n", index, ParmDataExt.parm_ID, (char
        *) (ParmDataExt.pData));
        return retCode;
break;

default:
    printf("GC_APP : [%d] ERROR!!! 3PCC SDP recieved, invalid IPPARM!
    (IPPARM=0x%x):\n%s\n\n", index, ParmDataExt.parm_ID, (char
    *) (ParmDataExt.pData));
    return retCode;
    break;
    }
}

UtilRet = gc_util_next_parm_ex(pParmBlk, &ParmDataExt);
}
    printf("No ParmBlk for IPSET_SDP found \n");
}
else
{
    printf("No ParmBlk for SDP found \n");
}
return retCode;
}

```

Send an UPDATE Response

Once an application has received a GCEV_CALLUPDATE event for a SIP UPDATE message and extracted the information from the event, it sends a response message.

The response is sent by passing a GC_PARM_BLK structure containing the following parameter to the **gc_Extension()** function:

```

IPSET_MSG_SIP
    IPPARM_MSGTYPE
    And one of the following parameter values:
        • IP_MSGTYPE_SIP_UPDATE_OK
        • IP_MSGTYPE_SIP_UPDATE_FAILED

```

The application may also set a specific SIP response code in the response message using the following parameter:

```

IPSET_MSG_SIP
    IPPARM_MSG_SIP_RESPONSE_CODE
    And one of the following values:
        • For an "OK" response, the values should be in the range 200 to 299. Default value is 200.
        • For a "Failed" response, the value should be 300 or higher. Default value is 501.

```

Sample Code

```

case GCEV_CALLUPDATE:
    printf("\n ---> GCEV_CALLUPDATE \n");
    printf("\n *** GCST_ACCEPTED State *** \n");
    pline->call_state = GCST_ACCEPTED;

    printf("\n Extract the SDP from GCEV_CALLUPDATE \n");

```

```

        if (get3PCCSDPInfo((GC_PARM_BLK *)metaeventp->extevtdatap, 0))
        {
            printf("ModifyMedia Already done .....\\n");
        }
        // Send 200 OK for UPDATE
        printf("Responding to the UPDATE message with 200 OK.... \\n");
        sendUpdateResponse(pline->crn);

break;

static void sendUpdateResponse(int callindex)
{
    printf("sendUpdateResponse:: callindex = %d \\n", port[0].crn);
    GC_PARM_BLKpparmblkp=NULL;
    GC_PARM_BLKPretblkp=NULL;

    gc_util_insert_parm_val(&parmbkp,
                           IPSET_MSG_SIP,
                           IPPARM_MSGTYPE,

sizeof(int),
IP_MSGTYPE_SIP_UPDATE_OK);

    gc_util_insert_parm_val(&parmbkp,
                           IPSET_MSG_SIP,
                           IPPARM_MSG_SIP_RESPONSE_CODE,
sizeof(int),
200);

    if (gc_Extension(GCTGT_GCLIB_CRN, port[0].crn, IPEXTID_SENDMSG, parmbkp, &retblkp,
EV_ASYNC) != GC_SUCCESS) {
        printf("\\n <---- gc_GetCallInfo(crn=0x%lx) Failure - calling party not available \\n",
callindex);
        exit(1);
    } else {
        printf("Sent UPDATE successfully \\n");
    }
}
}

```

For more information about Global Call IP, refer to the following documents:

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

1.59 SIP Session Timer

With Service Update 85, the user can set the session duration to keep the Session Initiation Protocol (SIP) sessions active. The feature includes a keep alive 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.

1.59.1 Feature Description

The feature provides a keep alive 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.

1.59.2 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.

1.59.3 Session Timer Negotiation

Session timer negotiation between UAS and UAC follows RFC 4028. The following SIP IP parameters have been added via the GC 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.

1.59.4 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 Interval 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.
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.

Parameter ID	Data Type & Size	Description
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.

1.59.5 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;
```

1.59.6 Documentation

For additional information, refer to the following documents:

- *Dialogic® Global Call IP Technology Guide*
- *Dialogic® Global Call API Library Reference*
- *Dialogic® IP Media Library API Programming Guide and 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>

1.60 Support for Initial Silence Termination

With Service Update 85, initial silence termination (TF_SETINIT bitmask) can be used on any standard voice library playback or record or CSP record or stream as originally intended and documented.

For information about the TF_SETINIT bitmask, refer to the *Dialogic® Voice API Library Reference*.

This feature is disabled by default in HMP and can be enabled using the new InitSilResume parameter. In order to control the feature on an HMP board basis, the *Hmp.Uconfig* file should be used to add the new Signal Detector parameter:

Parameter: 0x711

Value: 0 = Disable (default), 1 = Enable

Follow the steps described in the “Preserving Data in User Configuration Files” section in *Dialogic® Host Media Processing Software for Linux Configuration Guide* for details about configuring an *Hmp.Uconfig* file. The steps below describe how to enable the ability to set initial silence termination on all voice channels:

1. Add the new InitSilResume parameter within the sigDet section of the *Hmp.Uconfig* file. This example contains just one sigDet parameter, however if other parameters existed prior to this feature, you would add the following information in a new line at the end of the sigDet section:

```
[sigDet]
SetParm=0x0711,1 ! InitSilResume (0=Disable, 1=Enable)on sigDet
```

2. Restart the HMP system to apply the InitSilResume to the HMP board and all of its available voice channels as per the license purchased. If the voice channels have speech (CSP) capabilities, then this feature will also be enabled on CSP.
3. Refer to the *Dialogic® Voice API Library Reference* for information about TF_SETINIT and its usage in voice applications, and refer to the *Dialogic® Continuous Speech Processing API Library Reference* for TF_SETINIT usage in a speech application.

In summary, if using tp_termno=DX_MAXSIL as a termination condition, initial silence termination can now be enabled by ORing the TF_SETINIT bitmask in the DV_TPT tp_flags and setting the initial silence duration value in the tp_data field. tp_length continues to carry the length of the normal silence length for termination.

Note: With CSP (speech), the initial silence termination TF_SETINIT must be ORed with the TF_IMMEDIATE for it to take effect. This means that all silence detection starts immediately at the onset of **ec_stream()** or **ec_reciottdata()** instead of waiting for the playback on the same channel to finish (default).

1.61 PAE Support

Service Update 82 adds support for Physical Address Extensions (PAE). PAE allows x86 processors to access a physical address space (including random access memory and memory mapped devices) larger than 4 GB. To enable PAE in Linux, switch the Linux kernel to a PAE-enabled one (kernels with PAE in the name).

1.62 Using MCX Resources for Audio Conferencing

Service Update 82 increases support for up to 1500 channels of conferencing using either Dialogic® R4 APIs with the MCX device handle or the Dialogic® MSML Media Server APIs.

If you are using Dialogic® R4 APIs, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference* for more information. If you are using Dialogic® MSML Media Server APIs, refer to the *Dialogic® MSML Media Server Software User's Guide* for more information.

In addition, this feature introduces support for the MSML audit package. For more information, refer to the *Dialogic® MSML Media Server Software User's Guide*.

1.63 Support for Multiple NICs

With Service Update 71, Dialogic® HMP software permits the use of multiple local IP addresses for media sessions in Global Call 3PCC mode or with third-party SIP stacks. The feature is limited to the IP Media library and to audio and video media sessions.

1.63.1 Feature Description

Currently, the IP Media Library ties all RTP/RTCP ports used for media streaming into one local IP address (and thus only one network interface card (NIC)). With this feature, RTP ports for media streaming are freely associated to one or more local IP address, independent of the call control IP address, and selectable on a media device basis at runtime.

There are two main aspects of the feature; one is allowing for automatic OA&M HMP registration of all local, enabled, LAN network interfaces in the system, making them available to the run-time media library. The second aspect involves the IP Media library, which now provides the list of IP addresses provided by OA&M to the application. The application is able to select any of the IP addresses on a channel basis; a new API interface will allow for the selection of different local IP interfaces to IP media channels.

In addition, HMP allows for static configuration of RTP and RTCP base UDP ports for each of the available IP addresses in the list. In the case of multiple IP addresses per NIC, each address will be registered as a separate interface, thus making them all available to the IP Media library as if they were separate physical NICs.

The RTP and RTCP UDP ports assigned to each IP Media channel continue to be automatically calculated by HMP with the same algorithm used prior to this feature; that is the base RTP UDP port is assigned to the very first IP Media channel, and each subsequent channel is assigned the previous channel UDP port plus two (RTP+2).

By implementing this feature, HMP automatically retrieves the list of available local IP addresses enabled for IP traffic in a system, registers them, and then makes them available to the HMP IPVSC as a list. This list indicates which NIC is primary, with all remaining ones as secondary IP NICs. The primary NIC is used for all IP media streaming on all the IP Media channels by default. Current CLI and SNMP interfaces for the setting of the local “IP address to send RTP messages” (hmp-rtp-address) continues to be the method for changing the default primary IP address.

1.63.2 Runtime Implementation

An application uses the IP Media library functions to control media sessions within a dialog established using 3PCC mode or using a third-party call control stack. The **ipm_StartMedia()** function starts RTP streaming, while the **ipm_ModifyMedia()** function is used to modify a few properties of an active media session. The **ipm_GetLocalMediaInfo()** retrieves the local IP address and UDP port configuration of the associated IP media channel for audio and video RTP/RTCP. These properties are assigned during firmware download, which apply to all IP Media channels.

Media session attributes and values in the IP Media library are controlled by the settings in the IPM_MEDIA data structure. This structure contains information about RTP/RTCP ports, coders, security information, etc. It is a parent structure of the IPM_PORT_INFO and contains UDP port properties for audio or video RTP and RTCP.

```
typedef struct ipm_port_info_tag
{
    unsigned int unPortId;           /* The Port ID */
    char cIPAddress[IP_ADDR_SIZE];  /* IP Address */
} IPM_PORT_INFO, *PIPM_PORT_INFO;
```

1.63.3 Initialize the Interfaces Table

Initialization is accomplished when IP Media is started. Use the CLI (command line interface) to retrieve all the available local IP addresses. Example:

```
CLI>show system hmp-rtp-address
```

The output is similar to the following:

```
IF Name:  lo
MAC Address:      00:00:00:00:00:00
IPv4 Address:     127.0.0.1
IPv6 Address:     ::1
Selected:         false

IF Name:  eth0
MAC Address:      00:a0:cc:2d:83:7e
IPv4 Address:     10.10.20.31
IPv6 Address:     fe80::2a0:ccff:fe2d:837e
Selected:         false
```

```

IF Name: eth1
MAC Address: 00:1a:a0:cd:da:2f
IPv4 Address: 10.10.20.85
IPv6 Address: fe80::21a:a0ff:fe8d:da2f
Selected: true

```

Enter the following command to change the default address if desired:

```
conf system hmp-rtp-address <IPv4 or IPV6 address>
```

The IPv4 format is a dot string whereas the IPpv6 format is a semicolon string. If the changed address is IPV4, the IPV6 address of the same interface is selected and vice versa. The modification is taken into account **ONLY** after the next restart of ipmedia.

The audio port base can be also modified inside the config file, but the distribution of the ports for the channels are fixed with a calculation from base port value: $P = \text{BASE_PORT} + (\text{channel} * 2)$. The same applies for the video port that has a different base.

1.63.4 Retrieve the Interfaces Table

The application calls the existing **ipm_GetLocalMediaInfo**(nDeviceHandle, pMediaInfo, usMode) function with a new mediatype in pMediaInfo:

- MEDIATYPE_LOCAL_RTP_INFO_ENUM or
- MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6 for Ipv6 requests.

The information returned is the list of the available IP addresses in the system for the given type (IPv4 or IPv6). Each returned info is in a IPM_PORT_INFO or IPM_PORT_INFO_V6 structure according the requested mediatype. The returned port member has an undetermined value.

In synchronous mode, the table is returned in the pMediaInfo when the function returns. In asynchronous mode, the table is returned when the IPMEV_GET_INTERFACE_INFO event occurs.

Note: This request may include a lot of information in the returned structure in the case of several interfaces in the system. In order to avoid missing data, make only one request at a time, either IPv4 or IPv6, and do not include with any other media type.

Example

```

#define NOERROR 0
#define ERROR 1

int GetInterfaceInfo(int nDeviceHandle, int mode)
{
    IPM_MEDIA_INFO MediaInfo;
    MediaInfo.unCount = 1;

    if (mode==0) // IPV4 request
    {
        MediaInfo.MediaData[0].eMediaType = MEDIATYPE_LOCAL_RTP_INFO_ENUM;
    }
    else // IPV6 request

```

```

{
    MediaInfo.MediaData[0].eMediaType = MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6;
}

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

```

This an example of function that gets the events:

```

void *ProcessEvent(void *pContext)
{
    longlDev= 0;
    intnType= 0;
    long evt;
    longnDeviceHandle; // the handle of the device

    while(!g_bDone)
    {
        if(sr_waitevt(500) != -1)
        {
            evt = sr_getevtttype();
            nDeviceHandle = sr_getevtdev();
            void* pData = sr_getevtdatap();

            printf("Received event 0x%x h_dev 0x%x \n", evt, h_dev);
            switch(evt)
            {
                case IPMEV_ERROR:
                    printf("IPMEV_ERROR received device 0x%x \n", nDeviceHandle);
                    break;

                case IPMEV_GET_LOCAL_MEDIA_INFO:
                    ReceiveLocalMediaInfo(nDeviceHandle, (IPM_MEDIA_INFO*)pData);
                    break;

                // others cases ...

            default:
                printf("Received event 0x%x h_dev 0x%x \n", evt, nDeviceHandle);
                break;
            }

            } // end if

    } // end of while

    return 0;
}

```

This is an example of the function ReceiveLocalMediaInfo which manages the received info:

```

ReceiveLocalMediaInfo(int nDeviceHandle, IPM_MEDIA_INFO * pMediaInfo)
{
    unsigned inti;
    char cAddr6[48]={0};
    IPM_PORT_INFO*pInfo;
    IPM_PORT_INFO_V6*pInfoV6;

    printf("Received IPMEV_GET_LOCAL_MEDIA_INFO for device name = %s pMediaInfo 0x%x unCount
    %d \n", ATDV_NAMEP(nDeviceHandle), pMediaInfo, pMediaInfo->unCount);
    for(i=0; i<pMediaInfo->unCount; i++)
    {
        switch(pMediaInfo->MediaData[i].eMediaType)
        {
            case MEDIATYPE_LOCAL_RTP_INFO_ENUM:
                pInfo = &pMediaInfo->MediaData[i].mediaInfo.PortInfo;
                printf("MEDIATYPE_LOCAL_RTP_INFO_ENUM: PortId=%d, IP=%s \n",
                    pInfo->unPortId,
                    pInfo->cIPAddress);
                break;

            case MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6:
                pInfoV6 = &pMediaInfo->MediaData[i].mediaInfo.PortInfoV6;
                inet_ntop(AF_INET6, pInfoV6->bIPv6Address, cAddr6, 48);
                printf("MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6: PortId=%d, IP=%s \n",
                    pInfoV6->unPortId,
                    cAddr6);
                break;

            default:
                printf("MediaType[%d]=0x%x\n", i,pMediaInfo->MediaData[i].eMediaType);
                break;
        }
    }
}

```

1.63.5 Set an IP Address for a Channel

To set the IP address of an interface, use the **ipm_StartMedia()** function with the mediatype MEDIATYPE_LOCAL_RTP_INFO or MEDIATYPE_LOCAL_RTP_INFO_V6 for IPv6 type, and fill the structure IPM_PORT_INFO or IPM_PORT_INFO_V6 for IPv6 type with the address.

These mediatypes were previously limited to read local IP addresses with **ipm_GetLocalMediaInfo()**. With this new feature, this mediatype allows write data. The port number cannot be modified and should be set to zero. The mediatype sets audio and video with the same local IP address.

- Notes:**
1. These mediatypes cannot be used with the **ipm_ModifyMedia()** function.
 2. An IPMEV_ERROR will occur if an invalid address (one not found in the table) is provided.

Example

```
#define NOERROR 0
#define ERROR 1

// If desired the app can now select an appropriate interface from
// the list of available ones obtained from MEDIATYPE_LOCAL_RTP_INFO_ENUM or
// MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6 for starting a media session on a channel.
// As an example, we pass a selection as argument to a function as a char * ipAddress.
// Note: The application is responsible for selecting the appropriate one based on its specific
// algorithm.

int SendStartMedia(int nDeviceHandle, char * ipAddress, int mode)
{
    IPM_MEDIA_INFOMediaInfo;

    MediaInfo.unCount = 0;

    // set mediatypes needed to perform a valid ipm_StartMedia
    // ...

    // now set the optional mediatype to change local IP

    if (ipAddress) // you want to change local interface
    {
        if (mode == 0) // IPV4
        {
            MediaInfo.MediaData[MediaInfo.unCount].eMediaType = MEDIATYPE_LOCAL_RTP_INFO;
            strcpy(MediaInfo.MediaData[MediaInfo.unCount].mediaInfo.PortInfo.cIPAddress,
                ipAddress);

            MediaInfo.MediaData[MediaInfo.unCount].mediaInfo.PortInfo.unPortId = 0; // must be 0

            ++MediaInfo.unCount;
        }
        else // IPV6
        {
            MediaInfo.MediaData[MediaInfo.unCount].eMediaType = MEDIATYPE_LOCAL_RTP_INFO_V6;

            inet_pton(AF_INET6, ipAddress,
                MediaInfo.MediaData[MediaInfo.unCount].mediaInfo.PortInfoV6.bIPv6Address);

            MediaInfo.MediaData[MediaInfo.unCount].mediaInfo.PortInfoV6.unPortId = 0; // must be 0

            ++MediaInfo.unCount;
        }
    }

    if (ipm_StartMedia(nDeviceHandle, &MediaInfo, DATA_IP_TDM_BIDIRECTIONAL , EV_ASYNC) < 0 )
    {
        // process error
        // ...
    }
    // ...
    return NOERROR;
}
```

You can verify the setting of a channel with an the existing feature by calling `ipm_GetLocalMediaInfo` with `MEDIATYPE_LOCAL_RTP_INFO` or `MEDIATYPE_LOCAL_RTP_INFO_V6`.

1.64 dx_setchxfercnt() API Function Support

With this Service Update 71, the Voice API function, **dx_setchxfercnt()**, is supported with the Dialogic® PowerMedia™ HMP for Linux Release 4.1.

1.64.1 Function Information

The **dx_setchxfercnt()** function sets the bulk queue buffer size for a channel. It is typically used in conjunction with the user I/O feature or the streaming to board feature.

For details about this function, refer to the *Dialogic® Voice API Library Reference* available on the documentation bookshelf.

Note: The following Linux-specific cautions are currently not available on the bookshelf version of the *Dialogic® Voice API Library Reference*. These cautions will be added in a future version of that document.

On Dialogic® HMP Interface boards operating in Linux, for bulk queue buffer sizes of 4 Kbytes or more (bufsize_identifier is set to 0 to 3), the number of initial consecutive callbacks invoked depends on the buffer size used. To avoid underrun conditions, it is recommended that you have a minimum of 16 Kbytes of data required to successfully start playback or streaming. For example, for a buffer size of 4 Kbytes, the application UIO routine will get invoked 4 consecutive times initially with no delay. For a buffer size of 8 Kbytes, the application UIO routine will get invoked 2 consecutive times initially with no delay, and so on.

The minimum amount of data to be available also applies to the streaming to board interface. It is recommended that you have 16 Kbytes of data available in the circular stream buffer before streaming begins. This is to ensure that no underrun conditions occur.

On Dialogic® HMP Interface boards operating in Linux, for bulk queue buffer sizes of 2 Kbytes or less (bufsize_identifier is set to 4, 5 or 6), the application UIO routine will get invoked 6 to 7 consecutive times (with no delay) initially for a given buffer size. This means that a sufficient amount of data must be available for the entire sequence of consecutive invocations for successful playback of that data.

The minimum amount of data to be available also applies to the streaming to board interface. For example, if the buffer size is set to 1 Kbyte, it is recommended that you have 6 to 7 Kbytes of data available in the circular stream buffer before streaming begins. This is to ensure that no underrun conditions occur.

1.65 H263+ Support

Service Update 71 announces support for H263+ encoded using the video format transported by the RFC 2190 RTP payload type. This support was available in Service Update 65.

1.65.1 Feature Implementation

The **ipm_StartMedia()** function now supports CODER_TYPE_H263_1998 as a valid value in the eIPM_CODER_TYPE enumeration within the IPM_VIDEO_CODER_INFO data structure. For more information, refer to the *Dialogic® IP Media API Programming Guide and Library Reference*.

Note: When using the CODER_TYPE_H263_1998 value, you must specify a dynamic payload type (i.e., greater than 95) for the unCoderPayloadType field within the IPM_VIDEO_CODER_INFO structure, instead of the 34 used for CODER_TYPE_H263.

For the Multimedia API, both the **mm_Play()** and **mm_Record()** functions now support EMM_VIDEO_CODING_H263_1998 as a valid value for the Coding parameter in the MM_VIDEO_CODEC structure. For more information, refer to the *Dialogic® Multimedia API Programming Guide and Library Reference*.

1.66 Support for Enhanced RTCP Reports

With Service Update 71, the Dialogic® IP Media library supports Quality of Service (QoS) alarms and enhanced RTCP reports (RTCP-XR) compliant with RFC 3611 for RTCP Extended Reports and IETF draft RTCP High Resolution VoIP Metrics Report Blocks (RTCP-HR). For more information about this feature, refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* available on the documentation bookshelf.

1.67 Red Hat Enterprise Linux Release 5 Update 5 Support

Service Update 71 announces support for Red Hat Enterprise Linux Release 5 Update 5. This update has been available since Service Update 53.

1.68 MSML Server Software Support

Service Update 65 adds MSML media server software support with the Dialogic® PowerMedia™ HMP for Linux Release 4.1. Refer to the *Dialogic® MSML Media Server Software User's Guide* for additional information.

1.69 GCC 4.1.1 Compiler Support

Service Update 61 fully supports GNU Compiler Collection (GCC) 4.1.1. This removes the restriction added in Service Update 32. Now, all Dialogic libraries are fully linked to the GCC 4.1.1 libraries.

1.70 Playing and Recording Raw E1/T1 DS0 64 Kbps Bit Streams

Service Update 61 adds support for recording raw E1/T1 Digital Signal 0 (DS0) 64 Kbps bit streams using a voice device.

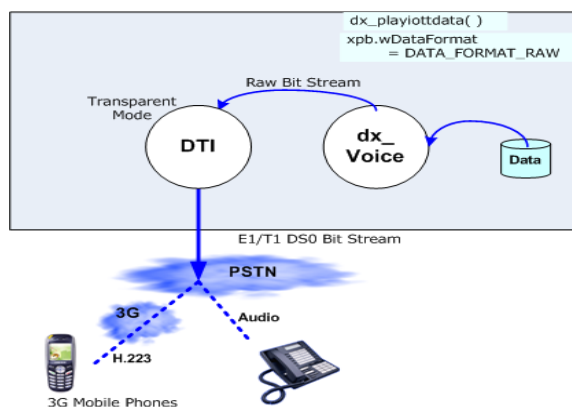
1.70.1 Feature Description

This feature supports monitoring type applications, where the Dialogic® HMP system is used to record the raw bit stream in real-time allowing for offline post call processing and analysis of the recorded data. The bit stream recorded can be audio, H.223, or any type of data that can be carried over an E1/T1 DS0. When using this feature, the DS0 bit stream is unaltered or processed in anyway other than being recorded.

In addition to recording, this feature also provides the ability to use a voice device to play raw data to an E1/T1 DS0 64 Kbps bit stream.

Record RAW Data

To record the raw E1/T1 DS0 bit stream, the incoming DTI device stream is connected to the voice device using the new the **dx_listenttransparent()** function. The following figure shows the raw DS0 record configuration.



In order for the bit stream to pass to the voice device unaltered, the DTI device must also pass the data transparently between the T1/E1 network interface and the system TDM bus (physical and/or soft).

For the Dialogic® DNI/2410TEPEHMP, DNI/1210TEPEHMP, DNI/610TEPEHMP, and DNI/310TEPEHMP boards, data is passed transparently when:

- On-board echo cancellation is disabled. On-board echo cancellation is disabled by default, however, the application can programmatically control it on a per channel basis using the **gc_SetConfigData()** function. Refer to the “Configuring On-Board Echo Cancellation” section in *Dialogic® Host Media Processing Software for Linux Configuration Guide* (previously named the *Dialogic® System Configuration Guide*).
- The system CT Bus encoding setting matches the target T1/E1 encoding setting (i.e., A-law and A-Law). The system PCM encoding is unique for all boards in the system and can be configured using the CLI configuration command `hardware dm3` and the `pcmencoding` parameter. Refer to the “CLI Configure (conf) Command Reference” section in *Dialogic® Host Media Processing Software for Linux Configuration Guide* (previously named the *Dialogic® System Configuration Guide*).
- Bridge devices used to stream data between the physical and soft CT Bus have sufficient data streams in the upstream direction (bh-datastreams) available for the number of raw channels to be recorded. Refer to the “Configuring Device-Level Bridging and Streaming” section in *Dialogic® Host Media Processing Software for Linux Configuration Guide* (previously named the *Dialogic® System Configuration Guide*).

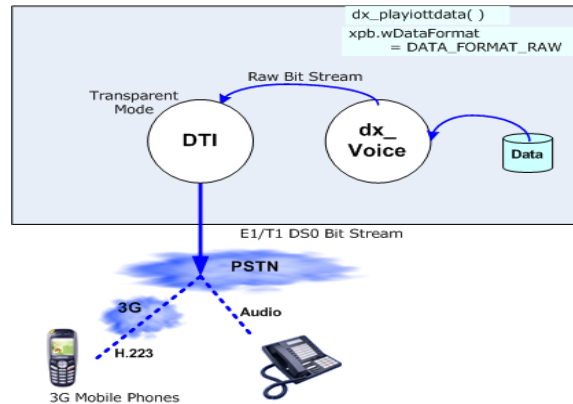
For the Dialogic® DNI/2410AMCTEHMP board, the DTI device must be set to transparent mode. This is accomplished at runtime using the **gc_SetConfigData()** function and setting the `CCPARM_TRANSPARENTMODE` parameter to `CCDM3FW_TRANSPARENTMODE_ENABLE`. On-board echo cancellation must also be disabled as noted above.

Note: Transparent mode can be disabled using the **gc_SetConfigData()** function and setting the `CCPARM_TRANSPARENTMODE` parameter to `CCDM3FW_TRANSPARENTMODE_DISABLE`.

The voice device records the raw bit stream using the **dx_reciottdata()** function with `xpb.wDataFormat = DATA_FORMAT_RAW`.

Play RAW Data

To play the raw data to a E1/T1 DS0 bit stream, the playing voice device is connected to the outgoing DTI device stream using the new **dt_listenttransparent()** function. The following figure shows the raw DS0 play configuration.



The DTI device must also be set to transparent mode so that the bit stream is passed from the voice device unaltered. This is accomplished in the same manner as described above for the voice device. The voice device is then used to play raw data to the bit stream using the **dx_playiottdata()** function with `xpb.wDataFormat = DATA_FORMAT_RAW`. The data is played using the same format used when recording the raw data.

Note: For the Dialogic® DNI/2410TEPEHMP, DNI/1210TEPEHMP, DNI/610TEPEHMP, and DNI/310TEPEHMP boards, ensure that sufficient data streams in the downstream direction (hb-datastreams) are available for the number of raw playback channels. Refer to the “Configuring Device-Level Bridging and Streaming” section in *Dialogic® Host Media Processing Software for Linux Configuration Guide* (previously named the *Dialogic® System Configuration Guide*).

Feature Limitation

When this feature is used to record or play raw data, no additional processing on that bit stream, other than playing or recording, is supported.

1.70.2 Voice API Library Changes

To support this feature, the following new value is added to the `wDataFormat` field in the Voice API `DX_XPB` structure:

DATA_FORMAT_RAW

raw 64 Kbps data format. Data is not encoded nor is any kind of processing performed on the data by the voice channel before it is stored in the recorded file.

Notes: 1. When set to `DATA_FORMAT_RAW`, the other fields in the `DX_XPB` structure are ignored.

2. `wFileFormat FILE_FORMAT_WAVE` is not supported.

In addition, a new function, **dx_listenttransparent()**, is added. Information about this function is provided below:

Name:	int dx_listenttransparent(chdev, sc_tsinfo, mode)	
Inputs:	int chdev	• valid channel device handle
	SC_TSINFO *sc_tsinfo	• pointer to time slot information structure
	unsigned short mode	• mode flag
Returns:	0 if success -1 if failure	
Includes:	srllib.h dxxlib.h	
Category:	TDM Routing	
Mode:	asynchronous or synchronous	

■ Description

The **dx_listenttransparent()** function connects a voice receive channel to a time slot in transparent mode using information stored in the SC_TSINFO data structure. The function then sets up a half-duplex connection. For a full-duplex connection, the receive channel of the other device must be connected to the voice transmit channel.

Parameter	Description
chdev	specifies the voice channel device handle obtained when the channel was opened using dx_open()
sc_tsinfo	specifies a pointer to the SC_TSINFO structure
mode	specifies the mode of operation: EV_SYNC – synchronous mode (default) EV_ASYNC – asynchronous mode

The SC_TSINFO structure is declared as follows:

```
typedef struct {
    unsigned long sc_numts;
    long *sc_tsarrayp;
} SC_TSINFO;
```

The sc_numts member of the SC_TSINFO structure must be set to 1 because a digital receive (listen) channel can connect to only one TDM time slot, even though multiple devices may listen (be connected) to the same TDM time slot. The sc_tsarrayp field of the SC_TSINFO structure must be initialized with a pointer to a valid array. The first element of this array must contain a valid time slot from the CT Bus or HMP soft Bus. Upon return from the **dx_listenttransparent()** function, the digital receive channel will be connected to this TDM time slot in transparent mode.

Use the **dx_unlisten()** function to stop the transparent mode connection.

In asynchronous mode, a TDX_LISTEN event is queued upon successful completion of the routing. If a failure occurs during routing, a TDX_LISTEN_FAIL event is queued. In some limited cases, such as when invalid arguments are passed to the library, the

function may fail before routing is attempted. In such cases, the function returns -1 immediately to indicate failure and no event is queued.

■ Cautions

- This function fails when an invalid channel device handle is specified or when an invalid TDM bus time slot number is specified.
- When using this function in asynchronous mode, do not issue another listen operation on the same channel using either **dx_listen()** or **dx_listenEx()** until the TDX_LISTEN event is received. If you attempt to do this, the listen function will return failure.
- To switch between transparent and non-transparent mode, you have to first call the **dx_unlisten()** function prior to the **dx_listen()** function and/or the **dx_listenttransparent()** function.

■ 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_BADPARAM

Parameter error

EDX_SH_BADCMD

Command is not supported in current bus configuration

EDX_SH_BADEXTTS

TDM bus time slot is not supported at current clock rate

EDX_SH_BADINDX

Invalid Switch Handler index number

EDX_SH_BADLCLTS

Invalid channel number

EDX_SH_BADMODE

Function not supported in current bus configuration

EDX_SH_CMDBLOCK

Blocking command is in progress

EDX_SH_LCLTSCNCT

Channel is already connected to TDM bus

EDX_SH_LIBBSY

Switch Handler library busy

EDX_SH_LIBNOTINIT

Switch Handler library uninitialized

EDX_SH_MISSING

Switch Handler is not present

EDX_SH_NOCLK

Switch Handler clock fallback failed

EDX_SYSTEM
Error from operating system

■ **See Also**

- **dx_listen()**
- **dx_listenEx()**
- **dx_unlisten()**

1.70.3 DTI API Library Changes

A new function, **dt_listenttransparent()**, is added. Information about this function is provided below:

Name:	int dt_listenttransparent (devh, sc_tsinfo)	
Inputs:	int devh	• digital network interface device handle
	SC_TSINFO *sc_tsinfo	• pointer to time slot information structure
Returns:	0 if success -1 if failure	
Includes:	srllib.h dtilib.h	
Category:	SCbus Routing	
Mode:	synchronous	

■ **Description**

The **dt_listenttransparent()** function connects the digital listen channel to the HMP soft CT Bus time slot in transparent mode. This function uses the information stored in the SC_TSINFO structure to connect the digital receive (listen) channel (T1/E1 time slot). This function sets up a half-duplex connection. For a full-duplex connection, the receive (listen) channel of the other device must be connected to the digital transmit channel.

Parameter	Description
devh	specifies the digital network interface device handle obtained when the channel was opened using dt_open()
sc_tsinfo	specifies a pointer to the SC_TSINFO structure

The SC_TSINFO structure is declared as follows:

```
typedef struct {  
    unsigned long sc_numts;  
    long *sc_tsarrayp;  
} SC_TSINFO;
```

The sc_numts member of the SC_TSINFO structure must be set to 1 because a digital receive (listen) channel can connect to only one TDM time slot, even though multiple devices may listen (be connected) to the same TDM time slot. The sc_tsarrayp field of the SC_TSINFO structure must be initialized with a pointer to a valid array. The first element

of this array must contain a valid time slot from the CT Bus or HMP soft Bus. Upon return from the **dt_listenttransparent()** function, the digital receive channel will be connected to this TDM time slot in transparent mode.

Use the **dt_unlisten()** function to stop the transparent mode connection.

■ Cautions

To switch between transparent and non-transparent mode, you have to first call the **dt_unlisten()** function prior to the **dt_listen()** and/or **dt_listenttransparent()** function.

■ Errors

If the function returns -1, use the SRL Standard Attribute function **ATDV_LASTERR()** to obtain the error code or use **ATDV_ERRMSGP()** obtain a descriptive error message. The error codes returned by **ATDV_LASTERR()** are:

EDT_BADBRDERR
board missing or defective

EDT_BADCMDERR
invalid command parameter to driver

EDT_FWERR
firmware returned an error

EDT_INVTS
invalid time slot device handle

EDT_INVMSG
invalid message

EDT_SH_BADLCLTS
invalid local time slot number

EDT_SH_BADEXTTS
external time slot unsupported at current

EDT_SH_BADINDX
invalid switch handler library index number

EDT_SH_BADMODE
invalid switch handler bus configuration

EDT_SH_BADTYPE
invalid local time slot type

EDT_SH_LCLTSCNCT
local time slot is already connected to

EDT_SH_LIBBSY
switch handler library busy

EDT_SH_LIBNOTINIT
switch handler library is not initialized

EDT_SH_MISSING
switch handler is not present

EDT_SH_NOCLK
switch handler clock fallback failed

EDT_SYSTEM
system error

EDT_TMOERR
timed out waiting for reply from firmware

■ **See Also**

- **dt_listen()**
- **dt_unlisten()**

1.70.3.1 **Example of dx_listenttransparent() and dt_listenttransparent()**

```
#include <stdio.h>
#include <srllib.h>
#include <dxxplib.h>
#include <dtilib.h>

main() {
    int dxdev, dtidev;          /* Channel device handles */
    SC_TSINFO sc_tsinfo;       /* Time slot information structure */
    long scts;                  /* TDM bus time slot */

    /* Open dti channel dtiB1T1 */
    if((ipdev = dt_Open("dtiB1T1", NULL)) == -1) {
        /* process error */
    }

    /* Open voice channel dxxxB1C1 */
    if ((dxdev = dx_open("dxxxB1C1", 0)) == -1) {
        /* process error */
    }

    /* Fill in the TDM bus time slot information */
    sc_tsinfo.sc_numts = 1;
    sc_tsinfo.sc_tsarrayp = &scts;

    /* Get transmit time slot of IP channel ipmB1C1*/
    if (dt_getxmitslot(dtidev, &sc_tsinfo) == -1) {
        /* process error */
    }

    /* Connect the receive time slot of voice channel dxxxB1C1
       to the transmit time slot
       ...of DTI channel dtiB1T1
    */
}
```

```

if (dx_listenttransparent(dxdev, &sc_tsinfo, EV_SYNC) == -1) {
    printf("Error message = %s", ATDV_ERRMSGP(dxdev));
}

/* Connect the receive time slot of dti channel dtiB1T1
   to the transmit time slot
   ...of DX channel DxxxB1C1
*/

if (dx_getxmitslot(dxdev, &sc_tsinfo) == -1) {
    /* process error */
}

if (dt_listenttransparent(dtidev, &sc_tsinfo) == -1) {
    printf("Error message = %s", ATDV_ERRMSGP(dxdev));
}
{
    DX_IOTT iott;

# ifdef _WIN32
    iott.io_fhandle=dx_fileopen("play_filename", O_RDWR|O_TRUNC|O_CREAT|O_BINARY, 0666);
#else
    // linux code
    iott.io_fhandle=open("play_filename", O_RDWR|O_TRUNC|O_CREAT|O_BINARY, 0666);
# endif

    iott.io_type = IO_DEV | IO_EOT;
    iott.io_offset = 0
    iott.io_length = -1
    iott.filetype = 0;

    DX_XPB xpb;
    xpb.wFileFormat = FILE_FORMAT_VOX ;    /* file format */
    xpb.wDataFormat = DATA_FORMAT_RAW;    /* data encoding */
    xpb.nSamplesPerSec = DRT_8KHZ ; /* sampling rate */
    xpb.wBitsPerSample = 8; /* bits per sample */
/* Play file in raw mode */
dx_playiottdata(dxdev, &iott,0,&xpb,EV_ASYNC);
/* wait for TDX_PLAY */
close(iott.fhandle);

/* Record file in raw mode */
# ifdef _WIN32
    iott.io_fhandle=dx_fileopen("rec_filename", O_RDWR|O_BINARY, 0666);
#else
    // linux code
    iott.io_fhandle=open("rec_filename", O_RDWR|O_BINARY, 0666);
# endif

    DV_TPT tpt;
    /* Set to terminate play in 50 sec */
    tpt.tp_type = IO_EOT;
    tpt.tp_termno = DX_MAXTIME;
    tpt.tp_length = 50;
    tpt.tp_flags = TF_MAXTIME;

    dx_reciottdata(dxdev, &iott,&tpt,&xpb,EV_ASYNC);
/* wait for TDX_RECORD */
close(iott.fhandle);
}

```

1.71 IPv6 Support

Service Update 61 introduces Internet Protocol Version 6 (IPv6) support for open media connections.

1.71.1 Feature Description

The key features of IPv6 are:

- Simplified header format
- Expanded addressing capabilities (128 bits long, compared to 32 bits in IPv4)
- Stateless and stateful address configuration
- Built-in security support
- Improved support for quality of service (QoS)
- New protocol for neighboring node interaction
- Improved support for extensions and options

Applications can be configured to run in a pure IPv4, IPv6, or a mixed (both IPv4 and IPv6 addressing) network environment.

Limitations

Support for IPv6 addressing with Dialogic® PowerMedia™ HMP for Linux Release 4.1 is implemented in phases. The initial phase, Phase 1, provides IPv6 support for bearer channel (multimedia) connections as well as for OA&M interfaces (CLI). Known limitations for Phase 1 are:

- IPv6 supports Audio/Video only. No Fax or NBUP support is provided.
- CLI (Telnet) may only be reached via an IPv4 address.

These limitations will be addressed in a subsequent release update.

1.71.2 Enhanced API and Data Structures

The following IP Media API library functions and data structures were modified to accept IPv4 and IPv6 addresses when creating a media connection. A new message was also added to transport the IPv6 address to the firmware for proper socket operations.

- IP_PARM_INFO
- IP_MEDIA
- IP_MODIFY_MEDIA
- **ipm_StartMedia()**
- **ipm_GetLocalMediaInfo()**

- **ipm_ModifyMedia()**

Note: IPv4 and IPv6 cannot be mixed within the same channel. That is, only one mode can be used at a time. For example, when calling the **ipm_GetLocalMediaInfo()** function, users cannot retrieve IPv4 and IPv6 local information at the same time.

IPM_PARM_INFO is used to set or retrieve parameters for an IP channel. The following enum was added for the IPM_PARM_INFO structure for IPv6 support of ToS:

eIPM_PARM Define	Description and Values
PARMCH_TOS_V6	Traffic Class field replace Type of Service (TOS) in IPv4. Type: char. Valid values: 0 to 255. Default: 0.

Example

The following example demonstrates setting the Type of Service (TOS) Value for IPv6:

```
#include <stdio.h>
#include <srllib.h>
#include <ipmlib.h>
void CheckEvent();
typedef long int(*HDLR)(unsigned long);
void main()
{
    int nDeviceHandle;
    // Register event handler function with srl
    sr_enbhdlr( EV_ANYDEV ,EV_ANYEVT , (HDLR)CheckEvent);
    /*
     * Main Processing
     */
    /*
     * ASSUMPTION: A valid nDeviceHandle was obtained from prior
     * call to ipm_Open().
     */
    IPM_PARM_INFO ParmInfo;
    unsigned long ulParmValue = 10;
    ParmInfo.eParm = PARMCH_TOS_V6;
    ParmInfo.pvParmValue = &ulParmValue;
    if(ipm_SetParm(nDeviceHandle, &ParmInfo, EV_ASYNC)==-1)
    {
        printf("ipm_SetParm failed for device name %s with error = %d\n",
            ATDV_NAMEP(nDeviceHandle), ATDV_LASTERR(nDeviceHandle));
        /*
         * Perform Error Processing
         */
    }
    /*
     * continue
     */
}
void CheckEvent()
{
    int nEventType = sr_getevtttype();
    int nDeviceID = sr_getevtdev();
    void* pVoid = sr_getevtdatap();
    switch(nEventType)
    {
        /*
         * Other events
         */
    }
}
```

```

        */
        /* Expected reply to ipm_GetQoSAlarmStatus */
        case IPMEV_SET_PARM:
        printf("Received IPMEV_SETPARM for device = %s\n",
        ATDV_NAMEP(nDeviceID));
        break;
        default:
        printf("Received unknown event = %d for device = %s\n",
        nEventType, ATDV_NAMEP(nDeviceID));
        break;
    }
}

```

A new data structure, `IPM_PORT_INFO_V6`, was added to the IP Media API library. It is a child of `IPM_MEDIA`, which is a child of the `IPM_MEDIA_INFO` structure that is used by the `ipm_GetLocalMediaInfo()`, `ipm_ModifyMedia()` and `ipm_StartMedia()` functions. As a result, `IPM_MEDIA` has been redefined as follows:

```

struct IPM_MEDIA_tag
{
    eIPM_MEDIA_TYPE eMediaType;
    union
    {
        IPM_PORT_INFO PortInfo;           /* RTP port information */
        IPM_PORT_INFO_V6 PortInfoV6;      /* RTP port information for Ipv6 */
        IPM_AUDIO_CODER_INFO AudioCoderInfo; /* Audio coder information */
        IPM_FAX_SIGNAL FaxSignal;          /* Fax signal information */
        IPM_VIDEO_CODER_INFO VideoCoderInfo /* Video coder information */
        IPM_SECURITY_INFO SecurityInfo     /* Security information */
        IPM_AUDIO_CODER_OPTIONS_INFO AudioCoderOptionsInfo;
        /* Extended Audio Coder Information */
        IPM_NBUP_PROFILE_INFO NBUPProfileInfo; /* NBUP Profile Information */
    } mediaInfo;
} IPM_MEDIA, *PIPM_MEDIA;

```

The following values (types) were added to the `eMediaType` field:

- `MEDIATYPE_AUDIO_REMOTE_RTP_INFO_V6` - remote RTP IPv6 audio port information
- `MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6` - local RTP IPv6 audio port information
- `MEDIATYPE_AUDIO_REMOTE_RTCP_INFO_V6` - remote RTCP IPv6 audio port information
- `MEDIATYPE_AUDIO_LOCAL_RTCP_INFO_V6` - local RTCP IPv6 audio port information

The new `IPM_PORT_INFO_V6` data structure contains RTP, RTCP, and T.38 UDP port properties when using IPv6. It is defined as follows:

```

typedef struct ipm_port_info_v6_tag
{
    unsigned int unPortId;           /* The Port ID */
    char bIPv6Address[ IPV6_ADDR_SIZE ]; /* IPv6 Address */
} IPM_PORT_INFO_V6, *PIPM_PORT_INFO_V6;

```

The fields of the `IPM_PORT_INFO_V6` data structure are:

`unPortId`
port identifier

bIPv6Address[IPv6_ADDR_SIZE]

IPv6 address of the port in binary format. To convert from string to binary format, use the standard API `inet_pton()`. `inet_ntop()` can be used to convert from binary to string format.

The following example demonstrates setting the media properties (remote IPv6 address), and starting the session using the **ipm_StartMedia()** function:

```
#include <stdio.h>
#include <string>
#include <srllib.h>
#include <ipmlib.h>
typedef long int(*HDLR)(unsigned long);
void CheckEvent();
void main()
{
    int nDeviceHandle, nMediaCnt = 0;
    // Register event handler function with srl
    IPM_MEDIA_INFO MediaInfo;
    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().
     */
    /* clear the RemoteMediaInfo structures */
    memset(&MediaInfo, 0, sizeof(IPM_MEDIA_INFO));
    /* remote audio */
    MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ULAW64K;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.eFrameSize =
        (eIPM_CODER_FRAMESIZE)m_nAudioFrameSize;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.unFramesPerPkt = 1;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.eVadEnable = CODER_VAD_DISABLE;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.unCoderPayloadType = 0;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.unRedPayloadType = 0;
    /* local audio */
    ++nMediaCnt;
    MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_INFO;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ULAW64K;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.eFrameSize =
        (eIPM_CODER_FRAMESIZE)m_nAudioFrameSize;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.unFramesPerPkt = 1;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.eVadEnable = CODER_VAD_DISABLE;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.unCoderPayloadType = 0;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.CoderInfo.unRedPayloadType = 0;
    /* remote video */
    ++nMediaCnt;
    MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_VIDEO_REMOTE_CODER_INFO;
    INIT_IPM_VIDEO_CODER_INFO(&MediaInfo.MediaData[nMediaCnt].mediaInfo.VideoCoderInfo);
    MediaInfo.MediaData[nMediaCnt].mediaInfo.VideoCoderInfo.eCoderType = VIDEO_CODING_H263;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.VideoCoderInfo.unCoderPayloadType = 100;
    /* local video coder */
    ++nMediaCnt;
    MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_VIDEO_REMOTE_CODER_INFO;
    INIT_IPM_VIDEO_CODER_INFO(&MediaInfo.MediaData[nMediaCnt].mediaInfo.VideoCoderInfo);
    MediaInfo.MediaData[nMediaCnt].mediaInfo.VideoCoderInfo.eCoderType = VIDEO_CODING_H263;
    MediaInfo.MediaData[nMediaCnt].mediaInfo.VideoCoderInfo.unCoderPayloadType = 100;
    /* remote audio port */
    ++nMediaCnt;
    MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTP_INFO_V6;
    inet_pton(AF_INET6,
        MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.bIPv6Address,
```

```

"fe80::2e0:18ff:fed9:9205");
MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.unPortId = 2328;
++nMediaCnt;
MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTCP_INFO_V6;
inet_pton(AF_INET6,
MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.bIPv6Address,
"fe80::2e0:18ff:fed9:9205");
MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.unPortId = 2329;
/* remote video port */
++nMediaCnt;
MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_VIDEO_REMOTE_RTP_INFO_V6;
inet_pton(AF_INET6,MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.bIPv6Address,
"fe80::2e0:18ff:fed9:9205");
MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.unPortId = 2340;
++nMediaCnt;
MediaInfo.MediaData[nMediaCnt].eMediaType = MEDIATYPE_VIDEO_REMOTE_RTCP_INFO_V6;
inet_pton(AF_INET6,
MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.bIPv6Address,
"fe80::2e0:18ff:fed9:9205");
MediaInfo.MediaData[nMediaCnt].mediaInfo.PortInfoV6.unPortId = 2341;
MediaInfo.unCount = nMediaCnt + 1;
if(ipm_StartMedia(nDeviceHandle, &MediaInfo, DATA_IP_TDM_BIDIRECTIONAL, 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;
}
}

```

The **ipm_GetLocalMediaInfo()** function retrieves properties for the local media channel. To retrieve RTP port information for both audio and video in a multimedia session, set the eMediaType fields to MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6 and MEDIATYPE_VIDEO_LOCAL_RTP_INFO_V6 respectively and unCount to 2.

To retrieve RTP port information for a video session only, set the eMediaType fields to MEDIATYPE_VIDEO_LOCAL_RTP_INFO_V6 respectively and unCount to 1.

To retrieve RTP port information for an audio session only, set the eMediaType fields to MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6 respectively and unCount to 1.

For audio and /or video media types, only RTP information needs to be queried. RTCP information is automatically provided when querying for RTP.

The following example demonstrates changing the coder from G.711 Mu-law to G.711 A-law, and changing the direction and remote IPv6 address using the ipm_ModifyMedia() function:

```
#include <stdio.h>
#include <string>
#include <srllib.h>
#include <ipmlib.h>
typedef long int (*HDLR) (unsigned long);
void CheckEvent();
void main()
{
    /*
     * 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 = 4;

    MediaInfo.MediaData[0].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTP_INFO_V6;
    MediaInfo.MediaData[0].mediaInfo.PortInfoV6.unPortId = 2328;
    inet_pton(AF_INET6,
    MediaInfo.MediaData[0].mediaInfo.PortInfoV6.bIPv6Address, "fe80::2e0:18ff:fed9:9205");

    MediaInfo.MediaData[1].eMediaType = MEDIATYPE_AUDIO_REMOTE_RTCP_INFO_V6;
    MediaInfo.MediaData[1].mediaInfo.PortInfoV6.unPortId = 2329;
    inet_pton(AF_INET6,
    MediaInfo.MediaData[1].mediaInfo.PortInfoV6.bIPv6Address, "fe80::2e0:18ff:fed9:9205");

    MediaInfo.MediaData[2].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
    MediaInfo.MediaData[2].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ULAW64K;
    MediaInfo.MediaData[2].mediaInfo.CoderInfo.eFrameSize = (eIPM_CODER_FRAMESIZE) 30;
    MediaInfo.MediaData[2].mediaInfo.CoderInfo.unFramesPerPkt = 1;
    MediaInfo.MediaData[2].mediaInfo.CoderInfo.eVadEnable = CODER_VAD_DISABLE;
    MediaInfo.MediaData[2].mediaInfo.CoderInfo.unCoderPayloadType = 0;
    MediaInfo.MediaData[2].mediaInfo.CoderInfo.unRedPayloadType = 0;
    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) 30;
    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;
    if (ipm_StartMedia(nDeviceHandle, &MediaInfo, DATA_IP_TDM_BIDIRECTIONAL, EV_SYNC) == -1)
    {
        printf("ipm_StartMediaInfo failed for device name = %s with error = %d\n",
        ATDV_NAMEP(nDeviceHandle), ATDV_LASTERR(nDeviceHandle));
        /*
         Perform Error Processing
         */
    }
    /*
    Continue processing
    */
    MediaInfo.unCount = 2;
```



```

MediaInfo.MediaData[0].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
MediaInfo.MediaData[0].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ALAW64K;
MediaInfo.MediaData[0].mediaInfo.CoderInfo.eFrameSize = (eIPM_CODER_FRAMESIZE) 30;
MediaInfo.MediaData[0].mediaInfo.CoderInfo.unFramesPerPkt = 1;
MediaInfo.MediaData[0].mediaInfo.CoderInfo.eVadEnable = CODER_VAD_DISABLE;
MediaInfo.MediaData[0].mediaInfo.CoderInfo.unCoderPayloadType = 0;
MediaInfo.MediaData[0].mediaInfo.CoderInfo.unRedPayloadType = 0;
MediaInfo.MediaData[1].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_INFO;
MediaInfo.MediaData[1].mediaInfo.CoderInfo.eCoderType = CODER_TYPE_G711ALAW64K;
MediaInfo.MediaData[1].mediaInfo.CoderInfo.eFrameSize = (eIPM_CODER_FRAMESIZE) 30;
MediaInfo.MediaData[1].mediaInfo.CoderInfo.unFramesPerPkt = 1;
MediaInfo.MediaData[1].mediaInfo.CoderInfo.eVadEnable = CODER_VAD_DISABLE;
MediaInfo.MediaData[1].mediaInfo.CoderInfo.unCoderPayloadType = 0;
MediaInfo.MediaData[1].mediaInfo.CoderInfo.unRedPayloadType = 0;
if (ipm_ModifyMedia(nDeviceHandle, &MediaInfo, DATA_IP_SENDFONLY, EV_SYNC) == -1)
{
    printf("ipm_Modify failed for device name = %s with error = %d\n",
        ATDV_NAMEP(nDeviceHandle), ATDV_LASTERR(nDeviceHandle));
    /*
    Perform Error Processing
    */
}
/*
continue processing
*/
}

```

The following example demonstrates retrieving the media properties, including local and remote addresses using the **ipm_GetLocalMediaInfo()** function.

```

#include <stdio.h>
#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
    */
    /*
    Get the local IP information for IP device handle, nDeviceHandle.
    ASSUMPTION: A valid nDeviceHandle was obtained from prior call to ipm_Open().
    */
    IPM_MEDIA_INFO MediaInfo;
    MediaInfo.unCount = 2;
    MediaInfo.MediaData[0].eMediaType = MEDIATYPE_VIDEO_LOCAL_RTP_INFO_V6;
    MediaInfo.MediaData[1].eMediaType = MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6;
    if(ipm_GetLocalMediaInfo(nDeviceHandle, &MediaInfo, EV_ASYNC) == -1)
    {
        printf("ipm_GetLocalMediaInfo failed for device name %s with error = %d\n",
            ATDV_NAMEP(nDeviceHandle), ATDV_LASTERR(nDeviceHandle));
        /*
        Perform Error Processing
        */
    }
    /*
    Continue processing
    */
}

void CheckEvent()

```

```

{
    unsigned int i;
    int nDeviceID = sr_getevtdev();
    int nEventType = sr_getevttype();
    void* pVoid = sr_getevtdatap();
    IPM_MEDIA_INFO* pMediaInfo;
    Char cAddr6[40]={0};

    switch(nEventType)
    {
        /*
        Other events
        */
        /* Expected reply to ipm_GetLocalMediaInfo */
        case IPMEV_GET_LOCAL_MEDIA_INFO:
            printf("Received IPMEV_GET_LOCAL_MEDIA_INFO_V6 for device name = %s\n",
                ATDV_NAMEP(nDeviceID));
            pMediaInfo = (IPM_MEDIA_INFO*)pVoid;
            for(i=0; i<pMediaInfo->unCount; i++)
            {
                switch(pMediaInfo->MediaData[i].eMediaType)
                {
                    case MEDIATYPE_VIDEO_LOCAL_RTP_INFO:
                        printf("MediaType=MEDIATYPE_VIDEO_LOCAL_RTP_INFO\n");
                        printf("PortId=%d\n",pMediaInfo->MediaData[i].mediaInfo.PortInfo.unPortId);
                        printf("IP=%s\n",pMediaInfo->MediaData[i].mediaInfo.PortInfo.cIPAddress);
                        break;
                    case MEDIATYPE_VIDEO_LOCAL_RTP_INFO_V6:
                        printf("MediaType=MEDIATYPE_VIDEO_LOCAL_RTP_INFO_V6\n");
                        printf("PortId=%d\n",pMediaInfo->MediaData[i].mediaInfo.PortInfoV6.unPortId);
                        inet_ntop(AF_INET6,MediaInfo->MediaData[i].mediaInfo.PortInfoV6.bIPv6Address, cAddr6,
                            sizeof(cAddr6));
                        printf("IP=%s\n",.cAddr6);
                        break;
                    case MEDIATYPE_VIDEO_LOCAL_RTCP_INFO:
                        printf("MediaType=MEDIATYPE_VIDEO_LOCAL_RTCP_INFO\n");
                        printf("PortId=%d\n",pMediaInfo->MediaData[i].mediaInfo.PortInfo.unPortId);
                        printf("IP=%s\n",pMediaInfo->MediaData[i].mediaInfo.PortInfo.cIPAddress);
                        break;
                    case MEDIATYPE_VIDEO_LOCAL_RTCP_INFO_V6:
                        printf("MediaType=MEDIATYPE_VIDEO_LOCAL_RTCP_INFO_V6\n");
                        printf("PortId=%d\n",pMediaInfo->MediaData[i].mediaInfo.PortInfoV6.unPortId);
                        inet_ntop(AF_INET6, pMediaInfo->MediaData[i].mediaInfo.PortInfoV6.bIPv6Address, cAddr6,
                            sizeof(cAddr6));
                        printf("IP=%s\n",cAddr6);
                        break;
                    case MEDIATYPE_AUDIO_LOCAL_RTP_INFO:
                        printf("MediaType=MEDIATYPE_AUDIO_LOCAL_RTP_INFO\n");
                        printf("PortId=%d\n",pMediaInfo->MediaData[i].mediaInfo.PortInfo.unPortId);
                        printf("IP=%s\n",pMediaInfo->MediaData[i].mediaInfo.PortInfo.cIPAddress);
                        break;
                    case MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6:
                        printf("MediaType=MEDIATYPE_AUDIO_LOCAL_RTP_INFO V6\n");
                        printf("PortId=%d\n",pMediaInfo->MediaData[i].mediaInfo.PortInfoV6.unPortId);
                        inet_ntop(AF_INET6, pMediaInfo->MediaData[i].mediaInfo.PortInfoV6.bIPv6Address, cAddr6,
                            sizeof(cAddr6));

```

```

        sizeof(cAddr6));
        printf("IP=%s\n", cAddr6);
        break;
    case MEDIATYPE_AUDIO_LOCAL_RTP_INFO:
        printf("MediaType=MEDIATYPE_AUDIO_LOCAL_RTP_INFO\n");
        printf("PortId=%d\n", pMediaInfo->MediaData[i].
            mediaInfo.PortInfo.unPortId);
        printf("IP=%s\n", pMediaInfo->MediaData[i].
            mediaInfo.PortInfo.cIPAddress);
        break;
    case MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6:
        printf("MediaType=MEDIATYPE_AUDIO_LOCAL_RTP_INFO_V6\n");
        printf("PortId=%d\n", pMediaInfo->MediaData[i].
            mediaInfo.PortInfoV6.unPortId);
        inet_ntop(AF_INET6, pMediaInfo->MediaData[i].
            mediaInfo.PortInfoV6.bIPv6Address, cAddr6,
            sizeof(cAddr6));
        printf("IP=%s\n", cAddr6);
        break;
    }
    break;
default:
    printf("Received unknown event = %d for device name = %s\n",
        nEventType, ATDV_NAMEP(nDeviceID));
    break;
}
}

```

1.71.3 OA&M CLI Changes

In most cases, the statistical data for IPv6 follows IPv4 data within the same existing commands. For example, the command “show ip route” will show routing information for IPv4 followed by the routing information for IPv6.

Several OA&M commands were updated to support the entry of a typical eight-field hexadecimal address. Examples of the impacted CLI commands and new display output are provided in this section.

Note: The CLI Agent application can only be called with an IPv4 address, however, both IPv4 and IPv6 statistics and settings are displayed shown.

CLI> conf system hmp-rtp-address

The "conf system hmp-rtp-address" command is used to set an IPv6 address. Since an ethernet device supports multiple IPv6 addresses, it is best to specify the specific IPv6 address to use for RTP. If an IPv4 address is selected as the HMP RTP address, then any one of the IPv6 addresses associated to the ethernet device will be selected. The selected HMP RTP IP addresses can be seen with the "show system" command. When the HMP RTP IPv6 address is set, the associated IPv4 address is automatically set.

The IPv6 address can be entered in either a non-compressed or compressed format. The following example shows how to set an IPv6 address in non-compressed format:

```
CLI> conf system hmp-rtp-address fe80:0000:0000:0000:02a0:ccff:fe29:e96b
```

The following example shows how to set an IPv6 address in compressed format:

```
CLI> conf system hmp-rtp-address fe80::2a0:ccff:fe29:e96b
```

1.71.3.1 CLI Show Command

The CLI `show` command is used to view any/all configuration parameters and statistics. Changes made for IPv6 implementation are noted below.

CLI> show icmp

This subcommand displays Internet Control Message Protocol information.

IPv4 display:

```
ICMP Statistics (IPv4)
Rcvd
  Total Messages: 73
  Total Errors: 0
  Total Destination Unreachable Messages: 48
  Total Time Exceeded Messages: 9
  Total Parameter Problem Messages: 0
  Total Source Quench Messages: 0
  Total Redirect Messages: 0
  Total Echo (request) Messages: 3
  Total Echo Reply Messages: 13
  Total Timestamp (request) Messages: 0
  Total Timestamp Reply Messages: 0
  Total Address Mask request Messages: 0
  Total Address Mask reply Messages: 0
Sent
  Total Messages: 39
  Total Errors: 0
  Total Destination Unreachable Messages: 36
  Total Time Exceeded Messages: 0
  Total Parameter Problem Messages: 0
  Total Source Quench Messages: 0
  Total Redirect Messages: 0
  Total Echo (request) Messages: 0
  Total Echo Reply Messages: 3
  Total Timestamp (request) Messages: 0
  Total Timestamp Reply Messages: 0
  Total Address Mask request Messages: 0
  Total Address Mask reply Messages: 0
```

IPv6 display:

```
ICMP Statistics (IPv6)
In
  Total Messages: 73
  Total Errors: 0
  Total Destination Unreachable Messages: 48
  Total Packets too big: 0
  Total Time Exceeded Messages: 9
  Total Parameter Problem Messages: 0
  Total Echos: 18
  Total Echo Replies: 7
  Total Group Memb Queries: 0
  Total Group Memb Responses: 0
  Total Group Memb Reductions: 0
  Total Router Solicits: 0
  Total Neighbor Solicits: 2
```

```

Total Neighbor Advertisements      2
Total Redirects:                    0

Out
Total Messages:                     39
Total Destination Unreachable Messages: 36
Total Packets too big:              0
Total Time Exceeded Messages:       9
Total Parameter Problem Messages:    0
Total Echo Replies:                 7
Total Router Solicits:              0
Total Neighbor Solicits:            2
Total Neighbor Advertisements       2
Total Redirects:                    0
Total Group Memb Responses:          0
Total Group Memb Reductions:         0

```

Note that many of the items are different between version IPv4 and IPv6 for ICMP. These differences will be explained in a subsequent Release Update.

CLI> show system

This subcommand displays system information with the added HMP RTP IPv6 address.

```

UNKNOWN RMS System
Name:      System Name Not Defined
Location:  System Location Not Defined
Contact:   System Contact Not Defined

Operational Status:  System is not ready
Run Status:          Inactive
Media Last Request:  Stop
Media Start Mode:    Manual
Software Media Support: Refer to license details

Uptime:              4 day(s) 2 hour(s) 49 minute(s) 4 second(s)
Last Boot Reason:    Power on
Services:            7
Form Factor:         RMS
Model:               Dialogic(R) Host Media Processing Software Release 4.1 LIN
Vendor:              Dialogic Research Inc.
Serial Number:       Undefined
HMP RTP Address:     10.10.20.5
HMP RTP IPv6 Address: fe80::211:43ff:fe06:7964

```

CLI> show tcp brief

This subcommand displays the status of TCP connections in the brief format for both IPv4 and IPv6. In addition to the IPv4 TCP connections, the following IPv6 information is shown:

ConnID	Local Address	Remote Address	State
0	[::]:22	[::]:22	LISTEN
1	::ffff:10.130.1.52]:22	::ffff:10.130.1.52]:22	ESTABLISHED

CLI> show udp brief

This subcommand displays the status of UDP connections in the brief format for both IPv4 and IPv6. In addition to the IPv4 TCP connections, the following IPv6 information is shown:

```
IPv6 UDP Listeners
ConnID  Local Address
0       [::]:53785
1       [::]:5353
2       [fe80::210:18ff:fe60:621]:123
3       [fe80::2a0:ccff:fe29:e96b]:123
4       [::1]:123
5       [::]:123
```

CLI> show ip addr

The ip parameter shows global IP configuration subcommands. The ip addr parameter shows the IP address table with the added IPv6 address.

Interface	Ifindex	IP Address	Netmask	IPv6 Address
eth 0	1	10.10.20.5	255.255.224.0	fe80::211:43ff:fe06:7964
eth 1	2	10.10.20.211	255.255.224.0	fe80::212:17ff:fef2:282b

CLI> show ip interface

This subcommand, also known as `show interface`, displays interface information. The IPv6 address is added to the interface information.

```
Ethernet Interface(s):
Name: eth0
  Admin Status:      up
  Current State:     up
  Type:              Base
  Ifindex:           1
  Description:       eth0
  Interface Trap:    enabled
  Last Change:       0 seconds
  Hardware Address:  00:11:43:06:79:64
  IP Address:        10.10.20.5
  IPv6 Address:      fe80::211:43ff:fe06:7964
  Netmask:           255.255.224.0
  Broadcast Bit:     1
  Reassemble Max:    0
  MTU:               1500 bytes
  Speed:             10000000
Rcvd
  Octets:              50173830
  Subnetwork Unicast Packets: 423614
  Subnetwork Broadcast/Multicast Packets: 0
  Discarded Packets:   0
  Error Packets:       0
  Unknown/Unsupported Protocol Packets: 0
Sent
  Octets:              2841364
  Subnetwork Unicast Packets: 21065
  Subnetwork Broadcast/Multicast Packets: 0
  Discarded Packets:   0
  Error Packets:       0
  Packet Queue Length: 100
```

CLI> show ip route

The ip route parameter shows the IP route table entries.

IPv4 display:

Destination	Dev	Netmask	M	NextHop	Type	Prot	Age
10.10.0.0	eth0	255.255.224.0	-	0.0.0.0	Dir	loc	0
10.10.0.0	eth1	255.255.224.0	-	0.0.0.0	Dir	loc	0
169.254.0.0	eth0	255.255.0.0	-	0.0.0.0	Dir	loc	0
0.0.0.0	eth0	0.0.0.0	-	10.10.0.5	Ind	loc	0

IPv6 display:

Destination	Dev	Next Hop
fe800000000000000000000000000000/64	eth0	00000000000000000000000000000000
fe800000000000000000000000000000/64	eth1	00000000000000000000000000000000
ff000000000000000000000000000000/8	eth0	00000000000000000000000000000000
ff000000000000000000000000000000/8	eth1	00000000000000000000000000000000

Show ip traffic

The ip traffic parameter shows all IP traffic data.

```
IPv4 Statistics
Forwarding:    disabled
Default ttl:   64
Rcvd
  Total Datagrams: 526382
  Delivered:      524342
  Header Errors:  0
  Address Errors: 2040
  Unknown Protos: 0
  Discards:      0
Sent
  Out Requests:   271653
  Route Forward Datagrams: 0
  Discards:      0
  Route Failures: 0
Frgs
  Fragmented:
  Created:      0
  Failed:       0
Reasm
  Reassembled:   0
  Requests:      0
  Timeouts:     0
  Fails:         0

IPv6 Statistics
In
  Receives:      342
  Header Errors: 0
  Too Big Errors: 0
  No Routes:     0
  Address Errors: 0
  Unknown Protos: 0
  Truncated Packets: 0
  Discards:      0
  Delivers:      342
Out
  Forward Datagrams: 0
```

```

Requests:          4313
Discards:          0
No Routes:         0
Re-assemble
Time outs:         0
Required:          0
OKs:               0
Fails:             0
Frag
OKs:               18
Fails:             0
Creates:           36
Multicast
In Packets:        261
Out Packets:        88

```

IPv4 ICMP Statistics

Rcvd

```

Total Messages:          19
Total Errors:            0
Total Destination Unreachable Messages: 19
Total Time Exceeded Messages: 0
Total Parameter Problem Messages: 0
Total Source Quench Messages: 0
Total Redirect Messages: 0
Total Echo (request) Messages: 0
Total Echo Reply Messages: 0
Total Timestamp (request) Messages: 0
Total Timestamp Reply Messages: 0
Total Address Mask request Messages: 0
Total Address Mask reply Messages: 0

```

Sent

```

Total Messages:          12
Total Errors:            0
Total Destination Unreachable Messages: 12
Total Time Exceeded Messages: 0
Total Parameter Problem Messages: 0
Total Source Quench Messages: 0
Total Redirect Messages: 0
Total Echo (request) Messages: 0
Total Echo Reply Messages: 0
Total Timestamp (request) Messages: 0
Total Timestamp Reply Messages: 0
Total Address Mask request Messages: 0
Total Address Mask reply Messages: 0

```

IPv6 ICMP Statistics

In

```

Total Messages:          49
Total Errors:            0
Total Destination Unreachable Messages: 27
Total Packets Too Big:   0
Total Time Exceeded Messages: 0
Total Parameter Problem Messages: 0
Total Echos:             0
Total Echo Replies:      0
Total Group Member Queries: 0
Total Group Member Responses: 19
Total Group Member Reductions: 0
Total Router Solicits:   0
Total Router Advertisements: 0
Total Neighbor Solicits: 1
Total Neighbor Advertisements: 2
Total Redirects:         0

```

Out

```

Total Messages:          102
Total Destination Unreachable Messages: 56

```



```

Total Packets Too Big:          0
Total Time Exceeded Messages:   0
Total Parameter Problem Messages: 0
Total Echos:                    0
Total Echo Replies:             0
Total Group Member Queries:     0
Total Group Member Responses:   0
Total Group Member Reductions:  0
Total Router Solicits:          6
Total Router Advertisements:    0
Total Neighbor Solicits:        31
Total Neighbor Advertisements:  1
Total Redirects:                0

TCP Statistics
Rcvd:
  Total segments:                503027
  Total tcp checksum errors:     0
Sent:
  Total segments:                268927
  Total retransmit segments:     3
  Total RST segments:            253

Maximum allowed connections (-1 indicates dynamic): 4294967295
Total CLOSED to SYN-SENT transitions: 747
Total LISTEN to SYNC-RCVD transistions: 177
Total SYN-SENT/SYN-RCVD to CLOSED and
  SYN-RCVD to LISTEN transitions: 159
Total ESTABLISHED/CLOSE-WAIT to CLOSE transitions: 5
Current ESTABLISHED/CLOSE-WAIT connections: 32
Unacknowledge octet retransmit timeout algorithm: Other
Minimum retransmission timeout in ms: 200
Maximum retransmission timeout in ms: 120000

UDP Statistics
Rcvd
  Total datagrams:                18235
  Total no port errors:           12
  Total errors other than no port: 0
Sent
  Total datagrams:                2233

```

Details and examples for future updates to CLI commands and display output will be documented in a subsequent Release Update.

1.72 Support for Dialogic® DNI/300TEPHMPW, DNI/601TEPHMPW, and DNI/1200TEPHMPW Boards

As of Service Update 56, the following Dialogic® HMP Interface Boards (DNI Boards) are supported:

- DNI/300TEPHMPW, Single T-1/E-1 interface
- DNI601TEPHMPW, Dual T-1/E-1 interfaces with on board tone, G.168 echo cancellation and call progress analysis
- DNI/1200TEPHMPW, Quad E1/T1 interfaces

These boards are high performance digital network interface boards in a full-length PCI form factor. For technical specifications, refer to the http://www.dialogic.com/products/ip_enabled/hmp_enabled_boards.htm.

Note: Refer to the Installation Guide (*Dialogic® Quick Install Card*) that is provided with each product for important installation information.

1.73 Increase in Channels Density for Conferencing and CSP

Service Update 56 increases support for up to 1200 channels of conferencing and 500 channels of continuous speech processing (CSP). Refer to the *Dialogic® Conferencing API Programming Guide and Library Reference* for information about using conferencing, and the *Dialogic® Continuous Speech Processing API Library Reference* and *Dialogic® Continuous Speech Processing API Programming Guide* for information about using CSP.

1.74 Recording and Playback of G.729A Files

With Service Update 56, the Voice API library supports record and playback of G.729A audio files in a Microsoft WAV file format.

1.74.1 API Implementation

To take advantage of this feature, the application needs to specify the following fields in the DX_XPB structure used in both the **dx_playiottdata()** and **dx_reciottdata()** functions.

```
xpb.wFileFormat = FILE_FORMAT_WAVE;  
xpb.wDataFormat = DATA_FORMAT_G729A;  
xpb.nSamplesPerSec = DRT_8KHZ;  
xpb.wBitsPerSample = 8
```

Refer to the *Dialogic® Voice API Library Reference* for more information about the DX_XPB data structure.

1.75 Virtualization Support

Service Update 53 introduces initial support for virtualization using VMware ESXi 4.0 VMware ESXi 4.0 Update 1 Installable. This release offers IP-only support with Red Hat Enterprise Linux Release 5.0 with Update 4.

For additional information about operating system requirements refer to the “Basic Software Requirements” section in *Dialogic® Host Media Processing Software Release 4.1LIN Release Guide*.

This feature specifically focuses on the VMware ESXi 4.0 installable product which provides a native (or full) virtualization layer running on physical servers for abstracting processor, memory, storage, and resources into multiple virtual machines. For more information about virtualization, refer to the VMware web site at www.vmware.com.

- Notes:**
1. It is assumed that the reader is familiar with common terms used to describe basic virtualization concepts, such as guest operating system, host, hypervisor, etc.
 2. Virtualization is not supported on thin-blade configurations.

1.75.1 VMware ESXi 4.0 Virtualization Support

Dialogic® HMP virtualization refers to the capability of running a separate instance of the Dialogic® HMP software release on the “guest” operating system of one or more virtual machines being hosted on the same physical platform (i.e., server). Each Dialogic® HMP software release has a separate runtime license, a number of dedicated resources, and requires a dedicated application (written to standard Dialogic® HMP Global Call and R4 Media API) to manage the resources.

HMP virtualization is implemented using VMware ESXi 4.0 Update 1 Installable. VMware ESXi partitions a physical server into multiple secure and portable virtual machines that can run side by side. Each virtual machine represents a complete system—with processors, memory, networking, storage and BIOS—so that an operating system and software applications can be installed and run in the virtual machine without any modification.

Refer to the VMware ESXi 4.0 documentation at <http://www.vmware.com/support/pubs> for more information.

The density achieved when operating in an virtual environment is directly dependent on the configuration settings of the virtual machine (i.e., CPU, memory, etc.) and the host platform hardware. Users should view the configuration settings provided as guidelines and not absolute, based on the target platform hardware characteristics in which feature validation was performed. Customizing the settings for optimal performance based on needs of the controlling application and host platform should be done by knowledgeable and experience personnel familiar with VMware ESXi products.

1.75.2 Configuring HMP Virtualization

To configure Dialogic® HMP software to run as close as it would in a physical server configuration, the hypervisor should be configured to distribute the host hardware CPU processor, memory, storage, and networking resources to enable the real-time processing of RTP, media, and call control on all instances of the Dialogic® HMP software. The following subsections examine the critical parameters to achieve this goal. Please refer to the vSphere Resource Management Guide found at http://www.vmware.com/pdf/vsphere4/r40_u1/vsp_40_u1_resource_mgmt.pdf for a thorough explanation of the terms and concepts utilized herein.

CPU Affinity Settings

To run real-time software on VMware ESXi, use CPU affinity. This is the recommended method for real-time voice since each virtual processor can get CPU resources directly from one or more of the available host CPUs, reducing the likelihood that virtual processors are rescheduled to give CPU time to another virtual machine.

Each virtual machine is more isolated, which helps real-time software run as though it were in a physical server environment. Due to HMP software's intensive use of the operating system kernel resources, it is also highly recommended to set aside one physical (host) CPU to the VMware ESXi 4.0 hypervisor. This host CPU should not be part of the affinity setting of any of the virtual machines.

For example, on a dual-processor, four-core host system without hyper-threading system, there will be eight physical CPUs available to VMware ESXi. In this scenario, two virtual machines are configured with two virtual processors each. The system administrator could set the first virtual machine CPU affinity to physical CPUs 0 through 3 (total 4), and the second virtual machine CPU with affinity to physical CPUs 4 through 6 (total 3); this leaves physical CPU 7 unassigned and available to the VMware ESXi hypervisor.

Virtual machine configuration is accomplished using the vSphere vCenter or via the VMware CLI. Refer to the vSphere Basic System Administration or equivalent guide at http://www.vmware.com/pdf/vsphere4/r40_u1/vsp_40_u1_admin_guide.pdf for vSphere vCenter information. For VMware CLI instructions, refer to http://www.vmware.com/pdf/vsphere4/r40_u1/vsp_40_u1_vcli.pdf.

- Notes:**
1. Be careful not to cross physical processor boundaries when assigning CPU affinity to virtual machines, so that all host CPUs assigned to a virtual machine belong to the same host physical processor.
 2. On NUMA host servers, it is recommended to keep all physical CPUs affine to a virtual machine residing in the same NUMA node in order to avoid a performance penalty when accessing non-local memory.

Timing Configuration

For optimal virtual machine timing and HMP operation in a virtualized environment, it is recommended that VMware Tools are installed in each virtual machine.

- Install VMware Tools in each virtual machine. Refer to the VMware ESXi Setup Guide for the installation procedure.
- Use the vSphere vCenter utility (or VMware CLI) to access the host system Time Configuration. Provide the address of an appropriate NTP Server in the Date and Time Options, and restart the NTP service to apply the changes.

Note: VMware Tools includes an optional clock synchronization feature “Time Synchronization between the virtual machine and the ESX Server” that can be enabled in the virtual machines, and could conflict with the native synchronization software. Be aware that having both enabled could affect the virtual machine's operating system's ability to correct long-term wall-clock drift, hence affect HMP audio quality.

For HMP media quality, it is recommended that the guest operating system operates with the ntpd process disabled to prevent NTP timing synchronization conflicts between the guest operating system and the ESX server. Use the setup command from the console to disable the ntpd feature prior to starting HMP software.

Resource Budgeting

The same HMP requirements for system resources are required when operating in a VMware ESXi environment. Refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Release Guide* for those requirements.

The user is responsible for distributing the host system so enough resources are available to the virtual machines at all times. In addition to the CPU affinity and timing settings discussed, VMware ESXi and vSphere provide a vast number of virtual machine configuration parameters that affect the configuration and behavior of virtual resources, such as reservation, shares, and resource pools that are outside of the scope of this document but are very important in providing a virtual environment to HMP as close as possible to a physical server environment.

Network Configuration

By default, VMware ESXi provides one virtual switch that handles all virtual machine network traffic according to each virtual machine's IP and MAC addresses and VMware ESXi management network traffic. Virtual machines can be assigned to virtual networks, and these to virtual switches in various network topologies, utilizing all available host physical network interfaces. The system integrator should carefully consider the virtual network layout based on the aggregated network traffic of all virtual machines and the capabilities and number of the physical network interfaces.

1.75.3 Density Limits

Aggregate density limits were tested at the currently supported limits as physical platforms. It is important to note that density projections are platform specific and are susceptible to the performance capabilities of the underlying hardware platform (host), and to the number of virtual machines. Initial density results show that the aggregate density of virtual machines running on the same host may be slightly less than the total capacity of the physical server. This is the result of additional overhead associated with each virtual machine.

1.76 Increased Channel Support for Conferencing

Service Update 53 increases support for up to 1000 channels of conferencing using Dialogic® HMP software.

1.77 H.264 Transcoding Support for 3G-324M Devices

Service Update 53 adds H.264 transcoding support for 3G-324M devices. Service Update 42 added H.264 Transcoding support for IP media and multimedia devices. Now, IP media, multimedia and 3G-324M devices are capable of transcoding inbound H.264 encoded data streams to another codec. For more information, refer to [Section 1.80, “3G-324M H.264 Native Support”](#), on page 177 and [Section 1.82, “H.264 Transcoding Support”](#), on page 184. For documentation updates due to these features, refer to [Section 3.4.1, “Dialogic® 3G-324M API Programming Guide and Library Reference”](#), on page 255.

1.78 Support for HD Voice Conferencing (Wideband Audio Conferencing)

Service Update 49 adds HD voice conferencing (also called wideband audio conferencing) support for G.722 at (64 kbps) and G.722.2 wideband audio codecs.

This feature supports a maximum of 500 parties in a conference using G.722, or a maximum of 150 parties in a conference using G.722.2, or a combination of wideband and narrowband parties.

This feature is an extension of [Section 1.90, “G.722.2 Adaptive Multi-Rate Wideband Codec \(AMR-WB\) Support”](#), on page 194 and [Section 1.93, “G.722 Wideband Codec Support”](#), on page 208. Wideband codecs are required when using HD voice conferencing.

For more information about this feature, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.79 Support for Additive Licensing

With Service Update 46, the application is able to increase the number of features or feature counts by simply adding a new license file to its existing license directory. This feature makes it easier to upgrade or scale up a system with additional license features or counts.

This feature supports the capability to scale licensed resources by augmenting multiple licenses. The licenses must meet the following criteria:

- all licenses must have an additive capable version (version 110 or above)
- all licenses must be located in the same single directory
- all license files in the designated directory must be of the same type. That is, all licenses must be either OEM, DNI or HOST.

If any of the above conditions are not met, then licensing validation will fail.

Note: When asking for a new additive license file, use the Product Center calculator to determine if the sum of all features is within the capacity of the system.

1.79.1 API Changes for Feature Implementation

License file activation is made with the OAM CLI interface. The HMP software was modified to accommodate additive file licenses. Instead of requesting that a license be upgraded to include additional resources for configuration features, the user can request a new license file with additional resources for configuration features. The result is two or more license files that can be combined together to form the desired amount of features.

Currently, the CLI interface has the command to set the license file directory (i.e., conf license directory). To activate the license, the user would call the command “conf license activate <license file name>”. With this feature, HMP now accepts the license file name “ALL” to access all license files found in the directory.

1.79.2 Activating a License File Directory

This section provides license activation procedures for more than one license file found under a directory. Dialogic® Services must be stopped before activating a license.

1. At the CLI prompt, enter the following command to stop services:

```
CLI> conf system ipmedia stop
updated- This action can take a few minutes to complete. Please wait
CLI>
EVENT--SYSTEM Run State [Shutdown]
CLI>
EVENT--SYSTEM Run State [Inactive]
CLI>
EVENT--SYSTEM [not ready]
CLI>
IPMedia stop complete
CLI>
```

2. At the CLI prompt, enter the following command to choose the license directory.

```
CLI> conf license directory /download/dialogic/license/
updated
```

3. At the CLI prompt, enter the following command to show the license files.

```
CLI> show license
Selected License: /usr/dialogic/data/1r1v0e0c0s0fli_ver.lic
License Type: Verification
License Category: Node locked specific
License Seed: 3000000000000
Expiration Date: permanent
License Options: RTP_G_711 1, Voice 1, IP_Call_Control 1,
OEM License
Function:
Library:
License Directory: /download/dialogic/license/
File Name: 30r30v30c30i_pur.lic
```

```

License Type: Purchased
License Category: Node locked specific
Expiration Date: permanent
License Options: RTP_G_711 30, Voice 30, Conferencing 30, IP_Call_Control 30,
File Name: 20r20v20c20i_pur.lic
License Type: Purchased
License Category: Node locked specific
Expiration Date: permanent
License Options: RTP_G_711 20, Voice 20, Conferencing 20, IP_Call_Control 20,
File Name: ALL
License Type: Purchased
License Category: Node locked specific
Expiration Date: permanent
License Options: RTP_G_711 50, Voice 50, Conferencing 50, IP_Call_Control 50, (estimated)

```

4. At the CLI prompt, enter the following command to activate all license files found in the license directory. For example:

```

CLI> conf license activate ALL
CLI>

```

The following output is displayed:

```

CLI> show license
Selected License: /download/dialogic/license/ALL
License Type: Purchased
License Category: Node locked specific
License Seed: 3000423D34400
Expiration Date: permanent
License Options: RTP_G_711 50, Voice 50, Conferencing 50, IP_Call_Control 50,

OEM License
Function:
Library:
License Directory: /download/dialogic/license/
File Name: 30r30v30c30i_pur.lic
License Type: Purchased
License Category: Node locked specific
Expiration Date: permanent
License Options: RTP_G_711 30, Voice 30, Conferencing 30, IP_Call_Control 30,
File Name: 20r20v20c20i_pur.lic
License Type: Purchased
License Category: Node locked specific
Expiration Date: permanent
License Options: RTP_G_711 20, Voice 20, Conferencing 20, IP_Call_Control 20,
File Name: ALL
License Type: Purchased
License Category: Node locked specific
Expiration Date: permanent
License Options: RTP_G_711 50, Voice 50, Conferencing 50, IP_Call_Control 50, (estimated)

```

Configuring a license file from the CLI involves the license file validation. It may take a few seconds to complete. Upon successful completion, a message similar to the following is displayed:

```

CLI>License verification is in progress. Please wait
updated - will be effective on the next ipmedia start"

```

In case of an invalid license, the following error message is displayed:

```

"License activation failed. Reverted to last selected license"

```

To view the current selected license file name, enter the command:


```
CLI> show license
```

5. Enter the following command to start media services:

```
CLI> conf system ipmedia start
updated- This action can take a few minutes to complete. Please wait
CLI>
EVENT--SYSTEM Run State [Initializing]
CLI>
EVENT--SYSTEM Run State [Active]
CLI>
EVENT--SYSTEM [ready]
CLI>
IPMedia start complete
CLI>
```

6. Exit the CLI using the logout command or open another terminal and use your new license.

Prior to this feature, the license file name generated the config (PCD, FCD) file names. With additive licensing, the config file names will be created by counting all of the features from additive licensing. For example, the generated single license file base PCD file was named *30r30v30c30i_pur.pcd*, which matched the license file name. With additive licensing, the generated PCD file name (in /usr/dialogic/data/) in this case would be *50r50v50c50i_gen.pcd*.

1.80 3G-324M H.264 Native Support

Service Update 46 adds support for the H.264 Native codec to the 3G-324M API. For information about H.264 native play/record and IP hairpinning, refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Release Guide*.

1.80.1 3G-324M API Changes

The section describes the changes made to 3G-324M API to support the H.264 Native codec. New data structures and events are described below. Refer to the [Section 3.4.1, “Dialogic® 3G-324M API Programming Guide and Library Reference”](#), on page 255 for updates to existing documentation due to this feature.

- m3g demo config file

New Data Structures

M3G_H264_OPTIONS

```
typedef struct
{
    unsigned char    profile;                /* H264 Profile */
    unsigned char    level;                  /* H264 Level   */
    unsigned char    decoderConfigLength;    /* H264 DCI length */
    unsigned char    decoderConfigInfo[OCTET_STRING_SIZE]; /* used in local and remote H.245
                                                                * OLC May be present
                                                                * exactly once for Logical
                                                                * Channel Signalling */
}
```

```

unsigned char    h264SignalingMask;                /* used to signal parameters
                                                    * described below in local and
                                                    * remote H.245 TCS - optional */

unsigned char    profileIOP;                       /* used in local and remote
                                                    * H.245 TCS - optional */

} M3G_H264_OPTIONS;

/* Bits in h264SignalingMask are assigned as follows:*/
#define M3G_H264_ACCEPT_REDUNDANT_SLICES 0x01|
#define M3G_H264_NAL_ALIGNED_MODE 0x02

```

Description

Specifies capabilities specific to the H.264 algorithm. This structure is a member of the M3G_VIDEO_OPTIONS union.

Field Descriptions

The fields of the M3G_H264_OPTIONS structure are described as follows:

profile

Mandatory H.264 capability as specified in ITU-T Rec. H.241. Latest version of 3GPP TS 26.111 mandates H.264 Baseline profile for 3G-324M. Default value returned in **m3g_GetLocalCaps()** is 64 (Baseline Profile)."

level

Mandatory H.264 capability as specified in ITU-T Rec. H.241. Latest version of 3GPP TS 26.111 mandates H.264 Level 1 (value 15) for 3G-324M. Default value returned in **m3g_GetLocalCaps()** is 15 (Level 1).

decoderConfigLength

Length of decoderConfigurationInformation octet string. This element is only used in encoding or decoding of H.245 OpenLogicalChannel requests. Note that 3GPP TS 26.111 does not mandate that a DCI shall be specified within a video OpenLogicalChannel request. Default value returned in **m3g_GetLocalCaps()** is 0.

decoderConfigInfo

Optionally used in OpenLogicalChannel requests to specify the configuration of the decoder or a particular object (bitstream). Format is specified in 3GPP TS 26.111. Default value returned in **m3g_GetLocalCaps()** is '\0'. When MONA MPC is used to establish a Video H.264 channel, the following DCI octet string is reported: "00 00 00 01 27 42 a0 0a 95 a0 b1 3a 01 e1 10 8d 40 00 00 00 01 28 ce 06 6a"

h264SignalingMask

- Mask 0x01 indicates the capability to use H.264 redundant slices. Format is specified in 3GPP TS 26.111. Default value returned in **m3g_GetLocalCaps()** is false.
- Mask 0x02 indicates that every AL-SDU carrying H.264 shall contain an integer number of NAL units and that the start of the AL-SDU shall be aligned with the start of a NAL. Format is specified in 3GPP TS 26.111. Default value returned in **m3g_GetLocalCaps()** is false.

profileIOP

Indicates that the capability to decode H.264 streams is limited to a common subset of the algorithmic features included in the indicated profile and level. Format is specified in 3GPP TS 26.111. Default value returned in **m3g_GetLocalCaps()** is 0.

M3G_IFRAME_DATA

```
typedef struct
{
    M3G_E_DIRECTION    direction;           /* Direction the Iframe was received from */
    unsigned int        frameSize           /* size of the Iframe */
    M3G_BOOL            dciChanged;         /* Indicates whether the DCI changed */
    unsigned char        decoderConfigLength; /* H.264 DCI length */
    unsigned char        decoderConfigInfo[OCTET_STRING_SIZE]; /* inband H264 DCI*/
} M3G_IFRAME_DATA;
```

Description

Specifies in-band information from either the remote 3G-324M endpoint, or from the device connected to the Mux3G device (i.e., IPM, MM, etc.). This data structure is received via the M3GEV_IFRAME_RCVD event. It only applies to H.264 Video codec.

Field Descriptions

The fields of the M3G_IFRAME_DATA data structure are described as follows:

direction

Direction from which the Iframe was received, Values:

M3G_E_TX: from the device connected to the Mux3G device

M3G_E_RX: from the remote 3G-324M endpoint

iframeSize

Size of the Iframe in bytes.

dciChanged

indicates if the in-band DCI has changed according to the application expectation. It means that the in-band DCI either changed initially versus the out-of-band DCI, or changed mid-stream.

decoderConfigLength

H.264 DCI length. If no DCI is attached to the reported Iframe, the length is 0.

decoderConfigInfo

Optional H.264 DCI Data, if decoderConfigLength is different from 0.

New Events

M3GEV_IFRAME_RCVD

Unsolicited event. Associated with video device type. Indicates that an Iframe was received in-band from either the remote 3G-324M endpoint, or from the device connected to the Mux3G device (i.e., IPM, MM, etc.). It only applies to H.264 Video codec. This event is only received if Iframe Report Eventing is enabled via **m3g_EnableEvents()**. This event can be masked. (Default is disabled.) Data type: M3G_IFRAME_DATA

M3G Demo config File

This new config file shows H.264 support:

```
#####
# Board Parameters:
# Parameters M3G_E_PRM_XXXXXXXXXXXX
#####
PRM: PRM_IFRAME_NOTIFY_CONTROL_MASK      15 #M3G_E_PRM_IFRAME_NOTIFY_CONTROL_MASK
```

```
#####

#####
# Event Notification:
# ENEV:=Enable DSEV:=Disable
#####
# ENEV: IFRAME_EVT
# ENEV: IFRAME_DCI_CHANGE_EVT
#####

#####
# 3G-324M Terminal Capabilities:
# Endpoints Capabilities (amr g723 h264 h263 mpeg4)
#####
# MCAP: 1 amr h264
##### #

#####
# H264 or MPEG4 Decoder Configuration Information (Tx to 3G peer):
# Mode: 1 - in-band DCI
#       2 - out-of-band DCI via H.245 OLC
#       3 - both inband and out-of-band
#
# Endpoints Mode DCI
#####
# DCI: 1 3 000000012742400C95A8B13B011000003E900007530840000000128CE3C80

#####

#####
# SIP Coders

# SIPVID: h263 mpeg4 h264
# SIPAUD: amr g711u g711a g723 g729
#
# This features is only supported on releases that include video transcoding
# (ie, MMP2.0, MMK...)
# Note: Used for sip outbound calls, sip incbound calls defined by SIP sdp
#       To enable transcoding: Audio Xcode and Video Xcode are set in EP line
#####
# SIPVID: h264
# SIPAUD: g711u
#####

#####
# Multimedia Endpoints
#
# Op - Operation (PLAY/RECORD) - what it does
# Digit - DTMF <0-9> which controls the Play/Record operation
#       - digit 0 must exist for default operation
# Typ - MM File Type (dmf, 3gp)
#       dmf = Dialogic MultiMedia Format(ie, hmp .vid/.aud 'proprietary')
#       3gp = *.3gp file format (!system restrictions apply!)
# Aud - Audio codec in file (amr, pcm, g723...)
# Vid - Video codec in file (h263, mpeg4, h264)
# VRes - Video Resolution in file (qcif, cif, sqcif)
# VFR - Video Frame Rate in fps (6, 7(=7.5fps), 10, 15, 25, 30)
# VBR - Video Bit Rate in kbps (ex. 40=40kbps, 256=256kbps)
# File Names - separate audio/video with unique names for play and record

# Digit Op Typ Aud Vid VRes VFR VBR Aud_Play_File Vid_Play_File Aud_Rec_File
# Vid_Rec_File
#####
# MMINFO: 0 PLAY dmf amr h264 qcif 10 40 dml.aud dml.vid
# rec.aud rec.vid
#####
```

1.81 Unspecified G.723.1 Bit Rate in Outgoing SIP Requests with SDP

With Service Update 46, a Global Call application in 1PCC mode can choose not to specify the G.723.1 codec bit rate, namely 5.3 kbps or 6.3 kbps, in an outgoing SIP message with SDP body. Instead, the application can let the far end UA request the bit rate. Feature enablement and disablement can be controlled either at the IPT board-level device or the IPT network device (channel).

1.81.1 Feature Overview

G.723.1 is a dual-rate codec supporting 5.3 kbps and 6.3 kbps bit rates. With SIP, the choice of one or the other rate for RTP streaming is specified in an SDP body in outgoing SIP messages using an optional bit rate parameter in the format transport (a=fmtp) attribute of the media announcement. Currently, when in 1PCC mode, Global Call always includes this optional bit rate parameter in SDP bodies irrespective of whether the application's IP_CAPABILITY data structure setting specifies a single G.723.1 bit rate (5.3 or 6.3 kbps), both G.723.1 bit rates (5.3 and 6.3 kbps), or “don't care” for local transmit rates.

When both G.723.1 rates are specified, only the one in the first IP_CAPABILITY structure is included. The following example shows an SDP message extract from an HMP SIP request where the application set both rates in the IP_CAPABILITY data structure. The outgoing SDP contains only the bit rate that was specified first. In this particular case, the capability field in the first IP_CAPABILITY element with IP_CAP_DIR_LCLTRANSMIT direction was set to GCCAP_AUDIO_g7231_5_3k for the IPT network device:

```
v=0
o=Intel_IPCCLib 50083120 50083121 IN IP4 192.168.185.64
s=Dialogic_SIP_CCLLIB
i=session information
c=IN IP4 192.168.185.64
t=0 0
m=audio 49152 RTP/AVP 0 8 4 101
a=rtpmap:4 G723/8000
a=fmtp:4 bitrate=5.3; annexa=no
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

With this feature, when the Global C all 1PCC application specifies both G.723.1 bit rates as transmit codec selection, the HMP software no longer sends the optional bit rate parameter in the SDP body of a SIP request or response. Instead, the remote UA has the responsibility to select one of the G.723.1 bit rates for bit rate negotiation.

Note: If only one of the G.723.1 bit rates is specified by the application transmit codec selection, the HMP software maintains its existing behavior and the specified selection applies when negotiating the bit rate to use irrespective of this feature enablement. Likewise, there is no change in behavior if the application selects “don't care” (GCCAP_dontcare) as transmit codec selection, meaning that the complete list of coders supported by a product is used when negotiating the coder type to be used, also irrespective of this feature enablement.

Note: This feature is only applicable in 1PCC Global Call mode.

1.81.2 Implementation

The Global Call functions, **gc_SetConfigData()** and **gc_SetUserInfo()**, are used to enable or disable the feature at the IPT board-level or IPT network device (channel) level. To control the behavior of the G.723.1 bit rate for the entire IPT board, use the **gc_SetConfigData()** with a Target Type GCTGT_CCLIB_NETIF and the board-level device handle for Target ID. Use **gc_SetUserInfo()** with the IPT network device (channel) to control the behavior of the G.723.1 bit rate for all calls on a given channel. In both cases the following applies:

The set ID parameter IPSET_CONFIG is used to configure general IP parameters. For this feature, this set ID used a new parameter ID. The pre-existing generic parameter values, IP_DISABLE and IP_ENABLE, are used to disable or enable this feature.

The following table shows the new parameter ID in the IPSET_CONFIG parameter set.

Parameter ID	Data Type & Size	Description	SIP/H.323
IPPARM_SEND_G723_UNSPECIFIED_BITRATE	Type: int Size: size of (int)	Specifies not send any bit rate for G.723.1 on outgoing SIP requests or responses. Valid values: IP_ENABLE 1 (Default) IP_DISABLE 0	SIP only (1PCC)

Note: The new parameter ID is only applicable if both 5.3 kbps and 6.3 kbps G.723.1 bit rates are specified in the capability field in IP_CAPABILITY elements with IP_CAP_DIR_LCLTRANSMIT direction. Otherwise, the IPPARM_SEND_G723_UNSPECIFIED_BITRATE parameter has no effect.

When controlling the new feature behavior at the board-level basis, the **gc_SetConfigData()** function should be used with the following arguments:

target_type

The target object type. Use GCTGT_CCLIB_NETIF.

cclib_target_id

Indicates the board-level device.

GC_PARM_BLK target_datap

Pointer to the data from target object. Use set_ID = IPSET_CONFIG and the new parm_ID = IPPARM_SEND_G723_UNSPECIFIED_BITRATE.

This feature is enabled by default, which eliminates the bit rate parameter in SDP bodies in SIP messages from HMP when both bit rates are specified in IP_CAPABILITY structures with IP_CAP_DIR_LCLTRANSMIT as direction. The application has the option to disable this feature using the IP_DISABLE value for

IPPARM_SEND_G723_UNSPECIFIED_BITRATE if desired to maintain backward compatibility.

Note: It is recommended that whenever the application desires to specify one G.723.1 bit rate as part of the SIP request, that only one G.723.1 bit rate is passed to the HMP software in a capability field of an IP_CAPABILITY element with IP_CAP_DIR_LCLTRANSMIT as direction. In that particular case, the new parameter ID IPPARM_SEND_G723_UNSPECIFIED_BITRATE is not applicable.

1.81.3 Examples

Since this feature is enabled by default, this example is intended to help customers revert to the previous behavior by disabling the feature if so desired. The example is divided in two parts, the first part demonstrates how to disable this feature on a board-level basis, i.e., make the HMP software behave prior to this feature implementation.

```
char *boardDeviceName = "N_iptB0:P_IP"; // example of board name for gc_openex()
LINEDEV boarddevh = 0; // IP board-level line device filled by gc_openex()

INT32 processEvtHandler()
{
    GC_PARM_BLK *parmbldp = NULL;
    long request_id = 0;

    switch (evtType)
    {
        ...
        case GCEV_OPENEX:
            // board case

            gc_util_insert_parm_val(&parmbldp, IPSET_CONFIG,
                                   IPPARM_SEND_G723_UNSPECIFIED_BITRATE, IP_DISABLE);

            if (gc_SetConfigData(GCTGT_CCLIB_NETIF, boarddevh, parmbldp, 0,
                                GCUUPDATE_IMMEDIATE, & request_id, EV_ASYNC) != GC_SUCCESS)
            {
                // Process error
            }
            gc_util_delete_parm_blk(parmbldp);
            break;
        ...
    }
```

This second part completes the example by showing how to include both G.723.1 audio bit rates in any outgoing SIP messages with SDP body on a given channel.

```
if (ipcap.capability != GCCAP_DATA_t38UDPFax) {
    ipcap.type = GCCAPTYPE_AUDIO;
    ipcap.direction = IP_CAP_DIR_LCLTRANSMIT;
    ipcap.extra.audio.frames_per_pkt = GCCAP_AUDIO_g7231_6_3k;
    ipcap.extra.audio.VAD = vad;
    /* append the GC_PARM_BLK with the respective TX codec */
    gc_util_insert_parm_ref(&parmbldp, GCSET_CHAN_CAPABILITY,
                           IPPARM_LOCAL_CAPABILITY, sizeof(IP_CAPABILITY), &ipcap);
    ipcap.type = GCCAPTYPE_AUDIO;
    ipcap.direction = IP_CAP_DIR_LCLTRANSMIT;
    ipcap.extra.audio.frames_per_pkt = GCCAP_AUDIO_g7231_5_3k;
    ipcap.extra.audio.VAD = vad;
    /* append the GC_PARM_BLK with the respective TX codec */
    gc_util_insert_parm_ref(&parmbldp, GCSET_CHAN_CAPABILITY,
                           IPPARM_LOCAL_CAPABILITY, sizeof(IP_CAPABILITY), &ipcap);
}
```

```

        if (gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, parmbldp,
                           scope) != GC_SUCCESS)
        {
            //Process error
        }
    }
}

```

1.82 H.264 Transcoding Support

Service Update 42 adds H.264 Transcoding support. The H.264 transcoding feature allows an IP media and multimedia device to transcode H.264 encoded data such that inbound H.264 streams can be transcoded to another codec (e.g., MPEG4 or H.263), and/or resized to a different size (e.g., CIF to SQCIF).

To take advantage of the H264 transcoding, new parameters have been added to the IPM_VIDEO_CODER_INFO_EX and the MM_VIDEO_CODEC data structures. For details about the changes, refer to [Section 3.4.14, “Dialogic® IP Media Library API Programming Guide and Library Reference”](#), on page 258 for IP media and [Section 3.4.16, “Dialogic® Multimedia API Programming Guide and Library Reference”](#), on page 258: for multimedia.

1.83 Support for Dialogic® DNI2410AMCTEHMP AdvancedMC Module

Service Update 42 provides support for the Dialogic® DNI2410AMCTEHMP AMC module. This is a high-density, high-performance, network interface with eight T1/E1 digital network interfaces in an AdvancedTCA AMC mid-size module. The AMC module provides support for on-board digital TDM E1 Clear Channel signaling and host-based voice, speech, conference, fax, and IP transcoding.

Note: When installing the Dialogic® DNI2410AMCTEHMP AMC module, be sure to refer to the Installation Guide (*Dialogic® Quick Install Card*) that is provided with each product for important information about how to install the module in a ATCA carrier board and information about interconnection to the network.

Software Installation Considerations

When installing the Dialogic® HMP software with the Dialogic® DNI2410AMCTEHMP AMC module in the system, there are six packages to include due to operating system dependency with the Open IPMI libraries. Three of those six are installed by default, and the other three need to be selected for install.

Installed by default:

OpenIPMI-2.0.6-11el5.i386.rpm

OpenIPMI-libs-2.0.6-11el5.i386.rpm

OpenIPMI-tools-2.0.6-11el5.i386.rpm

Not installed by default, but required for successful operation:

OpenIPMI-perl-2.0.6-11el5.i386.rpm
OpenIPMI-python-2.0.6-11el5.i386.rpm
OpenIPMI-devel-2.0.6-11el5.i386.rpm

1.84 32-bit Compatibility Mode on 64-bit Linux Systems

Service Update 42 adds support for 32-bit compatibility mode on 64-bit Linux systems for Dialogic® HMP Interface Boards. Previously, this support was for VoIP only.

This new (32/64 bit) feature applies to all Linux versions officially supported by the Dialogic® PowerMedia™ HMP for Linux Release 4.1 so there are no separate restrictions. Refer to the “Basic Software Requirements” section in *Dialogic® Host Media Processing Software Release 4.1 LIN Release Guide* for a list of supported Linux versions.

1.85 3G-324 Gateway + Media Server for IVVR

With Service Update 42, a direct .3gp file with hinted H.263 + AMR-NB file can be streamed natively to IP where the IP video H.263 packetization is either RFC 2190 (H.263) or RFC 2429 (H.263-1998).

Note: The file containing H.263-1998 using RFC 2429 (RFC 4629) packetization can only contain baseline H.263 (no options). This is the same restriction that applies for receiving H.263-1998 using RFC 2429 (RFC 4629) packetization from an RTSP server.

1.86 Overlap-Receive Support for Limited SIP-I Interworking Scenarios

Service Update 42 provides SIP-I interworking capability by providing a method for handling overlap-receive SIP calls, where called-party addressing is supplied in multiple INVITEs but needs to be propagated to the application as standard en bloc signaling calls.

This specific SIP-I interworking scenario is for an overlap signaling call originated at a remote ISUP exchange containing partial DNIS addressing in the called-party number field of a IAM and SAM(s) which are propagated to HMP as embedded MIME bodies in multiple SIP INVITE requests as part of a SIP transaction. In this scenario, each INVITE within the transaction contains an ISUP IAM and zero or more SAM messages embedded in a MIME body. IAM/SAM messages would carry partial called-party addressing until a final INVITE contains the complete called-party addressing within the ISUP IAM and optional SAM messages.

With this feature, HMP in 1PCC mode handles a SIP-I overlap-receive scenario by parsing the ISUP IAM and optional SAM MIME bodies in each INVITE and determining if the called-party address is incomplete. If it is incomplete, HMP automatically rejects it with

a 484 Address Incomplete response. The processing continues until the called-party address is complete, at which point the application is presented with the incoming call as if it were a standard en bloc signaling INVITE.

1.86.1 Feature Overview

This feature is concerned with ITU-T Q.763 IAM and SAM messages, in particular with the called-party number field in them, containing the DNIS. When this feature is implemented, HMP, acting as a SIP UAS, can support specific SIP-I overlap-receive scenarios where an interworking unit (gateway) at the far end embeds the IAM/SAM message in a multipart MIME body as part of the INVITE requests, in addition to the SDP MIME part. Each INVITE carries embedded ISUP IAM and optional SAM MIME bodies with partial called-party addressing (DNIS). In these scenarios, the first INVITE normally carries an ISUP IAM from the originating exchange, and subsequent INVITEs carry ISUP SAMs belonging to the same call transaction. The ISUP IAM contains an initial called-party addressing and a number of ISUP SAMs will contain additional called-party addressing until the complete called-party address signaling is provided.

Multiple INVITEs containing ISUP IAM and zero or more SAM messages, each one containing the original addressing plus additional addressing, are received by HMP until the called-party addressing is complete, at which point an end of signaling (stop digit) appears at the end of the called-party number field, indicating a complete address. This digit uniquely identifies the last INVITE in the sequence.

When this feature is enabled, HMP parses the MIME body in each incoming INVITE request to determine if ISUP IAM or SAM messages in ITU-T Q.763 format are present. If so, the called-party number field is parsed looking for an end of signaling (stop digit “F”) presence. If the IAM or SAM message does not contain the stop digit, then the INVITE is automatically rejected by HMP with a 484 Address Incomplete response with no application interaction. No further processing by HMP is done on this INVITE, nor is the application made aware of the automatic 484 Address Incomplete response. If the stop digit (“F”) is present, then the INVITE is presented to the application as a standard GCEV_OFFERED event with the complete en bloc DNIS address, which will be contained in the INVITE header fields.

1.86.2 Enabling MIME-based Overlap-Receive

The application enables MIME-based Overlap-Receive on a board basis using **gc_SetConfigData()** function. The IPSET_CONFIG set ID is used in conjunction with a new parameter ID, IPPARM_MIME_OVERLAP_RECEIVE, defined as follows:

Parameter ID	Data Type	Description	SIP/ H.323
IPPARM_MIME_OVERLAP_RECEIVE	Type: int Size: size of (int)	Enables the overlap receive feature of SIP-I based on embedded MIME ISUP messages. Possible values are: <ul style="list-style-type: none">GCPV_ENABLE – ISUP IAM and optional SAM messages embedded in a MIME body in the incoming INVITE message are parsed to check if the “F” stop digit is present. If not present, the call will be rejected with a 484 Address Incomplete response.GCPV_DISABLE – ISUP MIME bodies will not be parsed. This is the default behavior.	SIP only

Once the feature is enabled at the board-level basis, all IPT network devices (channels) are able to parse MIME bodies carrying ISUP IAM and SAM messages in ITU-T Q.763 format containing overlap called-party number fields and automatically reject those INVITEs within the transaction that do not contain complete addressing. Once the complete address is received, HMP presents the incoming call and its complete called-party addressing as a standard GCEV_OFFERED event to a channel. The channel can handle the incoming event as it would normally do with any other incoming call. The called-party addressing signaling is expected to be available in the relevant header fields within the INVITE which will be made available to the channel as a standard DNIS number.

- Notes:**
1. HMP requires that ISUP messages embedded as MIME bodies in the incoming INVITEs are of IAM or SAM type and adhere to the ITU-T Q.763 specification for the feature to be able to parse overlap-receive signaling.
 2. ISUP SAM messages are not always required to carry additional called-party signaling, and a single IAM message may be sufficient to carry complete en bloc called-party addressing.
 3. It is possible for one INVITE to carry an ISUP IAM and one or more SAMs, each containing partial called-party number fields.

Example

The following code shows how to enable this feature:

```
#include <stdio.h>
#include <srllib.h>
#include <gcplib.h>
#include <gcerr.h>
#include <gcip.h>
#include <gcip_defs.h>

/*
```

```

* Assume the following has been done:
* 1. Open board device iptB1 and save handle as boardDev which will be
* used in gc_SetConfigData()
*/

int enable_overlap()
{
    GC_PARM_BLK_PARM blkp = NULL;
    long request_id = 0;
    gc_util_insert_parm_val(&blkp,
        IPSET_CONFIG,
        IPPARM_MIME_OVERLAP_RECEIVE,
        sizeof(int),
        GCPV_ENABLE);

    if(gc_SetConfigData(GCTGT_CCLIB_NETIF, boardDev, blkp, 0, /*timeout*/,
        GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC) != GC_SUCCESS)
    {
        // handle error...
    }

    return (0);
}

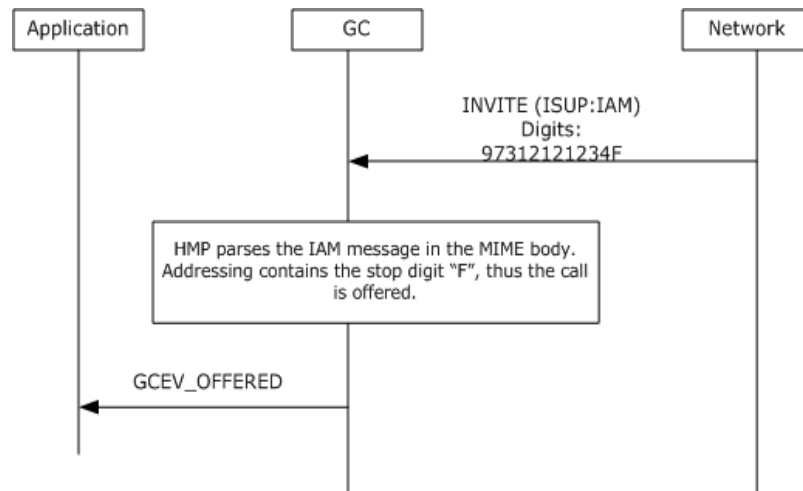
```

1.86.3 Use Cases

The overlap-receive feature involves one or more INVITE messages sent by the remote UAC. Each INVITE may contain initial and additional called-party addressing in embedded ISUP IAM and optional SAM MIME bodies within the request. The ISUP IAM and optional SAM bodies are parsed seeking for the called-party number field. Each INVITE is treated independent of any previous INVITE in the transaction until the last INVITE with complete addressing is received. The following figures demonstrate typical use cases for this feature.

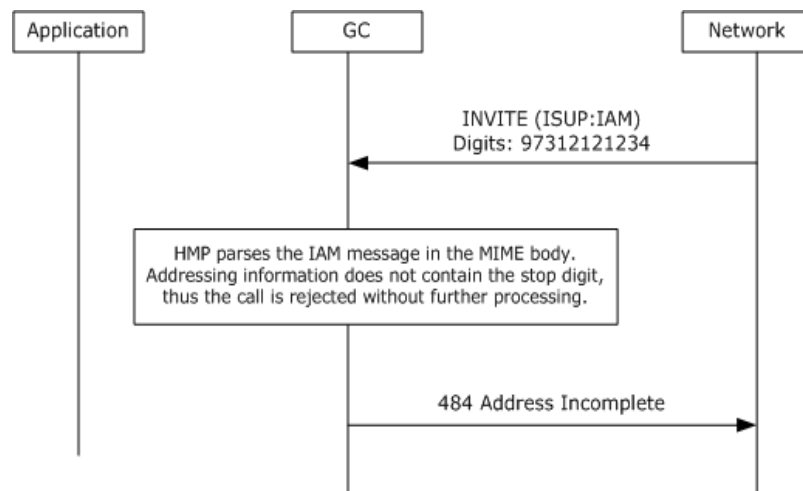
INVITE with Complete Address in IAM

The incoming INVITE contains an IAM message with the complete address (contains the stop digit "F").



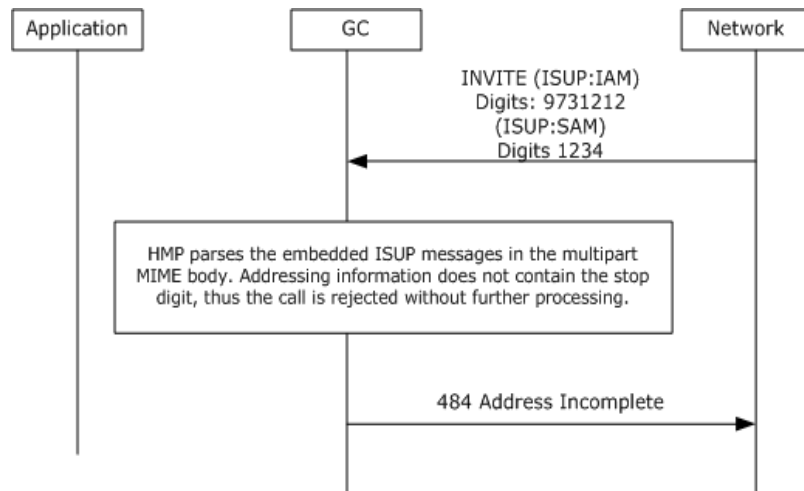
INVITE with Incomplete Address in IAM

The incoming INVITE contains an IAM message with incomplete address (does not contain the stop digit).



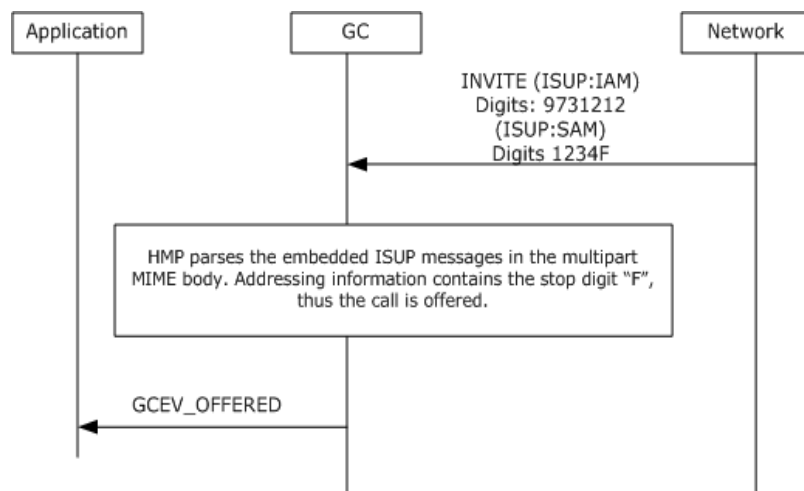
INVITE with Incomplete Address in IAM and SAM

The incoming INVITE contains an IAM and a SAM, and neither message contains the complete address (neither one contains the stop digit).



INVITE with Complete Address in IAM and SAM

The incoming INVITE contains both IAM and SAM, and the messages contain the complete address (contains the stop digit “F”).



For additional information about MIME bodies, refer to the “SIP MIME Overview” in *Dialogic® Global Call IP Technology Guide*.

1.87 Processing Multiple 18x Provisional Responses

Service Update 42 introduces a method for obtaining subsequent provisional 18x SIP responses using the GCEV_EXTENSION event. When this feature is enabled, the first incoming 18x response will generate a GCEV_ALERTING as expected; however, all subsequent 18x responses will be sent to application by the GCEV_EXTENSION event. The block parameter associated with the event will contain SIP response headers in the same block format as in the GCEV_ALERTING event for the first provisional response received on the channel for the same transaction.

The following 18x provisional responses are supported:

- 180 Ringing
- 181 Call is Being Forwarded
- 182 Queued
- 183 Session Progress

1.87.1 Feature Implementation

A new bitmask value, EXTENSIONEVT_SIP_18x_RESPONSE, is added to the existing IPSET_EXTENSIONEVT_MSK parameter set for feature enablement on a board-level basis using the **gc_SetConfigData()** function with the GCTGT_CCLIB_NETIF target_type and the board-level device handle as the target ID. Once enabled, all IPT network devices (channels) will have the ability to be notified of multiple 18x provisional responses from the UAS.

Since the GCEV_EXTENSION has several variances for IP, for example, it can be used to retrieve call-related information, get notification of underlying protocol connection or disconnection, or get notification of media streaming initiation and termination, a new extension ID, IPEXTID_RECEIVED_18x_RESPONSE, is introduced so a GCEV_EXTENSION event for this feature can be differentiated from another feature.

Once the GCEV_EXTENSION event is received, the application can easily find the reason by looking at the ext_id which is part of the EXTENSIONEVTBLK embedded in the event data pointer.

Furthermore, if the ext_id matches IPEXTID_RECEIVED_18x_RESPONSE, the 18x provisional response headers can be found in the same format as in the GCEV_ALERTING event for the first provisional response received on the channel in the same transaction. That is, the parmbldp associated with the GCEV_EXTENSION event contains the following mandatory set_ID and parm_ID:

- set_ID = IPSET_SIP_RESPONSE_CODE
- parm_ID = IPPARM_RECEIVED_RESPONSE_STATUS_CODE
- value = 3-digit integer representing the Status-Code from the response's Status-Line

If Reason-Phrase retrieval from 182 and 183 Provisional Responses has been enabled, then the parmbkp associated with the GCEV_EXTENSION event contains set_ID and parm_ID:

- set_ID = IPSET_MSG_INFO
- parm_ID = IPPARM_SIP_HDR
- value = NULL-terminated string which begins with the string "Reason-Phrase"

Processing Responses

The following behavior occurs within the application when the feature is enabled:

The first incoming 18x response will generate a GCEV_ALERTING as expected, or if a 183 was received and the application enabled support for Ingress 183 Session Progress Informational Response Containing SDP, a GCEV_PROGRESSING is received instead.

The GCEV_EXTENSION event indicates the receipt of any subsequent 18x responses within a transaction. The METAEVENT structure associated with it contains:

- EXTENSIONEVTBLK.ext_id = IPEXTID_RECEIVED_18x_RESPONSE
- EXTENSIONEVTBLK.parmblk will have at least one (mandatory) entry with the following (reused from GCEV_ALERTING):
set_ID = IPSET_SIP_RESPONSE_CODE
parm_ID = IPPARM_RECEIVED_RESPONSE_STATUS_CODE
value = 3-digit integer representing the Status-Code from Status-Line of the received provisional response

If Reason-Phrase retrieval from 182 and 183 Provisional Responses is enabled, the EXTENSIONEVTBLK.parmblk will contain the following additional entry:

- set_ID = IPSET_MSG_INFO
- parm_ID = IPPARM_SIP_HDR
- value = NULL-terminated string which begins with the string "Reason-Phrase" and contains the equivalent header from the response's Status-Line

Note: This feature is activated on the board level; however, the GCEV_EXTENSION event is IPT network device (channel) specific.

1.87.2 Enabling/Disabling GCEV_EXTENSION

Enabling and disabling unsolicited GCEV_EXTENSION notification events is done by manipulating the event mask using the **gc_SetConfigData()** function as described in the "Enabling and Disabling Unsolicited Notification Events" section in *Dialogic® Global Call IP Technology Guide*.

The following example shows how to set the EXTENSIONEVT_SIP_18X_RESPONSE mask:


```

char *boardDeviceName = "N_iptB0:P_IP"; // example of board name for gc_openex()
LINEDEV boardDevice = 0; // IP channel board-level line device filled by gc_openex()

static void evt_hdlr()
{
...
    case GCEV_OPENEX:
        ...
        // board case
        {

            GC_PARM_BLK * pparm = NULL;
            long req_id;

            gc_util_insert_parm_val(pparm,
                IPSET_EXTENSION_EVT_MSK,
                GCACT_ADDMSK,
                GC_VALUE_LONG,
                EXTENSION_EVT_SIP_18X_RESPONSE);

            gc_setConfigData(GCTGT_CCLIB_NETIF,
                boardDevice,
                pparm,
                0,
                GCUPDATE_IMMEDIATE,
                &req_id,
                EV_ASYNC);

        }
...
}

```

1.87.3 Retrieving 18x Code from GCEV_EXTENSION

The following example demonstrates how to retrieve the Status-Code and Reason-Phrase headers (if available) from an 18x response mapped to a GCEV_EXTENSION unsolicited event once the feature is enabled. Refer to the “Retrieving Reason-Phrase from 182 and 183 Provisional Responses” section in *Dialogic® Global Call IP Technology Guide* for information about setting up the application to receive Reason-Phrase header from these SIP responses.

```

case GCEV_EXTENSION:
GC_PARM_BLK pParmBlock;
EXTENSION_EVTBLK *pextensionBlk;
GC_PARM_DATAP l_pParmData;
pextensionBlk = (EXTENSION_EVTBLK *) (m_pMetaEvent->extevtdatap);
pParmBlock = (&(pextensionBlk->parmblock));

if (pextensionBlk->ext_id == IPEXTID_RECEIVED_18X_RESPONSE)
{
    while ((l_pParm = gc_util_next_parm(pParmBlock, l_pParm)) != 0)
    {
        int l_mtype = (int) (* (l_pParm->value_buf));
        switch (l_pParm->set_ID)
        {
            case IPSET_SIP_RESPONSE_CODE:
                if (l_pParm->parm_ID == IPPARM_RECEIVED_RESPONSE_STATUS_CODE)
                {
                    if (l_pParmData->value_size != 0)
                    {
                        int code_18x = *(int *) l_pParm->value_buf;
                        //...
                    }
                }
            }
        }
    }
}

```

```

        }
        break;
    case IPSET_SIP_MSGINFO:
        if (l_pParm->parm_ID == IPPARM_SIP_HDR)
        {
            if (l_pParmData->value_size != 0)
            {
                char siphdr_18x[IP_SIP_HDR_MAXLEN];

                strncpy(siphdr_18x, (char*)parmp->value_buf, parmp->value_size);

                siphdr_18x[parmp->value_size]='\0';
                //...
            }
        }
        break;
    default:
        //... warning no other type allowed
        break;
    }
}
}
else
{
    // other ext_id
}
break;

... other GCEV_ case

```

1.88 TLS and SRTP Channel Support Increase

With Service Update 42, Dialogic® HMP software now supports 500 Transport Layer Security (TLS) channels and 500 Secure Real Time Protocol (SRTP) channels. Previously, the software was limited to 250 channels of each.

1.89 33 Frames Per Packet Support (AMR)

Service Update 37 increases the maximum frames per packet from 10 to 33 for AMR.

Note: Voice activity detection (VAD) is disabled by firmware when frames per packet is greater than 10. This cannot be changed by the application.

Refer to [Section 3.2.2, “Dialogic® Host Media Processing Software for Linux Configuration Guide”](#), on page 254 for documentation updates due to this feature.

1.90 G.722.2 Adaptive Multi-Rate Wideband Codec (AMR-WB) Support

Service Update 37 adds support for the G.722.2 Adaptive Multi-Rate Wideband (AMR-WB) Speech Coder. G.722.2 uses a wideband audio band of 50 - 7000 Hz instead of 200 - 3400 Hz for wideband applications. The increased bandwidth improves the intelligibility

and naturalness of speech significantly. For more information about wideband codec support, refer to [Section 1.93, “G.722 Wideband Codec Support”](#), on page 208.

1.91 Registering Authentication Data without Realm String

Service Update 37 introduces a method for registering authentication data without using realm string.

Currently, the realm element of the authentication quadruplet must contain a non-empty string. This feature supports realm and identity as an empty string, so that identity authentication is verified using the same identity or username as the registrar server.

If the realm value is empty, the application uses the identity element to match with the identity provided by the server.

```
[ "", identity, username, password ]
```

If realm and identity are empty, the application can use the username. The username must be the same username in the string identity provided by the registrar server.

```
[ "", "", username, password ]
```

For example, if the registrar server provides [bob@10.10.20.25] as identity, then "bob" must be configured as the username for the registrar server and the authentication quadruplet.

For more information about SIP Digest Authentication, refer to the *Dialogic® Global Call IP Technology Guide*.

1.92 Handling non-2xx Responses to T.38 Switch

Service Update 37 introduces 1PCC Global Call support for RFC 3261 compliance for non-2xx responses to re-INVITE requests to switch to or from audio to T.38 fax and back.

1.92.1 Feature Description

Currently, when a Global Call SIP application initiates a media type switch from/to audio or to/from fax within a dialog with a re-INVITE request, the existing media session and dialog are terminated if the request is rejected by the UAS with any non-2xx response. RFC 3261 clearly requires that the UAC keep the exiting session in a dialog alive as though the re-INVITE never occurred.

With this feature, the existing media session within the dialog remains active upon a switching request from one media type to another (fax to audio or audio to fax).

This feature is enabled by default when the application calls the **gc_ReqModifyCall()** function or the **gc_Extension()** function with the codec switch value. On failure to switch, the application will receive the failure events, GCEV_REQ_MODIFY_REJ/GCEV_REQ_MODIFY_FAIL and the GCEV_EXTENSION event with parm ID set to IPPARM_REJECT for set ID IPSET_SWITCH_CODEC respectively. The existing media session will be reestablished underneath, and the requested local media information will contain the stored (existing) media information.

Because this feature is limited to “Manual” operating mode, an application must be configured in “Manual” mode to control the association and disassociation of media and T.38 fax devices during each call. The mode of operation is set on a board device basis. The operating mode for set ID/parm ID pair IPSET_CONFIG/IPPARM_OPERATING_MODE must be set to either of the following:

- IP_T38_MANUAL_MODE
- IP_T38_MANUAL_MODIFY_MODE

For additional information, refer to the documentation updates for Chapter 3. IP Call Scenarios and Chapter 4. IP Specific Operations in the *Dialogic® Global Call IP Technology Guide*.

1.92.2 Manual Mode Example

This example demonstrates “Manual” mode when the switch from T.38 fax to audio is unsuccessful.

```
INT32 switchFromFaxToAudio( )
{
    GC_PARM_BLK *parmbldp = NULL;

    IP_CONNECT ipConnect;
    ipConnect.version = 0x100;
    ipConnect.mediaHandle = pline->mediaH;

    gc_util_insert_parm_ref(&parmbldp, IPSET_FOIP, IPPARM_T38_DISCONNECT,
        (sizeof(IP_CONNECT)), (void *)&ipConnect);

    gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbldp, GC_SINGLECALL);

    gc_util_delete_parm_blk(parmbldp);
    /* Initiate audio codec switch */
    gc_util_insert_parm_ref(&parmbldp, IPSET_SWITCH_CODEC, IPPARM_AUDIO_INITIATE, sizeof(int),
        NULL);

    gc_Extension(GCTGT_GCLIB_CRN, pline->crn, IPEXTID_CHANGEMODE, parmbldp, NULL, EV_ASYNC);
    gc_util_delete_parm_blk(parmbldp);
}

INT32 processEvtHandler()
{
    METAEVENT metaEvent;
    GC_PARM_BLK *parmbldp = NULL;
    GC_INFO t_info;

    switch (evtType)
    {
        {
            case GCEV_EXTENSIONCMPLT:
                /* received extension complete event for audio initiation*/
            }
        }
    }
}
```

```

/* do nothing */
break;

case GCEV_EXTENSION:
/* received extension event for media readiness */
ext_evtblkp = (EXTENSIONEVTBLK *) metaEvent.extevtdatap;
parmbldp = &ext_evtblkp->parmbldp;
while (t_gcParmDatap = gc_util_next_parm(parmbldp, t_gcParmDatap))
{
switch(t_gcParmDatap->set_ID)
{
case IPSET_SWITCH_CODEC:
switch(t_gcParmDatap->parm_ID)
{
case IPPARM_REJECT:
gc_ResultInfo(&metaEvent,&t_info);
gc_util_insert_parm_ref(&parmbldp, IPSET_FOIP, IPPARM_T38_CONNECT,
(size_t)(sizeof(IP_CONNECT)), (void *)(&ipConnect));

gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbldp, GC_SINGLECALL);
break;
case IPPARM_READY:
/* Ready to send and receive audio */
gc_Listen();
break;
}
}
}
}

```

This example demonstrates “Manual” mode when the switch from audio to fax is unsuccessful.

```

INT32 processEvtHandler( )
{
METAEVENT metaEvent;
GC_PARM_BLK *parmbldp = NULL;
IP_CONNECT ipConnect;
GC_INFO t_info;

switch (evtType)
{
case GCEV_CONNECTED:
/* received Connect event */
/* in conversation */
ipConnect.version = 0x100;
ipConnect.mediaHandle = pline->mediaH;
ipConnect.faxHandle = pline->faxH;
ipConnect.connectType = IP_FULLDUP;
gc_util_insert_parm_ref(&parmbldp, IPSET_FOIP, IPPARM_T38_CONNECT,
(size_t)(sizeof(IP_CONNECT)), (void *)(&ipConnect));

gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbldp, GC_SINGLECALL);

gc_util_delete_parm_blk(parmbldp);
/* Initiate T.38 codec switch */
gc_util_insert_parm_ref(&parmbldp, IPSET_SWITCH_CODEC, IPPARM_T38_INITIATE,
sizeof(int), NULL);

gc_Extension(GCTGT_GCLIB_CRN, pline->crn, IPEXTID_CHANGEMODE, parmbldp, NULL, EV_ASYNC);
gc_util_delete_parm_blk(parmbldp);
break;

case GCEV_EXTENSIONCMPLT:
/* received extension complete event for T.38 initiation*/
/* do nothing */
break;
}
}

```

```

case GCEV_EXTENSION:
/* received extension event for media readiness */
ext_evtblkp = (EXTENSIONEVTBLK *) metaEvent.extevtdatap;
parmbldp = &ext_evtblkp->parmbldp;
while (t_gcParmDatap = gc_util_next_parm(parmbldp, t_gcParmDatap))
{
switch(t_gcParmDatap->set_ID)
{
case IPSET_SWITCH_CODEC:
switch(t_gcParmDatap->parm_ID);
{
case IPPARM_REJECT:

gc_ResultInfo(&metaEvent,&t_info);

gc_util_insert_parm_ref(&parmbldp, IPSET_FOIP,
                        IPPARM_T38_DISCONNECT, (sizeof(IP_CONNECT)), (void
                        *)(&ipConnect));

gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbldp, GC_SINGLECALL);

gc_Listen();
/* IPT to IPM*/
break;
case IPPARM_READY:
/* Ready to send and receive fax */
fx_sendfax();
break;
}
break;
}
}

```

1.92.3 Manual Modify Mode Examples

This example demonstrates “Manual” modify mode when the switch from T.38 fax to audio is unsuccessful.

```

INT32 switchFromFaxToAudio()
{
GC_PARM_BLK *parmbldp = NULL;
IP_CONNECT ipConnect;
ipConnect.version = 0x100;
ipConnect.mediaHandle = pline->mediaH;

gc_util_insert_parm_ref(&parmbldp, IPSET_FOIP, IPPARM_T38_DISCONNECT,
                        (sizeof(IP_CONNECT)), (void *)(&ipConnect));

gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbldp,GC_SINGLECALL);

gc_util_delete_parm_blk(parmbldp);
/* Initiate audio codec switch */
if( gc_util_insert_parm_ref(&parmbldp,GCSET_CHAN_CAPABILITY, IPPARM_LOCAL_CAPABILITY,
                        sizeof(IP_CAPABILITY), &ipcap) != GC_SUCCESS )
{
//error
}
gc_ReqModifyCall (GCTGT_GCLIB_CRN,pline->crn, parmbldp, EV_ASYNC);
gc_util_delete_parm_blk(parmbldp);
}
INT32 processEvtHandler()
{
METAEVENT metaEvent;
GC_PARM_BLK *parmbldp = NULL;

```

```

switch (evtType)
{
    case GCEV_EXTENSIONCMPLT:
        /* received extension complete event for audio initiation*/
        /* do nothing */
        break;

    case GCEV_MODIFY_CALL_ACK:
        // switch complete
        gc_Listen();
        break;

    case GCEV_MODIFY_CALL_REJ:
    case GCEV_MODIFY_CALL_FAIL:
        gc_util_insert_parm_ref(&parmbkp, IPSET_FOIP, IPPARM_T38_CONNECT,
                               (sizeof(IP_CONNECT)), (void *)(&ipConnect));
        break;
}

```

This example demonstrates “Manual” modify mode when the switch from audio to fax is unsuccessful.

```

INT32 processEvtHandler()
{
    METAEVENT metaEvent;
    GC_PARM_BLK *parmbkp = NULL;
    IP_CONNECT ipConnect;
    switch (evtType)
    {
    case GCEV_CONNECTED:
        /* received Connect event */
        /* in conversation */
        ipConnect.version = 0x100;
        ipConnect.mediaHandle = pline->mediaH;
        ipConnect.faxHandle = pline->faxH;
        ipConnect.connectType = IP_FULLDUP;
        gc_util_insert_parm_ref(&parmbkp, IPSET_FOIP, IPPARM_T38_CONNECT,
                               (sizeof(IP_CONNECT)), (void *)(&ipConnect));
        gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbkp, GC_SINGLECALL);

        if( gc_util_insert_parm_ref(&parmbkp, GCSET_CHAN_CAPABILITY,
                                   IPPARM_LOCAL_CAPABILITY, sizeof(IP_CAPABILITY), &ipcap) != GC_SUCCESS )
        {
            //error
        }
        gc_ReqModifyCall (GCTGT_GCLIB_CRN, pline->crn, parmbkp, EV_ASYNC);
        gc_util_delete_parm_blk(parmbkp);
        break;

    case GCEV_MODIFY_CALL_ACK:
        // Switch Complete
        fx_sendfax();
        break;

    case GCEV_MODIFY_CALL_REJ:
    case GCEV_MODIFY_CALL_FAIL:

        /* received extension event for media readiness */
        gc_util_insert_parm_ref(&parmbkp, IPSET_FOIP,
                               IPPARM_T38_DISCONNECT, (sizeof(IP_CONNECT)), (void *)(&ipConnect));

        gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->crn, parmbkp, GC_SINGLECALL);

        gc_Listen();
        /* IPT to IPM*/
        break;
    }
}

```

1.92.4 MIME Insertion in Outgoing ACK

Service Update 37 introduces a method for embedding a MIME body in an outgoing SIP ACK request message for 4xx/5xx/6xx response messages that terminate certain transactions.

1.92.5 Feature Description

Some interworking SIP configurations use embedded MIME bodies in messages to pass call state information to a non-SIP destination network. Although Dialogic[®] HMP software supports the encapsulation of MIME bodies in many request messages out of HMP, this is not the case for ACK requests.

In particular, when an outbound call out of HMP (UAC) cannot be completed by the remote UAS, it will send a rejection/ message using the appropriate 4xx, 5xx, or 6xx SIP final response message.

Currently, once a 4xx/5xx/6xx response is received, the stack automatically generates the ACK request, terminates the transaction, and informs the application of the failure with a GCEV_DISCONNECTED event. Since the ACK is automatically generated without application intervention, MIME body encapsulation is not possible.

This feature allows the application to create a MIME body ahead of time so that it is then available for the IP cclib to add it to outgoing ACK requests that are a result of certain outbound call rejections.

Once the feature is enabled at the IPT network device (channel), and the application builds the desired MIME body ahead of time, the MIME body will be added ACK messages out of HMP that are generated as a response to any 4xx/5xx/6xx SIP final response messages in most transaction scenarios.

An exception is the 487 “Request Terminated” message, generated by the server (UAS) as a final response to a CANCEL to a previous INVITE out of HMP. In this case, the ACK message for this 487 will not contain any MIME body irrespective of feature enablement. The **gc_DropCall()** that generated the ACK will internally disable MIME body encapsulation to the subsequent ACK to this 487.

Note: A 487 message generated by the server (UAS) resulting from a transaction timer expiration, as per an “Expires” header in the outgoing INVITE out of HMP, will be handled like most 4xx/5xx/6xx final response scenarios. The ACK message sent in this case will contain a MIME body pre-built by the application as part of this feature.

Disclaimers

- The feature is intended for use with proxy or general UAS that are able to handle encapsulated bodies in ACK messages. In particular, IETF RFC 3261 SIP stipulates that the placement of bodies in ACK for non-2xx is NOT RECOMMENDED since they cannot be rejected if the body is not understood.

- If bodies are to be inserted, IETF RFC 3261 SIP RECOMMENDS that they be the same as they appeared in the original INVITE. This feature does not restrict the building of any type of valid MIME bodies, and the application is responsible for MIME content generation and applicability.
- Although this feature is exemplified in a SIP <-> ISDN User Part (ISUP) interworking, known as SIP-I, it does not imply SIP-I compliance. The feature is very specific to limited outbound scenarios.

1.92.6 Enabling MIME Insertion

To implement this feature, a new set ID is introduced. This set ID is defined as:

```
#define IPSET_MIME_ACK_TO_REJECTION BASE_SETID+42
```

It may be used with the **gc_SetUserInfo()** function, along with target_type GCTGT_GCLIB_CHAN, and duration set to GC_SINGLECALL.

The method of building the MIME body using IPSET_MIME_ACK_TO_REJECTION is the same as the method used for building the two-level GC_PARM_BLK data structures as documented in the *Dialogic® Global Call IP Technology Guide* for IPSET_MIME or IPSET_MIME_200OKTOBYE.

Note: To create MIME bodies requires the manipulation of MIME bodies to be enabled at the IP virtual board level, prior to a calling the **gc_Start()** function, and using the mask sip_msginfo_mask = IP_SIP_MSGINFO_ENABLE | IP_SIP_MIME_ENABLE.

1.92.7 Setting the MIME Body for the ACK Message

After the feature is enabled, and prior to making an outbound call, the application creates the MIME body containing the desired data to be sent along with ACK responses to final 4xx/5xx/6xx responses for those transactions as defined in the Feature Description section.

The following table shows the parameter IDs in the IPSET_MIME_ACK_TO_REJECTION set ID:

Parameter ID	Data Type & Size	Description	SIP/ H.323
IPPARM_MIME_PART	Type: pointer to GC_PARM_BLK Size: 4 bytes	Required parameter. Used to set or get SIP message MIME part(s). Parameter value is a pointer to a GC_PARM_BLK structure that contains a list of pointers to one or more GC_PARM_BLK structures that contain MIME message parts.	SIP only
IPPARM_MIME_PART_BODY	Type: char * Size: 4 bytes	Required parameter. Used to copy MIME part body between application and Global Call space. Parameter value is a pointer to a MIME part body.	SIP only
† For parameters with data of type String, the length in a GC_PARM_BLK is the length of the data string plus 1.			

Parameter ID	Data Type & Size	Description	SIP/ H.323
IPPARM_MIME_PART_BODY_SIZE	Type: Unsigned int Size: 4 bytes	Required parameter. Used to indicate the actual size of the MIME part body, not including MIME part headers.	SIP only
IPPARM_MIME_PART_HEADER	Type: Null-terminated string † Size: max. length = max_parm_data_size (configured at start-up via IPCCLIB_START_DATA)	Optional parameter. Used to contain MIME part header field in format of "field-name: field-value". Field-name can be any string other than "Content-type". Content is not checked by Global Call before insertion into SIP message.	SIP only
IPPARM_MIME_PART_TYPE	Type: Null-terminated string † Size: max. length = max_parm_data_size (configured at start-up via IPCCLIB_START_DATA)	Required parameter. Used to contain name and value of the MIME part content type field. String must begin with the field name "Content-Type:".	SIP only
† For parameters with data of type String, the length in a GC_PARM_BLK is the length of the data string plus 1.			

- Notes:**
1. The application is responsible for building the appropriate MIME bodies. IP cclib does not make any attempt to parse the raw data in the MIME, and encapsulates it in the outgoing ACK as built.
 2. IPSET_MIME_ACK_TO_REJECTION is supported only on a channel basis specifically for the GCTGT_GCLIB_CHAN target type. The MIME message set is not valid after the call is cleared. For the next outbound call, the application must call the **gc_SetUserInfo()** function to reset the MIME body.

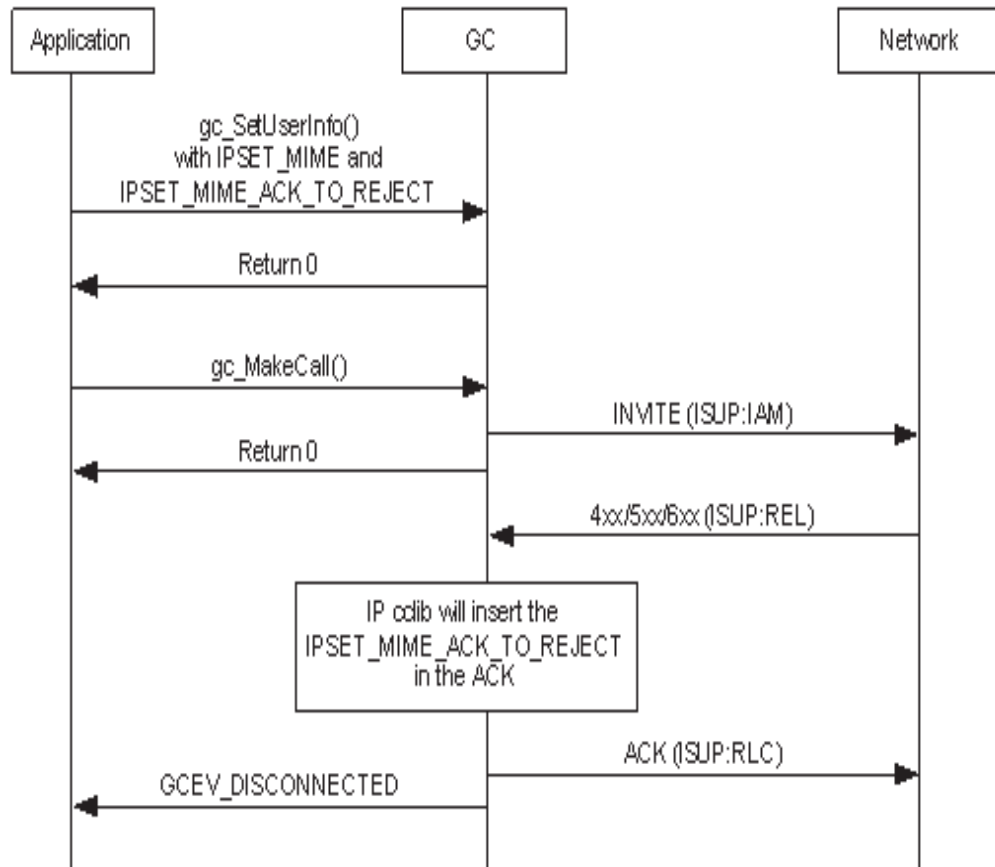
For more information about creating MIME messages, refer to the *Dialogic® Global Call IP Technology Guide*.

1.92.8 Possible Use Case and Code Example

The following code examples illustrate how to enable this feature and send embedded MIME bodies in outgoing ACK response messages. The examples assume that the MIME body manipulation mask was enabled using the INIT_IP_VIRTBOARD data structure.

The following diagram shows a very specific example of encapsulating MIME bodies in two different request messages out of HMP, one of them is specific to this feature. This use case scenario represents a possible SIP <-> ISUP interworking scenario.

- Notes:**
1. This is for a specific use-case scenario. It is not intended to imply that this is the only SIP interworking use case where this feature may be used.
 2. The following diagram and code shows how to send a possible ISUP:IAM body as part of an outgoing INVITE out of HMP. This functionality is already available in HMP as part of the IPSET_MIME set ID.
 3. An example ISUP:RLC body is sent automatically (via this feature) as part of an outgoing ACK out of HMP from this 4xx/5xx/6xx final rejection response from UAS.



Two different MIME bodies are to be created, one for the INVITE request and the other one for the ACK request (this feature) out of HMP. The examples show the following two possible ways of building the two MIME bodies:

- Single call to **gc_SetUserInfo()**
- Two separate **gc_SetUserInfo()** calls

Note: The MIME bodies in this example are purely for the sake of a complete example, and should not be relied on to contain valid ISUP messages.

Single call to **gc_SetUserInfo()**

```

void setMIMEInfo(int index)
{
    char str[MAX_STRING_SIZE];
    int frc;

    /* The following variable is used for the MAIN GC_PARM_BLK that
     * will contain the different MIME blocks */
    GC_PARM_BLK *pParmBlockMain = NULL;

    /* The following variables are used for the IAM MIME */
    GC_PARM_BLK *pParmBlockIAM_B = NULL;
  
```

```

char *pBodyTypeIAM = "Content-Type: application/ISUP; version = itu-t92+";
char *pBodyIAM = "01 00 49 00 00 03 02 00 07 04 10 00 33 63 21\r\n43 00 00 03 06 0d 03 80 90
a2 07 03 10 03 63\r\n53 00 10 0a 07 03 10 27 80 88 03 00 00 89 8b\r\n0e 95 1e 1e 1e 06 26 05 0d
f5 01 06 10 04 00";
char *pPartHeaderIAM1 = "Content-Disposition: signal ;handling=optional";
char *pPartHeaderIAM2 = "Content-User: Dialogic ;type=demo";

/* The following variables are used for the RLC MIME */
GC_PARM_BLK *pParmBlockRLC_B = NULL;

char *pBodyTypeRLC = "Content-Type: application/ISUP; version = itu-t92+";
char *pBodyRLC = "01 00 49 00 00 03 02 00 07 04 10 00 33 63 21\r\n43 00 00 03 06 0d 03 80 90
a2 07 03 10 03 63";
char *pPartHeaderRLC1 = "Content-Disposition: signal ;handling=optional";
char *pPartHeaderRLC2 = "Content-User: Dialogic ;type=demo";

/* Set the IAM MIME */
/* Add 1 to strlen for the NULL termination character */
gc_util_insert_parm_ref_ex(&pParmBlockIAM_B,
                           IPSET_MIME,
                           IPPARM_MIME_PART_TYPE,
                           (unsigned long)(strlen(pBodyTypeIAM) + 1),
                           pBodyTypeIAM);

/* Insert Body Size */
gc_util_insert_parm_val(&pParmBlockIAM_B,
                       IPSET_MIME,
                       IPPARM_MIME_PART_BODY_SIZE,
                       sizeof(unsigned long),
                       strlen(pBodyIAM));

/* Insert MIME part Body Pointer */
gc_util_insert_parm_val(&pParmBlockIAM_B,
                       IPSET_MIME,
                       IPPARM_MIME_PART_BODY,
                       sizeof(unsigned long),
                       (unsigned long)pBodyIAM);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockIAM_B,
                           IPSET_MIME,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderIAM1) + 1),
                           pPartHeaderIAM1);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockIAM_B,
                           IPSET_MIME,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderIAM2) + 1),
                           pPartHeaderIAM2);

/* Insert parm block B pointer to Main parm block */
gc_util_insert_parm_val(&pParmBlockMain,
                       IPSET_MIME,
                       IPPARM_MIME_PART,
                       sizeof(unsigned long),
                       (unsigned long)pParmBlockIAM_B);

/* Set the RLC MIME */
/* Add 1 to strlen for the NULL termination character */
gc_util_insert_parm_ref_ex(&pParmBlockRLC_B,
                           IPSET_MIME_ACK_TO_REJECTION,
                           IPPARM_MIME_PART_TYPE,
                           (unsigned long)(strlen(pBodyTypeRLC) + 1),
                           pBodyTypeRLC);

```

```

/* Insert Body Size */
gc_util_insert_parm_val(&pParmBlockRLC_B,
                        IPSET_MIME_ACK_TO_REJECTION,
                        IPPARM_MIME_PART_BODY_SIZE,
                        sizeof(unsigned long),
                        strlen(pBodyRLC));

/* Insert MIME part Body Pointer */
gc_util_insert_parm_val(&pParmBlockRLC_B,
                        IPSET_MIME_ACK_TO_REJECTION,
                        IPPARM_MIME_PART_BODY,
                        sizeof(unsigned long),
                        (unsigned long)pBodyRLC);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockRLC_B,
                           IPSET_MIME_ACK_TO_REJECTION,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderRLC1) + 1),
                           pPartHeaderRLC1);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockRLC_B,
                           IPSET_MIME_ACK_TO_REJECTION,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderRLC2) + 1),
                           pPartHeaderRLC2);

/* Insert parm block B pointer to MAIN parm block */
gc_util_insert_parm_val(&pParmBlockMain,
                        IPSET_MIME_ACK_TO_REJECTION,
                        IPPARM_MIME_PART,
                        sizeof(unsigned long),
                        (unsigned long)pParmBlockRLC_B);

frc = gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, pParmBlockMain, GC_SINGLECALL);
if (GC_SUCCESS != frc)
{
    sprintf(str, "gc_SetUserInfo failed");
    printandlog(index, GC_APIERR, NULL, str, 0);
}

gc_util_delete_parm_blk(pParmBlockIAM_B);
gc_util_delete_parm_blk(pParmBlockRLC_B);
gc_util_delete_parm_blk(pParmBlockMain);
}

```

Two separate calls to **gc_SetUserInfo()**

```

void setMIMEInfo(int index)
{
    char str[MAX_STRING_SIZE];
    int frc;

    /* The following variable is used for the MAIN GC_PARM_BLK that
     * will contain the different MIME blocks */
    GC_PARM_BLK *pParmBlockMain = NULL;

    /* The following variables are used for the IAM MIME */
    GC_PARM_BLK *pParmBlockIAM_B = NULL;

    char *pBodyTypeIAM = "Content-Type: application/ISUP; version = itu-t92+";
    char *pBodyIAM = "01 00 49 00 00 03 02 00 07 04 10 00 33 63 21\r\n43 00 00 03 06 0d 03 80 90
a2 07 03 10 03 63\r\n53 00 10 0a 07 03 10 27 80 88 03 00 00 89 8b\r\n0e 95 1e 1e 1e 06 26 05 0d

```

```

f5 01 06 10 04 00";
char *pPartHeaderIAM1 = "Content-Disposition: signal ;handling=optional";
char *pPartHeaderIAM2 = "Content-User: Dialogic ;type=demo";

/* The following variables are used for the RLC MIME */
GC_PARM_BLK *pParmBlockRLC_B = NULL;

|
char *pBodyTypeRLC = "Content-Type: application/ISUP; version = itu-t92+";
char *pBodyRLC = "01 00 49 00 00 03 02 00 07 04 10 00 33 63 21\r\n43 00 00 03 06 0d 03 80 90
a2 07 03 10 03 63";
char *pPartHeaderRLC1 = "Content-Disposition: signal ;handling=optional";
char *pPartHeaderRLC2 = "Content-User: Dialogic ;type=demo";

/* Set the IAM MIME */
/* Add 1 to strlen for the NULL termination character */
gc_util_insert_parm_ref_ex(&pParmBlockIAM_B,
                           IPSET_MIME,
                           IPPARM_MIME_PART_TYPE,
                           (unsigned long)(strlen(pBodyTypeIAM) + 1),
                           pBodyTypeIAM);

/* Insert Body Size */
gc_util_insert_parm_val(&pParmBlockIAM_B,
                       IPSET_MIME,
                       IPPARM_MIME_PART_BODY_SIZE,
                       sizeof(unsigned long),
                       strlen(pBodyIAM));

/* Insert MIME part Body Pointer */
gc_util_insert_parm_val(&pParmBlockIAM_B,
                       IPSET_MIME,
                       IPPARM_MIME_PART_BODY,
                       sizeof(unsigned long),
                       (unsigned long)pBodyIAM);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockIAM_B,
                           IPSET_MIME,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderIAM1) + 1),
                           pPartHeaderIAM1);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockIAM_B,
                           IPSET_MIME,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderIAM2) + 1),
                           pPartHeaderIAM2);

/* Insert parm block B pointer to Main parm block */
gc_util_insert_parm_val(&pParmBlockMain,
                       IPSET_MIME,
                       IPPARM_MIME_PART,
                       sizeof(unsigned long),
                       (unsigned long)pParmBlockIAM_B);

frc = gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, pParmBlockMain, GC_SINGLECALL);
if(GC_SUCCESS != frc)
{
    sprintf(str, "gc_SetUserInfo failed for IAM MIME");
    printandlog(index, GC_APIERR, NULL, str, 0);
}

gc_util_delete_parm_blk(pParmBlockIAM_B);
gc_util_delete_parm_blk(pParmBlockMain);

pParmBlockMain = NULL;

```

```

/* Set the RLC MIME */
/* Add 1 to strlen for the NULL termination character */
gc_util_insert_parm_ref_ex(&pParmBlockRLC_B,
                           IPSET_MIME_ACK_TO_REJECTION,
                           IPPARM_MIME_PART_TYPE,
                           (unsigned long)(strlen(pBodyTypeRLC) + 1),
                           pBodyTypeRLC);

/* Insert Body Size */
gc_util_insert_parm_val(&pParmBlockRLC_B,
                       IPSET_MIME_ACK_TO_REJECTION,
                       IPPARM_MIME_PART_BODY_SIZE,
                       sizeof(unsigned long),
                       strlen(pBodyRLC));

/* Insert MIME part Body Pointer */
gc_util_insert_parm_val(&pParmBlockRLC_B,
                       IPSET_MIME_ACK_TO_REJECTION,
                       IPPARM_MIME_PART_BODY,
                       sizeof(unsigned long),
                       (unsigned long)pBodyRLC);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockRLC_B,
                           IPSET_MIME_ACK_TO_REJECTION,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderRLC1) + 1),
                           pPartHeaderRLC1);

/* Insert other header fields */
gc_util_insert_parm_ref_ex(&pParmBlockRLC_B,
                           IPSET_MIME_ACK_TO_REJECTION,
                           IPPARM_MIME_PART_HEADER,
                           (unsigned long)(strlen(pPartHeaderRLC2) + 1),
                           pPartHeaderRLC2);

/* Insert parm block B pointer to MAIN parm block */
gc_util_insert_parm_val(&pParmBlockMain,
                       IPSET_MIME_ACK_TO_REJECTION,
                       IPPARM_MIME_PART,
                       sizeof(unsigned long),
                       (unsigned long)pParmBlockRLC_B);
frc = gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, pParmBlockMain, GC_SINGLECALL);
if (GC_SUCCESS != frc)

{
    sprintf(str, "gc_SetUserInfo failed for ACK MIME");
    printandlog(index, GC_APIERR, NULL, str, 0);
}

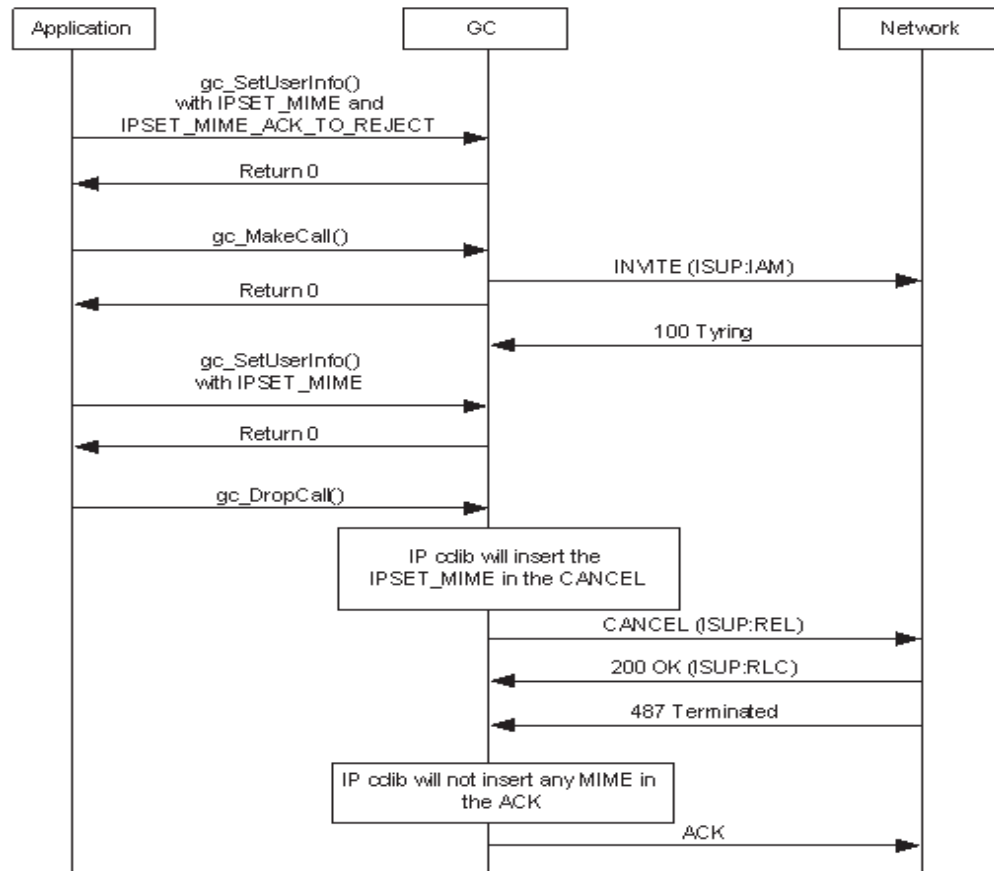
gc_util_delete_parm_blk(pParmBlockRLC_B);
gc_util_delete_parm_blk(pParmBlockMain);
}

```

1.92.9 Exception INVITE/CANCEL/487 Use Case

As described in the Feature Description section, there is a particular use case where the ACK will not contain an embedded MIME body even if the feature was enabled, and the IPSET_MIME_ACK_TO_REJECTION was set ahead of time. Since the call cancellation was already process by the network (ISUP:RLC in 200OK example below), there is no further need to embed a MIME body in the ACK that completes the transaction. The

diagram below shows this specific scenario specifically for a possible SIP <-> ISUP interworking configuration.;



1.93 G.722 Wideband Codec Support

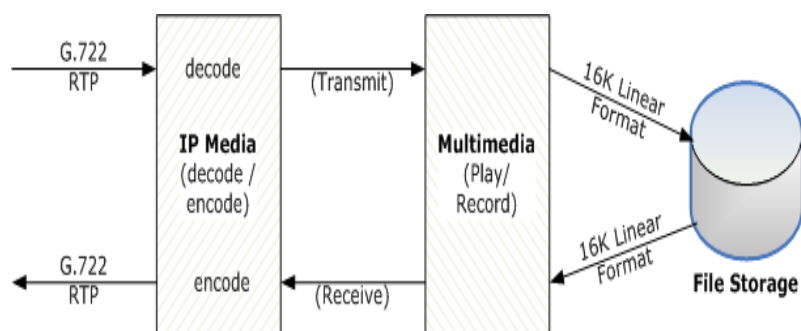
Service Update 32 adds support for the G.722 wideband audio codec [ITU-T G.722]. This audio coder provides 7 kHz wideband audio samples at a 16 kHz sampling rate.

1.93.1 Feature Description

The G.722 audio codec format is used in fixed network voice over IP applications where the required bandwidth is typically not restricted. The G.722 codec offers a significant improvement in speech quality over narrowband codecs such as G.711. G.722 wideband support can reduce or increase the sampling rate of linear data exchanged between internal media device resources, and includes transcoding between narrowband and wideband coding formats.

This feature also provides a common file format for storage of wideband-only or both wideband and narrowband recorded audio media for performing play/record operations. Currently, Dialogic® HMP software supports Linear PCM (128 kbps) 16-bit, 8 kHz; and a proprietary narrowband Dialogic Media Format (DMF), herein known as file format. The file format is used to store compressed audio data for native play/record operations. This feature extends the file format support to 16 bit, 16 kHz compressed audio data (wideband file format) for native G.722 audio play/record operations. Additionally, this feature supports 16 bit, 16 kHz Linear PCM encoded audio data in Wave (.wav) file format (container) for storage of both wideband and narrowband data. The Wave file format will contain uncompressed Linear PCM audio data.

The figure below shows the basic media flow. Note the connections and devices for a basic G.722 play/record operation where network endpoints have negotiated (during SIP session establishment) the use of G.722 encoding/decoding. This figure also highlights the source or target file, which is stored in the new 16 kHz Linear PCM common file format.



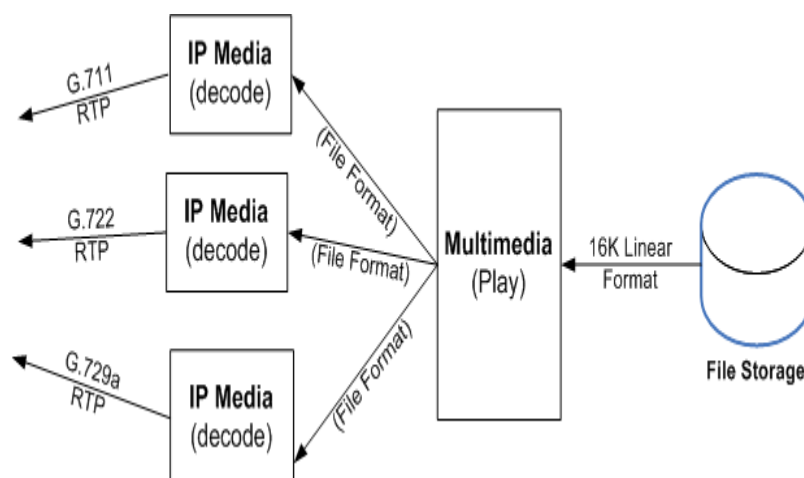
1.93.2 Use Case Scenarios

This section provides common scenarios for this feature.

Multi-party audio content distribution

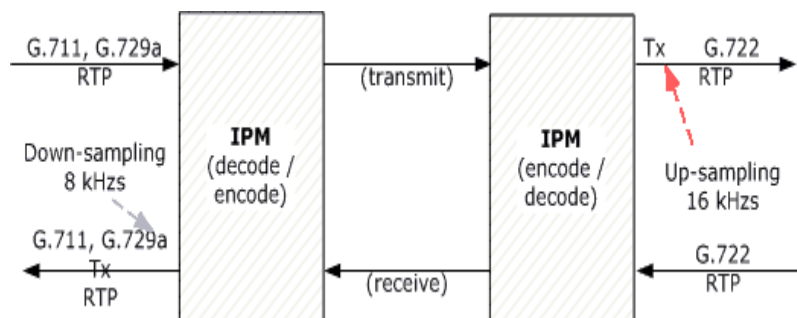
This scenario demonstrates playing stored media content out to multiple IP network endpoints, where the content is transcoded to G.722 wideband in addition to other

supported narrowband audio code formats, such as G.711 or G.729a. The following figure illustrates the connections and the configuration to support this capability.



Transcode between IP network endpoints

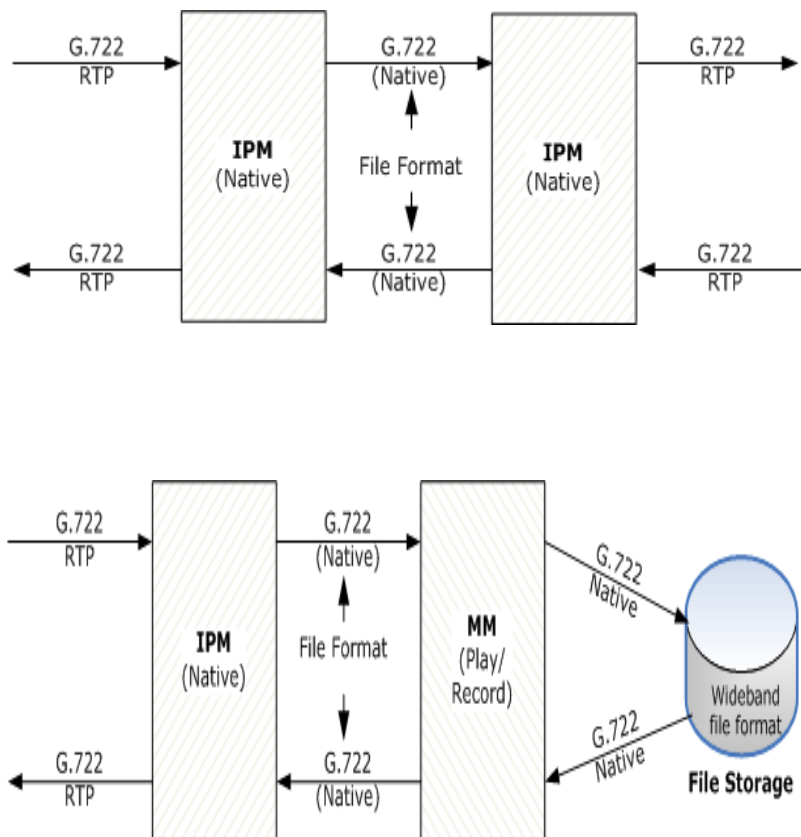
Transcoding between IP network endpoints with up or down sampling at the transmit network sides streaming G.722 wideband reduces or increases the sampling rate of linear data exchanged between internal media device resources. Transcoding between narrowband and wideband coding formats includes both narrowband and wideband endpoints with sampling rate conversion.



Native wideband connection and operation

The following two figures illustrate the native operations and data flow when performing IP PBX switching type functions or play/record G.722 wideband streams. This enables customers to achieve higher densities and reduce latency by eliminating the processing

overhead associated with transcoding (decode/encode). The first figure shows a bidirectional native wideband connection between IP resources (hairpinning). The second figure shows a play/record native wideband operation.



1.93.3 IP Media API Library Update

The following values are added to the IPM_AUDIO_CODER_INFO data structure eCoderType field to support G.722:

- CODER_TYPE_G722_64K
- CODER_TYPE_G722_64K_NATIVE

Note: The Dialogic® Multimedia API supports the play/record of 16 kbps linear PCM and 16 kbps wave. It does not support playing the G.722 Dialogic® proprietary file format when transcoding is enabled. The G.722 Dialogic® proprietary file format is supported for native play/record only.

The following table lists the new audio coder properties:

eCoderType	Frame Size (ms)	Frames per Packet (fpp)	eVadEnable Value
CODER_TYPE_G722_64K	20 or 10	1	Must be CODER_VAD_DISABLE
CODER_TYPE_G722_64K_NATIVE	20 or 10	1	Must be CODER_VAD_DISABLE

For more information, refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

1.93.3.1 Usage Guidelines

The following usage guidelines as well as restrictions and limitations are described for this feature:

- ITU-T G.722 defines three basic modes of operation for the audio codec corresponding to the bit rates used for the 7 kHz audio coding: 64, 56 and 48 kb/s respectively. This feature is restricted to mode 1 operation, where only 64kb/s coding is supported. Modes 2 and 3 (48 and 56 kbps with auxiliary data channel) are not supported.
- Wideband play/record operations are supported using the multimedia device and the supported file formats (G.722 Dialogic® proprietary file format or 16 kHz Linear PCM Wave), wideband for native audio play/record, and wave for decode/encode cases.
- Voice resource devices are used for inband DTMF detection and tone generation. Tone detection and generation during wideband sessions is confined to narrow band frequencies. No special custom tones in the upper frequency range are supported.

1.94 Increase in Channel Density for G.711 Codec

Service Update 32, increases support for up to 1500 channels of play and record for the G.711 codec. For additional channel density information, select Dialogic® PowerMedia™ HMP for Linux Release 4.1 from this location:

http://www.dialogic.com/products/ip_enabled/hmp_software.htm.

1.95 Monitor Mode Support for HMP Conferencing

With Service Update 32, a conferencing application is able to form audio connections from multiple HMP devices listening half duplex to a single conference party and from multiple conference parties listening half duplex to a single device transmitting on the TDM bus.

1.95.1 Feature Description

Prior to this feature, there was no way to create multiple connections from or to one conference party with the conferencing API. This limited the number of HMP devices “listening” to a conference to the number of conference resources in a license.

With this feature, an application can listen to conference output by retrieving a conference party transmit TDM bus time slot. Multiple HMP devices can then listen to conference output in half-duplex mode by using their technology-specific TDM bus listen APIs (for example, `ipm_listen()`). This feature also adds the ability for a conference party to listen in half-duplex mode to any TDM time slot from traditional devices.

For more information about this feature, refer to the *Dialogic® Conferencing API Programming Guide and Library Reference*.

1.96 Increase in TLS and SRTP Channels

With Service Update 32, Dialogic® HMP software supports 250 Transport Layer Security (TLS) channels and 250 Secure Real Time Protocol (SRTP) channels. Previously, the software was limited to 125 channels of each.

1.97 IPM(H.263+) to IPM(H.263) Connection Support

Service Update 32 adds support for streaming H.263 Baseline Profile (Profile 0) Level 10 video natively between two IP Media endpoints, one using RFC 2429 packetization and the other using RFC 2190 packetization. When using this feature, the media server converts between the two H.263 packetization formats without using video transcoding.

Refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for more information about H.263 using RFC 2429 packetization. For documentation changes as a result of this feature, refer to the [Section 3.4.14, “Dialogic® IP Media Library API Programming Guide and Library Reference”](#), on page 258.

- Notes:**
1. The codec using RFC 2429 packetization is also known as H.263-1998 or H.263+.
 2. Baseline Profile is supported.
 3. H.263 annexes are not supported.

1.98 3PCC Support for Dynamic Selection of Outbound SIP Proxy

Service Update 32 adds third-party call control (3PCC) mode for the dynamic selection of outbound SIP proxy feature. Previously, this feature was available in 1PCC mode only. Once enabled in 3PCC, this feature applies to outgoing SIP requests ACK, INFO, INVITE, OPTIONS, REFER, REGISTER, BYE, NOTIFY, SUBSCRIBE, UPDATE and PRACK.

For details about this feature, refer to the [Support for Dynamic Selection of Outbound SIP Proxy](#) section.

1.99 Native H.264 Support

Service Update 25 supports H.264 native play/record and IP hairpinning. For more information about this feature, refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Release Guide*.

1.100 32-bit Compatibility Mode

Service Update 23 adds support for 32-bit compatibility mode on a 64-bit operating system for VoIP only.

1.101 Support for Four Octal-Span Boards

With Service Update 23, a maximum of four octal-span Dialogic® HMP Interface Boards (DNI Boards) can be used in a system to provide 960 TDM-based calls.

1.102 Support for HMP 3.1LIN Features

Service Update 23 now supports the following features of the Dialogic® HMP Software Release 3.1LIN:

- Continuous Speech Processing (CSP)
- H.323 Signaling Support
- Modified Media (SIP re-INVITE and IP Call Control)
- Multimedia user I/O
- Automatic Gain Control and Volume Control

Refer to the *Dialogic® Host Media Processing Software Release 4.1LIN Release Guide*, available on the documentation bookshelf, for detailed information about these features.

1.103 Support for WaitCall Cancellation

With Service Update 23, a SIP application is able to dynamically block the ability of an IPT network device to receive a call. This action prevents possible glare conditions during an attempt to make an outbound call while an incoming call is being processed on the same channel. With this feature, the application can block the channel from accepting calls before making an outbound call. If an incoming call is already in progress, the application is notified and the call in progress is not affected.

1.103.1 Feature Description

The Global Call and SIP call control libraries provide IPT network devices (channels) with the ability to receive incoming calls using the **gc_WaitCall()** function. With IP call control, incoming calls are presented to the application on channels that issued **gc_WaitCall()** based on an internal algorithm. Prior to this feature, **gc_MakeCall()** would fail on a channel that was processing an incoming call. To avoid this condition, the application had to call the **gc_Close()** or **gc_ResetLineDev()** functions to disable a previously issued **gc_WaitCall()** function on a channel-by-channel basis. As a result, any incoming call that existed on the line device would be also terminated without notification.

This feature introduces a new API specific to SIP call control. This new **gc_CancelWaitCall()** function lets the application prevent incoming SIP calls on a particular IPT network device without losing any incoming calls that may have just arrived. The application can use this functionality to cancel any previously issued **gc_WaitCall()** on a channel-by-channel basis. If the application intends to make an outbound call, it issues **gc_CancelWaitCall()** prior to the **gc_MakeCall()** function. The **gc_CancelWaitCall()** failure indicates that a call is already in progress so it will not be cancelled. When the application receives the error, it does not attempt an outbound call, thus avoiding a possible glare condition.

1.103.2 API Library Changes

The **gc_CancelWaitCall()** function is introduced to cancel a **gc_WaitCall()** previously issued on an IPT network device. In addition, the GCEV_CANCELWAITCALL event is added to the library to indicate that notification of incoming calls was successfully disabled.

Name: int gc_CancelWaitCall(linedev, mode)

Inputs: LINEDEV linedev Global Call line device handle
 unsigned long mode asynchronous

Returns: 0 if successful
 <0 if unsuccessful

Includes: gclib.h
 gccerr.h

Category: Basic

Mode: Asynchronous

Dialogic® IP†

Platform and Technology: †Refer to the *Dialogic® Global Call Technology Guides* for additional information.

■ Description

The **gc_CancelWaitCall()** function cancels any previously issued **gc_WaitCall()** and disables the ability to receive incoming calls on the given line device.

Notes: 1. This function has no effect if the application did not call the **gc_WaitCall()** function to enable notification of incoming calls. Notification is disabled by default when the channel is opened.

2. The **gc_CancelWaitCall()** function is currently only supported in asynchronous mode, and only for SIP call control.

Parameter	Description
linedev	Global Call line device handle.
mode	Set to EV_ASYNC for asynchronous execution.

■ Termination Events

GCEV_CANCELWAITCALL

Indicates that the notification of incoming calls was successfully disabled.

GCEV_TASKFAIL

Indicates that the function failed. For more information, refer to the “Error Handling” section in *Dialogic® Global Call API Programming Guide*.

■ Errors

- If this function returns <0 to indicate failure, use the **gc_ErrorInfo()** function to retrieve the reason for the error; however, if a GCEV_TASKFAIL event is generated, use the **gc_ResultInfo()** function instead. Refer to the “Error Handling” section in *Dialogic® Global Call API Programming Guide*. All Global Call error codes are defined in the *gcerr.h* file, while IP-specific errors codes are specified in *gcip_defs.h*.
- If the line device is processing an incoming call, the function will either fail or generate a GCEV_TASKFAIL event. When the function fails, EGC_GLARE/IPERR_ADDRESS_IN_USE are set as the GCcclib error codes. If a GCEV_TASKFAIL event is generated, GCEV_CCLIBSPECIFIC/IPEC_InternalReasonIncomingCall are set as the GCcclib cause value.

■ Example

```
#include <stdio.h>
#include <srllib.h>
#include <gclib.h>
#include <gcerr.h>
#include <gcip.h>
#include <gcip_defs.h>

#define MAXCHAN 30 /* max. number of channels in system */
/*
 * Data structure which stores all information for each line
 */
struct linebag {
    LINEDEV ldev; /* line device handle */
    CRN crn; /* GlobalCall API call handle */
    int state; /* state of first layer state machine */
} port[MAXCHAN+1];

struct linebag *pline; /* pointer to access line device */

/*
 * Assume the following has been done:
 * 1. Open line devices for each time slot on iptB1.
 * 2. Each Line Device ID is stored in linebag structure, 'port'.
```



```

* 3. The gc_WaitCall() has been issued on each Line Device in async mode
* 4. No call exists on the Line Device
*/

int cancel_waitcall(int port_num)
{
    GC_INFO gc_error_info;
    /* GlobalCall error information data */
    /* Find info for this time slot, specified by 'port_num' */
    pline = port + port_num;
    /*
    * Cancel the wait call
    */
    if (gc_CancelWaitCall(pline->ldev, EV_ASYNC) != GC_SUCCESS) {
        /* process error return as shown */
        gc_ErrorInfo( &gc_error_info );
        printf ("Error: cancel_waitcall() - gc_CancelWaitCall() on device handle: 0x%lx,
                GC ErrorValue: 0x%hx - %s, CCLibID: %i - %s, CC ErrorValue: 0x%lx -
                %s\n", pline->ldev, gc_error_info.gcValue, gc_error_info.gcmMsg,
                gc_error_info.ccLibId, gc_error_info.ccLibName, gc_error_info.ccValue,
                gc_error_info.ccMsg);
        return (gc_error_info.gcValue);
    }

    return (0);
}

```

For more information about Global Call APIs, refer to the *Dialogic® Global Call API Library Reference* and *Dialogic® Global Call API Programming Guide*. For IP-specific call control, refer to the *Dialogic® Global Call IP Technology Guide*.

1.104 Defer the Sending of SIP Messages

With Service Update 23, the application can delay sending the appropriate response to an incoming BYE request (such as 200OK), as well as delay the sending of a BYE request until the **gc_ReleaseCallEx()** function is issued on the channel.

1.104.1 Feature Description

Call termination is accomplished using the **gc_DropCall()** and **gc_ReleaseCallEx()** functions. With the SIP protocol, the **gc_DropCall()** function handles signaling messages by default, while the **gc_ReleaseCallEx()** is used to free up the resource. For instance, if call termination was initiated by the remote side (the remote side sent the BYE request), then the **gc_DropCall()** function will send out a 200OK response. If the call termination was initiated locally, the **gc_DropCall()** function will send out the BYE request.

When a call is terminated using the **gc_DropCall()** function, an incoming call could be received before the application has a chance to call the **gc_ReleaseCallEx()** function to release resources. In situations where there is a lot of call traffic, the incoming call is rejected because even though the last call was dropped, channels are not yet available since the resource has not been released.

By enabling this feature, the **gc_ReleaseCallEx()** function is responsible for the sending of SIP messages, in addition to the freeing of resources.

This allows the application to defer sending the appropriate response to an incoming BYE or CANCEL request or to the BYE and CANCEL requests themselves, or to DECLINE the incoming INVITE, depending on the call state (refer to the [Deferring SIP Signaling](#) section).

Notes: 1. Once enabled, this feature applies to the entire virtual HMP board.

2. The **gc_ReleaseCallEx()** function is the preferred equivalent to the deprecated **gc_ReleaseCall()** function.

1.104.2 Deferring SIP Signaling

When **gc_DropCall()** is issued, GCcclib sends out a SIP message (BYE-Discard/CANCEL/200OK) depending on the state of the call. Prior to this feature, the behavior was as follows:

- Connected state – when **gc_DropCall()** is issued to initiate the call termination, the BYE message will be sent out. The GCEV_DROPCALL event will be generated once the 200OK response is received.
- Disconnected state – when the BYE request is received in the connected state, the GCEV_DISCONNECTED will be sent to the application and the state will transition to the Disconnected state. The **gc_DropCall()** function will send the 200OK response and the GCEV_DROPCALL event will be generated immediately.
- Offered/Accepted state – when **gc_DropCall()** is issued in the one of these states, the “603-Discard” message will be sent out to reject the incoming call. The GCEV_DROPCALL event will be generated once the ACK is received.
- Dialing state – when **gc_DropCall()** is issued after **gc_Makecall()** is issued to initiate the call termination, the CANCEL message will be sent out to cancel the request to start a dialog.

By enabling this feature as described in [Enabling SIP Deferral](#), all the SIP messages will be deferred to the **gc_ReleaseCallEx()** function. The new behavior is:

- Connected state – when **gc_DropCall()** is issued, the media session will be terminated and the GCEV_DROPCALL event will be generated. When the **gc_ReleaseCallEx()** is issued, the BYE message will be sent out. The GCEV_RELEASECALL event will be generated when the 200OK response is received.
- Disconnected state – after the BYE is received in the connected state, the **gc_ReleaseCallEx()** will send the 200OK final response. The GCEV_RELEASECALL event will be generated immediately.
- Offered/Accepted state – when the **gc_ReleaseCallEx()** function is issued in the one of these states, the “603-Discard” message will be sent out to reject the incoming call. The GCEV_RELEASECALL event will be generated when the ACK is received.
- Dialing state – when **gc_ReleaseCallEx()** is issued after **gc_Makecall()** is issued to initiate the call termination, the CANCEL message will be sent out to cancel the request to start a dialog.

1.104.3 Enabling SIP Deferral

To defer sending SIP messages until the **gc_ReleaseCallEx()** function is issued, this feature must be enabled for all channels using the **gc_SetConfigData()** function.

The existing set ID IPSET_CONFIG with the new parameter ID, IPPARM_SIGNALING_DEFERRED, are used to dynamically enable and disable this feature on a virtual board basis. By default, this feature is disabled. This new parameter ID is inserted into the GC_PARM_BLK data structure using **gc_insert_parm_ref()**. The following table shows the new parameter ID in the IPSET_CONFIG parameter set.

Parameter ID	Data Type & Size	Description	SIP/H.323
IPPARM_SIGNALING_DEFERRED	Type: int Size: sizeof(int)	This parameter is used for enabling or disabling the deferral of SIP signaling messages when gc_DropCall() is issued. Possible values are: <ul style="list-style-type: none">• GCPV_ENABLE - The signaling messages (BYE/Decline/CANCEL/200OK, etc) will not be sent out when the gc_DropCall() is issued. They will be deferred to gc_ReleaseCallEx().• GCPV_DISABLE – Default. The signaling messages (BYE/Decline/CANCEL/200OK) will be sent when the gc_DropCall() is issued.	SIP only

Example Code

This example demonstrates how to enable the deferral of sending SIP messages during call termination.

```
#include <stdio.h>
#include <srllib.h>
#include <gcplib.h>
#include <gcerr.h>
#include <gcip.h>
#include <gcip_defs.h>

/*
 * Assume the board device iptB1 has been opened
 */

int defer_signaling()
{
    GC_PARM_BLK parmblkp = NULL; /* input parameter block pointer */
    long request_id = 0;
    gc_util_insert_parm_val(&parmblkp,
        IPSET_CONFIG,
        IPPARM_SIGNALING_DEFERRED,
        sizeof(int),
        GCPV_ENABLE);

    if (gc_SetConfigData(GCTGT_CCLIB_NETIF, boardDev, parmblkp, 0 /*timeout*/,
        GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC) != GC_SUCCESS)
    {
```

```

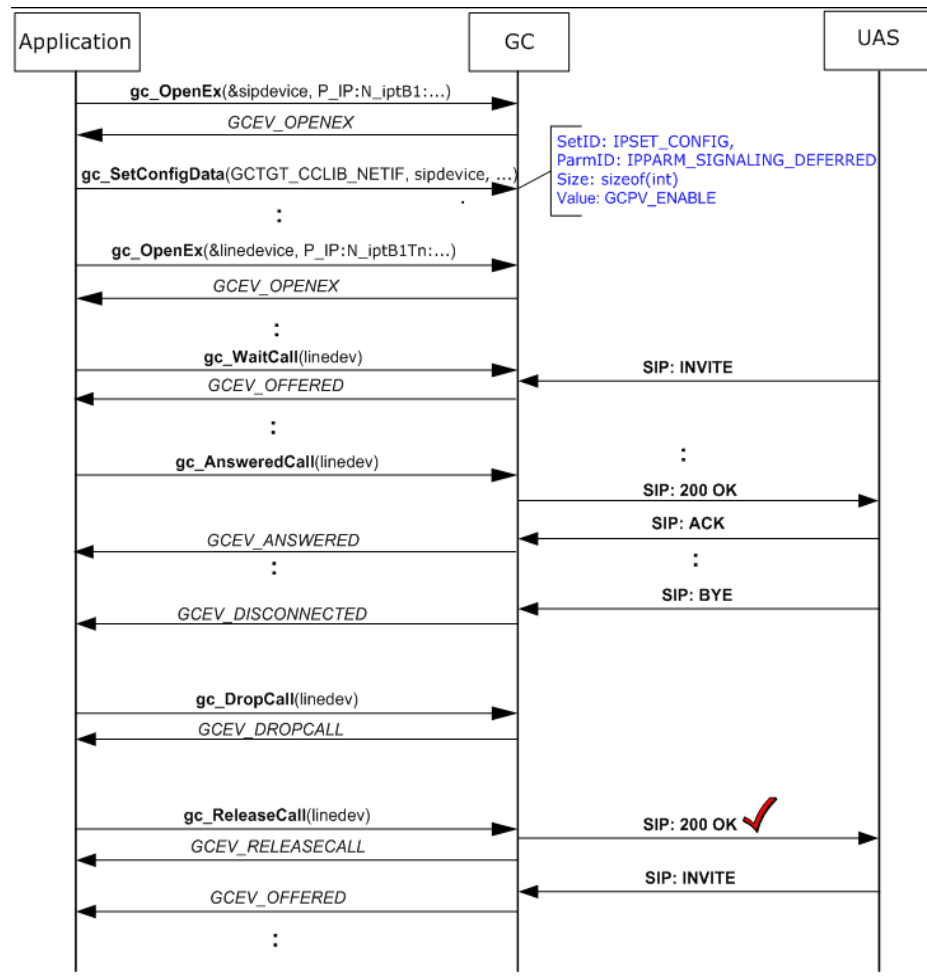
        // handle error...
    }

    return (0);
}

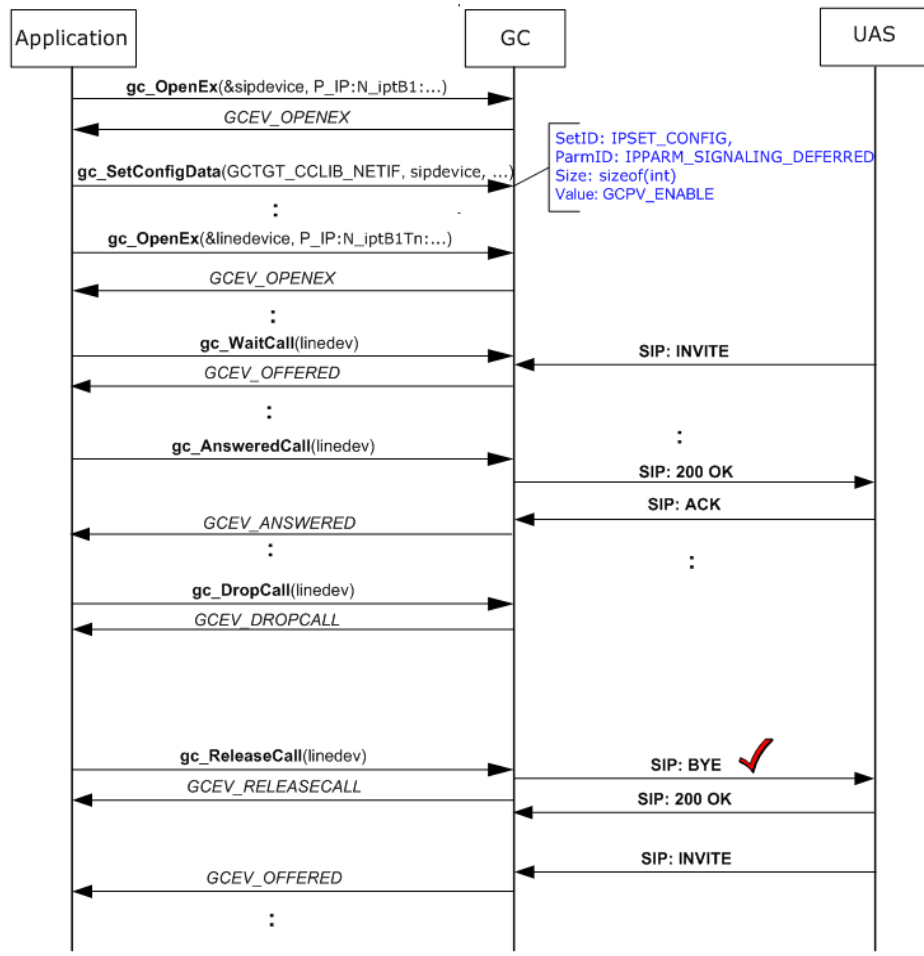
```

High-level Scenarios

The following diagrams illustrate deferrals of a 200OK response and a BYE request, respectively. Other possible scenarios under different call states are not shown. For additional information about Global Call IP and basic call control scenarios, refer to the *Dialogic® Global Call IP Technology Guide*.



The following scenario demonstrates a BYE request.



1.105 Support for Dynamic Selection of Outbound SIP Proxy

With Service Update 23, the application can select an outbound SIP proxy server on the Dialogic® HMP virtual board device dynamically. If an outbound SIP proxy server was selected at board initialization it will be overridden, otherwise it will be selected for the first time.

1.105.1 Feature Description

Currently, the outbound proxy address setting is established statically as the Dialogic® HMP board is initialized by setting a specific outbound proxy IP address or a host name in

the IP_VIRTBOARD structure that is used by the **gc_Start()** function. Afterwards, all outgoing SIP messages will pass through the configured outbound proxy. While this method remains in effect, the ability to temporarily and dynamically select the outbound proxy IP address is introduced with this feature.

The application can now configure (or reconfigure) the proxy address to use temporarily for routing all future SIP messages on a virtual HMP board basis. The temporary proxy can be deselected dynamically to revert the outbound proxy setting to the prior setting. If a prior setting was set at virtual board initialization time, then it takes effect immediately; otherwise outbound proxy is disabled.

When the set ID and parameter ID pairs are passed to the Dialogic® HMP virtual board via the **gc_SetConfigData()** function, outbound messages on all IP channels are routed via the newly set outbound proxy server. The same method is used to deselect the alternate outbound proxy. Once the outbound proxy selection is in place, all supported SIP requests and all outbound SIP responses, within a dialog or not, are affected.

Note: A Contact header field in a 200OK response from the called party (User Agent Server, or UAS) resulting from a Dialogic® HMP INVITE will cause the ACK message and all future requests out of Dialogic® HMP to bypass the proxy and go directly to the UAS for the remaining of the dialog.

Once enabled, this feature applies to the following outgoing SIP requests, in addition to SIP responses out of Dialogic® HMP software.

- 1PCC mode
ACK, INFO, INVITE, OPTIONS, REFER, REGISTER, BYE, NOTIFY, SUBSCRIBE and CANCEL.
- 3PCC mode
ACK, INFO, INVITE, OPTIONS, REFER, REGISTER, BYE, NOTIFY, SUBSCRIBE, UPDATE, PRACK and CANCEL.

Limitations

The following limitations apply to this feature:

- The explicit manipulation of Route and Record-Route headers in the application will override the proxy settings.
- This feature is disabled by default. In order to have the ability of setting an alternate proxy dynamically, explicit feature enablement must be done at virtual board initialization time, using the IP_VIRTBOARD structure in the **gc_Start()** function.
- The new version of the IP_VIRTBOARD data structure must be used to enable this feature.

1.105.2 API Changes to Enable Proxy Override

To allow for backwards compatibility, a new version of the IP_VIRTBOARD data structure is introduced as indicated below. Setting the IP_VIRTBOARD structure to this version makes the new mask, `dynamic_outbound_proxy_mask`, available to the application.

In addition, a new IP_PROXY_INFO data structure is added to the API. This data structure is used with the **gc_SetConfigData()** function to dynamically select or deselect the alternate outbound proxy. The in-line function call **INIT_IP_PROXY_INFO()** is used to initialize this data structure.

New Version Define

```
#define VIRTBOARD_VERSION_DYNAMIC_OUTBOUND_PROXY_SUPPORT 0x10f
```

New Mask

```
typedef struct
{
    unsigned short    version;
    unsigned int      total_max_calls;
    unsigned int      h323_max_calls;
    unsigned int      sip_max_calls;
    IP_ADDR localIP;
    .....
    .....
    EnumSIP_Enabled  E_SIP_UPDATE_Access;
    unsigned int      dynamic_outbound_proxy_mask;
} IP_VIRTBOARD;
```

Once the feature is enabled, the dynamic outbound proxy can be selected or deselected at any time using an existing set ID and two new parameter IDs as detailed in the following sections.

New Data Structure

```
typedef struct
{
    unsigned long      ulVersion;
    unsigned short      usPort;
    EnumProxyProtocol   eProtocol;
    EnumProxyAddressType eAddrType;
    char                arrAddr[CCLIB_SIP_TRANSPORT_IPSTRING_LEN];
    char                arrHostName[CCLIB_SIP_MAX_HOST_NAME_LEN];
} IP_PROXY_INFO, *pIP_PROXY_INFO;
```

■ Field Descriptions

The fields of IP_PROXY_INFO are described as follows:

ulVersion

identifies the version of the data structure implementation

usPort

identifies the port to which the proxy protocol is assigned

eProtocol

identifies the proxy protocol

eAddrType

qualifies the eProtocol field; use ePROXY_ADDRESS_TYPE_IP

arrAddr
used to enter a hard-coded proxy IP address

arrHostName
sets the specified hostname as the SIP outbound proxy instead of arrAddr. If arrAddr is set to 0, this hostname is resolved as the outbound proxy address; otherwise this field is ignored. The default value is NULL.

New Parameter Defines

The capability to override the proxy is accomplished by the set ID IPSET_PROXY_INFO, used for configuring proxy information, and two new parameter IDs IPPARM_PROXY_ACTION and IPPARM_PROXY_INFO. The set ID and parameter ID pair are inserted into the GC_PARM_BLK data structure. Two new values, IP_PROXY_SELECT and IP_PROXY_DESELECT, are added to IPPARM_PROXY_ACTION. The IP_PROXY_SELECT value instructs the library to dynamically choose the proxy, and the IP_PROXY_DESELECT value deselects it. The following table shows the parameter IDs in the IPSET_PROXY_INFO parameter set.

Parameter ID	Description	SIP/H.323
IPPARM_PROXY_ACTION (0x01)	Determines whether to use the proxy for the specific request. Valid values: IP_PROXY_USE 0x00 IP_PROXY_BYPASS 0x01 IP_PROXY_SELECT 0x02 IP_PROXY_DESELECT 0x03	SIP only
IPPARM_PROXY_INFO (BASE_SETID+41)	Denotes the proxy information structure. Valid value: IP_PROXY_INFO data structure	SIP only

1.105.3 Dynamic Proxy Selection Sample Code

This example demonstrates how to enable the feature using the IP_VIRTBOARD structure. Here, the new mask is set to ENUM_Enabled and the version is set to VIRTBOARD_VERSION_DYNAMIC_OUTBOUND_PROXY_SUPPORT.

```
GC_START_STRUCT gclib_start;
IPCCLIB_START_DATA cclibStartData;
IP_VIRTBOARD virtBoards[1]; // only 1 NIC supported in current release
memset(&cclibStartData,0,sizeof(IPCCLIB_START_DATA));
memset(virtBoards,0,sizeof(IP_VIRTBOARD));

INIT_IP_VIRTBOARD(virtBoards);
virtBoards[0].total_max_calls = g_h323count + g_sipcount;
virtBoards[0].h323_max_calls = g_h323count;
virtBoards[0].sip_max_calls = g_sipcount;
virtBoards[0].sip_signaling_port = g_sipport;
virtBoards[0].sip_msginfo_mask = IP_SIP_MSGINFO_ENABLE | IP_SIP_MIME_ENABLE;
virtBoards[0].E_SIP_OPTIONS_Access = ENUM_Enabled;
virtBoards[0].dynamic_outbound_proxy_mask = ENUM_Enabled;
virtBoards[0].E_SIP_DefaultTransport = ENUM_UDP;
virtBoards[0].sip_signaling_port =5060;
virtBoards[0].version = VIRTBOARD_VERSION_DYNAMIC_OUTBOUND_PROXY_SUPPORT;

INIT_IPCCLIB_START_DATA(&cclibStartData, 1, virtBoards);
cclibStartData.max_parm_data_size = 512;
```



```

CCLIB_START_STRUCT cclib_start[]={{"GC_H3R_LIB", &cclibStartData},
                                   {"GC_IPM_LIB", NULL},
                                   {"GC_DM3CC_LIB", NULL}};

gclib_start.num_cclibs = 2;
gclib_start.cclib_list = cclib_start;

if (gc_Start(&gclib_start) != GC_SUCCESS)
{
    // Handle error
}

```

The following example demonstrates how to set the dynamic proxy information to be selected:

```

// Declare a new structure
IP_PROXY_INFO proxyinfo;

// Open the board device and get the handle
if(gc_OpenEx( &g_sipbd, ":N_iptB1:P_IP", EV_SYNC, NULL) == -1)
{
    // Handle Error
}

// Select the proxy action to select, which will cause the proxy information to be used in
// setting the outbound details
// Insert this information into the GC Parameter block
rc = gc_util_insert_parm_val(&l_pParmBlk,
                            IPSET_PROXY_INFO,
                            IPPARM_PROXY_ACTION,
                            sizeof(char),
                            IP_PROXY_SELECT);

if (rc < 0)
{
    // Handle error
}

// Initialize the proxy structure using the macro INIT_IP_PROXY_INFO(&proxyinfo);

// Fill in the appropriate values in the structure
strcpy(&(proxyinfo.arrAddr[0]), "146.152.97.100");
proxyinfo.eAddrType = ePROXY_ADDRESS_TYPE_IP;
proxyinfo.eProtocol = ePROXY_PROTOCOL_UDP;
proxyinfo.usPort = 5060;

// Insert the proxy info structure into the GC Parameter Block
rc = gc_util_insert_parm_ref_ex (&l_pParmBlk,
                                IPSET_PROXY_INFO,
                                IPPARM_PROXY_INFO,
                                sizeof(IP_PROXY_INFO),
                                (void*) &(proxyinfo)
                                );

if (rc < 0)
{
    // Handle Error
}

// Use the parameter block from above and send it down to the IPCCLIB
rc = gc_SetConfigData(GCTGT_CCLIB_NETIF,
                     g_sipbd,
                     l_pParmBlk,
                     0,
                     GCUPDATE_IMMEDIATE,
                     &l_requestID,

```

```

        EV_ASYNC);
if(rc < 0)
{
    // Handle error
}

```

The following example shows how to disable the currently active proxy selection:

```

// Set the appropriate value for the proxy action
rc = gc_util_insert_parm_val(&l_pParmBlk,
    IPSET_PROXY_INFO,
    IPPARM_PROXY_ACTION,
    sizeof(char),
    IP_PROXY_DESELECT);

if (rc < 0)
{
    // Handle error
}

// Use the parameter block from above and send it down to the IPCCLIB
rc = gc_SetConfigData(GCTGT_CCLIB_NETIF,
    g_sipbd,
    l_pParmBlk,
    0,
    GCUPDATE_IMMEDIATE,
    &l_requestID,
    EV_ASYNC);

if(rc < 0)
{
    // Handle error
}

```

For more information about Global Call IP, refer to the following documents:

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

1.106 Retrieving SIP Inbound RFC 2833

With Service Update 23, the application has the ability to retrieve the RFC 2833 payload type value specified by a remote SIP user agent, using Global Call first party call control (1PCC). Since the ability to set the RFC 2833 payload type on outgoing media streams is already available in 1PCC, applications can take advantage of this new feature to match the outgoing RFC 2833 payload type with the RFC 2833 payload type of the incoming media stream, if its mapping is available in an incoming Session Description Protocol (SDP).

1.106.1 Feature Description

In a typical RFC 2833 application, the IP gateway detects DTMF on the incoming circuits and converts them as RTP payload as described in the RFC 2833, instead of embedding the digits as part of a regular RTP audio payload. On the opposite direction, the gateway recreates the DTMF tones injecting them into PSTN circuits.

RFC 2833 specifies the use of a dynamic payload type and as such its mapping is specified in advance of the media session. The RFC 2833 payload type is described in a SDP media attribute in a SIP message. While it is possible to set the RFC 2833 payload type mapping of an outgoing SDP with Global Call 1PCC mode, it was not possible to retrieve it from an incoming SDP.

With this feature, the application can retrieve the incoming RFC 2833 payload type on a call (CRN) basis using the existing API **gc_GetUserInfo()** with set ID: IPSET_DTMF and parameter ID: IPPARM_DTMF_RFC2833_PAYLOAD_TYPE. If available, the RFC 2833 payload type of the remote User Agent Client (UAC) can be retrieved on the inbound side upon receipt of a GCEV_OFFERED event. Alternatively the outbound side can retrieve the remote User Agent Server (UAS) RFC 2833 payload type upon receipt of a GCEV_CONNECTED event.

To enable the feature, the application follows the same procedure for the manipulation of the local RFC 2833 payload type by calling the **gc_SetUserInfo()** function with set ID: IPSET_DTMF and parameter ID: IPPARM_SUPPORT_DTMF_BITMASK with the value set as IP_DTMF_TYPE_RFC_2833 after an IPT network device is opened; that is, after the GCEV_OPENEX event is received on the channel device (logical board number and logical channel number). Once this is accomplished, the application can set its own RFC 2833 payload type and now also retrieve the remote RFC 2833 payload type.

Note: If an incoming SDP does not contain RFC 2833 payload type mapping, the **gc_GetUserInfo()** function will return the IPERR_INVALID_STATE error code. This error code is retrieved by calling the **gc_ErrorInfo()** function.

For more information about Global all APIs, refer to the *Dialogic® Global Call IP Technology Guide*, *Dialogic® Global Call API Programming Guide*, and *Dialogic® Global Call API Library Reference*.

Code Examples

The following example shows how to enable RFC 2833 payload type manipulation:

```
gc_util_insert_parm_val(&parmbkp, IPSET_DTMF, IPPARM_SUPPORT_DTMF_BITMASK,
                      sizeof(char), IP_DTMF_TYPE_RFC_2833);

if (gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, parmbkp, GC_ALLCALLS) != GC_SUCCESS) {
    // process error
}

gc_util_delete_parm_blk(parmbkp);
```

The following example shows how to retrieve the remote payload type from an incoming SIP message SDP carrying the appropriate media attribute with the RFC 2833 payload type mapping in an INVITE.

```
int payloadType = 0;
case GCEV_OFFERED:
    INIT_GC_PARM_DATA_EXT(&parmdata);
    gc_util_insert_parm_ref(&infoparmbkp,
                          IPSET_DTMF,
                          IPPARM_DTMF_RFC2833_PAYLOAD_TYPE,
                          sizeof(int),
                          &payloadType );
```

```

if(gc_GetUserInfo(GCTGT_GCLIB_CRN, pline->call[pline->index].crn, &infoparmblkp) !=
GC_SUCCESS) {
    // Process error
}

while (t_gcParmDatap = gc_util_next_parm(infoparmblkp, t_gcParmDatap)){
    if(t_gcParmDatap->set_ID == IPSET_DTMF && t_gcParmDatap->parm_ID ==
        IPPARM_DTMF_RFC2833_PAYLOAD_TYPE){
        memcpy(&payloadType, (char*)t_gcParmDatap->value_buf, t_gcParmDatap->value_size);
        printf("Payload type retrieved for the CRN= %ld payload = %d\n ", port
            [index].crn, payloadType);
    }

    if(payloadType < 96 && payloadType > 127){
        // Invalid payload type range; process error
    }
}
gc_util_delete_parm_blk(infoparmblkp);
break;

```

Limitations

The following limitations apply:

- To retrieve the RFC 2833 payload type, the **gc_GetUserInfo()** function can only be called on CRN. If passed an IPT board device or network device handle, the function will return an error.

An application can call the **gc_GetUserInfo()** function any time after receiving a GCEV_OFFERED event (inbound) or GCEV_CONNECTED (outbound) event (outbound). Calling the function prior to receipt of the events will return the error IPERR_INVALID_STATE.

The table below lists issues that can affect the hardware and software supported in the Dialogic® PowerMedia™ HMP for Linux Release 4.1. The following information is provided for each issue:

Issue Type

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

- Known – A minor hardware or software issue. This category includes interoperability issues and compatibility issues. Known issues are still open but may or may not be fixed in the future.
- Known (permanent) – A known hardware or software issue or limitation that is not intended to be fixed in the future.
- Resolved – A hardware or software issue that was resolved (usually either fixed or documented) in this release.

Defect No.

A unique identification number that is used to track each issue reported via a formal Change Control System.

SU No.

For defects that were resolved in a Service Update, the Service Update number is shown. For defects that were resolved when the base release was generally available (before any Service Updates), a "--" is shown. For non-resolved issues, this information is left blank.

Product or Component

The product or component to which the problem relates; for example, an API.

Description

A summary description of the issue. For non-resolved issues, a workaround is included when available.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00118012	236	Firmware	There are error messages in <code>/var/log/messages</code> which appear after a load test.
Resolved	IPY00117987	236	Firmware	During video conferencing, the image is displayed in two regions instead of one.
Resolved	IPY00117984	236	Firmware	A crash in <code>ssp_x86Linux_boot</code> occurs during video testing.
Resolved	IPY00117917	236	Firmware	During video load test, there is a failure observed.
Resolved	IPY00117870	236	Firmware	During 3GP recording, the audio quality is poor and the recorded file has some artifacts.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00117846	236	Firmware	When playing an image, it only works the first time.
Resolved	IPY00117826	236	Firmware	When playing an image, there is a crash observed.
Resolved	IPY00117825	236	Firmware	When playing an image without audio, the mm_play() is failing.
Resolved	IPY00117588	236	Firmware	There are ipm channels that get stuck and recover automatically after a while.
Resolved	IPY00117649	236	Global Call	The application receives unexpected GCEV_ATTACH event after calling gc_ReleaseCallEx() .
Resolved	IPY00117979	236	Host Library	HMP doesn't allow adding multiple route header entries to outgoing SIP messages.
Resolved	IPY00117925	236	Host Library	The record route set in PRACK is not based on the corresponding provisional response received.
Resolved	IPY00117723	236	Host Library	Issue is observed with disabling active talker conference feature at the board level.
Resolved	IPY00117628	236	Host Library	When attempting to open 96 channels, each one in its own process or executable/binary, only the first 58 channels can be opened.
Resolved	IPY00117518	236	Host Library	In certain WebRTC scenarios, there is missing DMEV_PORT_CONNECT event.
Resolved	IPY00118064	236	Multimedia	There is missing MMEV_PLAY event when using MM streaming I/O interface with AMR-WB.
Resolved	IPY00117953	236	SIP Call Control	The gc_DropCall() is not generating an event when using overlap sending.
Resolved	IPY00117919	236	SIP Call Control	There is no SDP in SIP UPDATE after 180 Ringing.
Resolved	IPY00117871	236	SIP Call Control	The SIP UPDATE is no longer able to being sent.
Resolved	IPY00117860	236	SIP Call Control	HMP doesn't set either the default "Allow" header or let the application set its own with outgoing "183 Session Progress" message.
Resolved	IPY00115803	236	SIP Call Control	G.726 won't start when REMOTE_CODER info is not specified for RECEIVE ONLY connection.
Resolved	IPY00118219	236	SSP	There is audio stutter in the recorded incoming RTP stream.
Resolved	IPY00118186	236	SSP	A core dump is observed during video load test.
Resolved	IPY00118141	236	SSP	The ssp_x86Linux_boot reports excessive RTCP unknown PT 204 prints in messages file.
Resolved	IPY00118061	236	SSP	There are errors in <i>/var/log/messages</i> which appear after video load test.
Resolved	IPY00117467	236	SSP	Noise is observed during some calls when using BICC/NbUP 5 ms.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Known (permanent)	IPY00116826	236	Firmware	When using ipm_ModifyMedia() , the MMEV_PLAY event is not generated for mm_Play() . Workaround: Use ipm_Stop() and ipm_StartMedia() and do not use ipm_ModifyMedia() in this scenario.
Known (permanent)	IPY00118040	236	OA&M	The DNIXXXTEPE2HMP does not support the Hong Kong DTMF T1 CAS PDK protocol (pdk_hk_dtmf_io).
Resolved	IPY00117401	213	Firmware	When trying to execute a video without audio test case, the play completion event MMEV_PLAY is not received.
Resolved	IPY00117383	213	Firmware	Intermittent segmentation fault in the SSP during video load testing.
Resolved	IPY00117377	213	Firmware	Intermittent segmentation fault in the SSP while playing a still image.
Resolved	IPY00117362	213	Firmware	During video load testing, the MMEV_PLAY_ACK_FAIL event occurs sometimes.
Resolved	IPY00117300	213	Firmware	HMP crashes when moving video regions in a video conference.
Resolved	IPY00117286	213	Firmware	ipm_GetLocalMedia is limited to the first 16 network interfaces in the system.
Resolved	IPY00117285	213	Firmware	There is an issue playing static JPEG images in a conference.
Resolved	IPY00117156	213	Firmware	There are malformed packets being sent out to the NbUP caller when adding to an audio conference.
Resolved	IPY00117109	213	Firmware	When using BICC/NbUP, SSP is going down and calls are silent.
Resolved	IPY00117073	213	Firmware	When ISDN channel glare condition occurs, the calling channel reports error in RTF log, while gc_MakeCall() on such channel returns SUCCESS, but does not return any event to the application.
Resolved	IPY00117002	213	Firmware	Sometimes no GCEV_DROPCALL is received after a gc_Dropcall() is issued.
Resolved	IPY00116922	213	Firmware	HMP fails to start six DNI Boards occasionally when restarting.
Resolved	IPY00116840	213	Firmware	Audio artifacts are generated in a recording using AMR-WB.
Resolved	IPY00116745	213	Firmware	Audio artifacts are generated when AMR payloads include comfort noise (SID) frames.
Resolved	IPY00116740	213	Firmware	The recorded AMR-WB in DMF file format plays too fast due to wrong timestamp.
Resolved	IPY00116709	213	Firmware	Firmware crashes while recording audio or video calls with AMR-WB in Bandwidth Efficient mode and the beep sound enabled.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00116681	213	Firmware	Audio artifacts are generated when AMR payloads include comfort noise (SID) frames in an inbound AMR audio stream.
Resolved	IPY00116541	213	Firmware	Noise is observed in the RTP stream when playing a voice prompt.
Resolved	IPY00116490	213	Firmware	AMR is not changing modes properly.
Resolved	IPY00116366	213	Firmware	HMP firmware crashes when receiving AMR frames with invalid CMR values.
Resolved	IPY00117209	213	Host Library	TDM clock settings are reset to the defaults after an ipmedia stop/start.
Resolved	IPY00117161	213	Host Library	Outbound call attempts result in “All available Resource Reservations are in use” message in the RTF logs and the call attempts fail.
Resolved	IPY00116798	213	Host Library	A compilation failure is observed when including ipmlib.h.
Resolved	IPY00116741	213	Host Library	An IPv6 address is not returned when using ipm_GetLocalMediaInfo() with MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6 .
Resolved	IPY00116616	213	Host Library	When ipmedia is stopped via CLI, the HMP system occasionally reboots.
Resolved	IPY00116597	213	Host Library	While starting up DNxxxxTEPE2HMP, occasional kernel panics caused by mercd are observed.
Resolved	IPY00115335	213	Host Library	There are intermittent call failures with gc_GetResourceH(ResType=GC_VOICEDevice) .
Resolved	IPY00115919	213	Installation	There is an issue installing HMP in a 64-bit SUSE 11 system.
Resolved	IPY00117262	213	MSML	The media server responds to an INVITE containing a 4G style codec with invalid SDP, which causes the 200 OK to be ignored.
Resolved	IPY00117011	213	MSML	Codec negotiation with AMR-WB is producing an error.
Resolved	IPY00116793	213	MSML	The media disconnect of resource returns failed errors after restarting HTTP servers.
Resolved	IPY00116714	213	MSML	When fetching prompts from HTTP server, there is a crash on the MSML Media Server.
Resolved	IPY00116593	213	MSML	When removing and adding connection to a conference, there is a crash on the MSML Media Server.
Resolved	IPY00116509	213	MSML	When SIP ACK carries a hold SDP answer, the system responds with a BYE.
Resolved	IPY00116482	213	MSML	There is intermittent conference join failure on the system with “Internal Media Server Error” or no response from the MSML Media Server.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00116477	213	MSML	The MSML Media Server deadlocks after announcements are expired from cache and web server is unavailable.
Resolved	IPY00116463	213	MSML	The media disconnect of resource returns a failed error but call recording file is written.
Resolved	IPY00116443	213	MSML	There is no MSML INFO error returned when both HTTP servers are unavailable.
Resolved	IPY00116430	213	MSML	MSML announcements and recordings can fail when a single HTTP server is unavailable.
Resolved	IPY00116411	213	MSML	Unable to process any calls after HTTP server fails and recovers when using MSML.
Resolved	IPY00115963	213	MSML	The play.complete.maxtime variable was added to differentiate between normal Play termination versus termination due to Duration Exceeded.
Resolved	IPY00115874	213	MSML	When there is no SDP in Invite and video codecs are not available, the MSML call is not established.
Resolved	IPY00115554	213	MSML	HTTP "External Document Fetch Error" is returned instead of "Put Error" when web server is unavailable.
Resolved	IPY00101895	213	MSML	HTTP requests are done in serial when multiple HTTP server addresses are provided when using MSML.
Resolved	IPY00117478	213	OA&M	DNlxxxTEPE2HMP PLL fails to lock even though a green LED (in-sync) line is chosen for the netref clocksource.
Known	HMP-263	213	Firmware	As a result of AMR and AMR-WB codec issues addressed in PowerMedia HMP SU 213, CPU utilization has increased when using these codecs.
Known	HMP-245	213	Video	When attempting to add an image overlay to an MM device in the network direction with eSM_OVERLAY_DIRECTION_NETWORK, the SMEV_ADD_OVERLAY_FAIL termination event is received.
Known (permanent)	N/A	213	Fax	As of SU 213, HMP fax functionality has been deprecated and development has been capped. HMP is recommended for low capacity faxing only in a unified communications environment with recommended maximum density of 30 fax channels. For FoIP functionality (high capacity), it is recommended to use Dialogic® Brooktrout SR140.
Known (permanent)	HMP-126	213	Fax	HMP fax sends incorrect page number after receiving a fax and being requested to send a fax (turn around poll use case).

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Known (permanent)	IPY00102460	213	Installation	<p>PDKmanager fails on 64-bit OSD with DNIXXXTEPE2HMP boards; this prevents PDK protocols from being loaded which causes HMP startup errors.</p> <p>This is due to a PDKmanager dependency on the 32-bit version of 3rd party zlib and X11 packages. 64-bit OSD installation will not necessarily include the 32-bit version of these packages.</p> <p>Workaround: Install the proper version of 32-bit zlib and X11 for this OSD manually. For instance:</p> <pre># yum install zlib.i686 and # yum install X11.i686</pre>
Resolved	IPY00116257	165	IP Media	Using CODER_TYPE_AMRNB_NONE in ipm_StartMedia() results in error message.
Resolved	IPY00116226	165	IP Media	Using ipm_ModifyMedia() on AMR octet aligned receive side switches to bandwidth efficient resulting in recording garbled audio.
Resolved	IPY00115438	161	Conferencing	One way media on some long lived (over 6 days) calls in conference.
Resolved	IPY00115365	161	Conferencing	Intermittent one way audio with calls placed into conference.
Resolved	IPY00115659	161	Global Call IP	Internal library crashing while disconnecting the call due to the SIP stack failing to fetch the transaction in the connection.
Resolved	IPY00115568	161	Global Call IP	GCEV_TASKFAIL seen after T.38 re-INVITE with 488 response.
Resolved	IPY00115655	161	MSML	The MSML Media Server will segmentation fault intermittently when unjoining and joining the same connection.
Resolved	IPY00115534	161	MSML	The MSML Media Server will segmentation fault in some cases when mixing <var> and audio files.
Resolved	IPY00115417	161	MSML	The MSML Media Server will not play audio fetched via HTTP with chunked encoding.
Resolved	IPY00115371	161	MSML	The MSML Media Server stops processing calls after HTTP server fails and recovers.
Resolved	IPY00115350	161	MSML	The MSML Media Server was not allowing DTMFs to pass thru the conference when <clamp dtmf="false"> was sent.
Resolved	IPY00102771	161	MSML	The MSML Media Server responds with "Bad Request - Invalid Dialog Id Requested" during DTMF detect under load.
Resolved	IPY00102592	161	MSML	The MSML Media Server attempts to fetch HTTP announcement after 404 file not found had already returned, resulting in garbled audio being played.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00102516	161	MSML	The MSML Media Server does not refresh the session when "refresher=uas" is specified.
Resolved	IPY00102512	161	MSML	The MSML Media Server does not honor the OFFERED preferred codec list during SDP negotiation.
Resolved	IPY00102452	161	MSML	G.726 codec errors when handling calls.
Resolved	IPY00102449	161	MSML	The MSML Media Server fails to respond with a 200 OK to an INVITE with a disabled audio stream (audio m= line contains port 0).
Resolved	IPY00102398	161	MSML	The maxptime in SDP was ignored and not parsed.
Resolved	IPY00102351	161	MSML	The MSML Media Server does not refresh the session when "refresher=uas" is specified.
Resolved	IPY00102264	161	MSML	Clearing a cancelled call results in a BYE being transmitted from the Media Server.
Resolved	IPY00102159	161	MSML	The MSML Media Server unjoin failures when using wildcard "**".
Resolved	IPY00115769	161	SIP Call Control	SIP stack did not handle resends correctly for UDP transactions.
Resolved	IPY00115559	161	SIP Call Control	An application exception occurs when using tel URI in 3PCC mode.
Resolved	IPY00115365	151	Conferencing	Intermittent one way audio with calls placed into conference.
Resolved	IPY00102729	151	IP Media	Using ipm_StartMedia() for NbUP interface returns success even if it fails to start.
Resolved	IPY00102398	151	MSML	The maxptime in SDP was ignored and not parsed.
Resolved	IPY00102351	151	MSML	The MSML Media Server does not refresh the session when "refresher=uas" is specified.
Resolved	IPY00094573	151	MSML	DTMF pattern matching issues have been addressed (min, max, rtk, fdt, idt).
Resolved	IPY00057301	151	SIP	Failure to retrieve SIP headers from SIP PRACK messages.
Resolved	IPY00115515	151	Video	During video transcoding, SSP crashes on Intel E-series processors.
Resolved	IPY00101733	141	Conferencing	Generating CNFEV_LISTEN events are not implemented for MCX devices.
Resolved	IPY00101513	141	IP Media	Using ipm_GetLocalMediaInfo() with MEDIATYPE_LOCAL_RTP_INFO_ENUM_V6 does not return IPv6 addresses.
Resolved	IPY00101292	141	IP Media	Using ipm_StartMedia() with remote NbUP RTP port number 0 returns IPMEV_ERROR.
Resolved	IPY00101595	141	MSML	MSML server requires sample rate and sample size for WAV files that are played from an HTTP server. These values are no longer required.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00101523	141	MSML	The MSML server is now able to send and receive INFO messages up to 30 KB (previous limit was 4 KB).
Resolved	IPY00101338	141	MSML	A dialog.exit is sent for no apparent reason after the dialog is created for a play/collect action.
Resolved	IPY00101289	141	MSML	The application cannot play video files into a conference.
Resolved	IPY00101269	141	MSML	A conference resource leak occurs causing subsequent conferences to fail until the media server is restarted.
Resolved	IPY00101259	141	MSML	In the play element, "iterate=forever" plays only once whereas "iterate=-1" plays the audio file forever.
Resolved	IPY00101256	141	MSML	The media server fails when a user joins a conference with a payload containing "Stream" commands that set gain to zero (0) for the joining participant.
Resolved	IPY00101238	141	MSML	Audio stops recording prematurely when the maxtime attribute is set to more than 65000 seconds.
Resolved	IPY00101124	141	MSML	The maxtime attribute in the play element does not honor "iterate=-1".
Resolved	IPY00101283	141	Multimedia	A ssp_x86Linux_boot segmentation fault occurs when calling mm_Play() multiple times consecutively.
Resolved	IPY00101111	141	SIP Call Control	IP media server ignores the compact form of Session-Expires and uses default values in its response.
Resolved	IPY00100886	141	SIP Call Control	The SIP Via header in an INVITE message cannot be altered and a second Via header is added instead. Refer to the documentation update for the Dialogic® Global Call IP Technology Guide .
Resolved	IPY00101133	141	SNMP	SNMP version 1 of the following trap ipmsSysEnvRunStatusNotification has an agent-addr always set to 0.0.0.0.
Resolved	IPY00101441	141	Voice	Using dx_stopch() in ASYNC mode doesn't return completion event when calling.
Resolved	IPY00100927	141	Voice	A segmentation fault occurs when calling dx_PutStreamData() while using streaming to board functionality.
Resolved	IPY00100802	129	Board Detection	Unable to download DNI2410TEPE2HMP boards on 32-bit CentOS 6.2 operating system.
Resolved	IPY00100466	129	Board Detection	Dialogic® DNI2410TEPE2HMP boards are not detected on some systems.
Resolved	IPY00099842	129	Fax	The application receives TFX_FAXERROR without TFX_PHASEB while receiving the fax using the fax pass thru protocol.
Resolved	IPY00099784	129	Fax	DM3FAX fax channels hang in a CS_STOPD state.
Resolved	IPY00099743	129	Fax	The fx_rcvfax() function fails to return the TFX_FAXRECV event from an asynchronous call.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00101185	129	MSML	MSML AMR codec offer is sent while receiving INVITE without SDP mode-set=0,1,2,3,4,5,6,7 which is against RFC recommendation.
Resolved	IPY00101141	129	MSML	MSML dialog.exit and playback.exe events are not generated when dialogstart is quickly followed by dialogend.
Resolved	IPY00101140	129	MSML	An Intermittent MSML server crash occurs when removing parties from a conference.
Resolved	IPY00101082	129	MSML	Playing a .wav file fails unless the sample rate and size are specified in MSML.
Resolved	IPY00100883	129	MSML	MSML destroy conference fails to clean up a play resource properly when issuing a play to the conference and a destroy conference before the play completes.
Resolved	IPY00100006	129	MSML	MSML race condition causes the media server to “leak” multimedia resources and IP Media resources resulting in a 486 busy for all requests.
Resolved	IPY00099770	129	OAM	ipc message queues are not cleaned when an application core dumps.
Resolved	IPY00100146	129	OAM	HMP4.1 fails to start on IBM x3850 systems.
Resolved	IPY00100455	129	Voice	AMR narrow band playback experiences audio quality issues when VAD is enabled.
Resolved	IPY00100752	129	Voice	AMR codec compatibility issues have been resolved.
Resolved	IPY00099948	118	DM3SPP	A segmentation fault occurs in the signaling server process (SSP).
Resolved	IPY00099319	118	Global Call IP (SIP)	An application exception occurs due to array overflow on INVITEs containing From header larger than 128 bytes.
Resolved	IPY00099307	118	Global Call IP (SIP)	After calling the gc_DropCall() function, a SIP release code 422 MinSE value is received from application.
Resolved	IPY00099177	118	Global Call IP (SIP)	When issuing a ReINVITE request, a GCEV_REQ_MODIFYCALL event is reported before receiving a GCEV_CONNECTED event for the initial call. This results in a failure due to an invalid call state if the application issues the gc_AcceptModifyCall() function.
Resolved	IPY00094390	118	Installation	HMP is using Linux entropy without any code running.
Resolved	IPY00099585	118	IP Media	An application crash occurs when initializing IP media resources.
Resolved	IPY00094485	118	IP Media	A segmentation fault occurs upon application exit.
Resolved	IPY00100289	118	MSML	Noise is intermittently heard at the end of <play> function calls.
Resolved	IPY00100242	118	MSML	The <play iterate> function plays multiple audio files in incorrect order.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00099975	118	MSML	The application is not able to configure bandwidth-efficient when using the AMR codec.
Resolved	IPY00099949	118	MSML	The MSML Media Server does not handle G.726 with static payload type 2 as a local capability.
Resolved	IPY00099906	118	MSML	The MSML media server does not handle “keep alive” ReINVITE properly and restarts IP media for the SIP session.
Resolved	IPY00099830	118	MSML	The MSML Media Server fails to play audio with the content “type=audio/basic” and the http file extension <i>.ulaw</i> .
Resolved	IPY00099801	118	MSML	A race condition occurs during termination of a play dialog where the media played is located on an http server.
Resolved	IPY00099789	118	MSML	The MSML media server fails to handle a Re-INVITE with “recvonly video” in the SDP.
Resolved	IPY00099774	118	MSML	The Media Server rejects calls with “a=recvonly” in the SDP.
Resolved	IPY00099669	118	MSML	Delay when terminating a play dialog has been reduced.
Resolved	IPY00099618	118	MSML	“486 Busy” responses are returned from the Media Server for INFO messages.
Resolved	IPY00099602	118	MSML	The Media Server fails to respond to <destroyconference>, and all subsequent requests receive a “486 Busy” response.
Resolved	IPY00099563	118	MSML	Update to allow termkey=”x” (wildcard) termination for MSML audio DTMF record termination (terminate on ANY DTMF).
Resolved	IPY00099465	118	MSML	The Media Server HTTP fetched audio is always interpreted as raw Mu-law.
Resolved	IPY00099434	118	MSML	Update to Media Server to apply AGC to a conference party.
Resolved	IPY00100050	118	SIP Call Control	A SIP session timer issue results in calls expiring before their configured time.
Resolved	IPY00099861	118	SIP Call Control	Outbound SIP calls result in GCEV_TASKFAIL events.
Resolved	IPY00099736	118	SIP Call Control	The GCEV_REQ_MODIFY_CALL event is not returned to application when a re-INVITE is received.
Resolved	IPY00099601	118	SIP Call Control	Dynamic G.726 fast_start_codec is not recognized.
Resolved	IPY00099028	118	SIP Call Control	Conflicts occur with the OpenSSL library and HMP.
Resolved	IPY00099495	118	Voice	Enabling initial silence termination stops user-defined tone detection.
Resolved	IPY00099594	115	Conferencing	Firmware crashes when using image overlay with images whose width is an odd size.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00098993	108	Conferencing	In an MCX conference, an audio-only party is specified as the loudest talker and a blank screen is displayed since party has no video. Voice-activated video switching should display the loudest video party.
Resolved	IPY00094628	108	Conferencing	Automatic Gain Control (AGC) on a party in an MCX conference is not disabled by default with the MSML media server.
Resolved	IPY00094465	108	Configuration	On some systems with certain GCC 4.1 versions, the Clockdaemon is unable to configure a Dialogic® DNI board as TDM master due to an incorrect Dev mapper attributes size. This potentially causes any board to work in TDM Slave mode without the ability to synchronize clock with a remote T1/E1 switch.
Resolved	IPY00098981	108	Firmware	A missing TEC_STREAM event due to no free buffers results in a CSP stuck channel.
Resolved	IPY00094477	108	Firmware	Voice API completion events are not returned when running systems under load.
Resolved	IPY00099007	108	IP Media Session Control (RTP)	A core dump in ssp_x86Linux_boot occurs when an echo cancellation parameter is enabled in HMP.Uconfig.
Resolved	IPY00099084	108	MSML	A rounding error in conference layout sizing occurs when fractional sizes were used; causing black borders on some regions.
Resolved	IPY00098916	108	OA&M	Memory leaks occur in bmserver, RtfServer and dlgsysmonitorserver components.
Resolved	IPY00093899	108	PSTN Call Control	Unable to send NSI IE in ISRM using the gc_SetInfoElem() function prior to the gc_MakeCall() function while setting the SIC per call basis in DPNSS.
Resolved	IPY00098980	103	Global Call IP (SIP)	A GCEV_ACCEPT_MODIFYCALL event is not reported to the application after it issues a gc_AcceptModifyCall() .
Resolved	IPY00094359	103	Global Call IP (SIP)	When a SIP call ID size is more than 63 characters long, the application crashes while sending a NOTIFY ACCEPT.
Resolved	IPY00056756	103	Global Call IP (SIP)	After repeating some Call Hold/Retrieve operations, the GCEV_ACCEPT_MODIFY_CALL event is not reported after the gc_AcceptModifyCall() function returns SUCCESS.
Resolved	IPY00094016	100	Firmware	Resolved a time slot routing issue that caused firmware timeout errors.
Resolved	IPY00094202	100	Global Call	A segmentation fault occurs after calling the gc_Start() function.
Resolved	IPY00093791	100	Global Call IP	Resolved delays that occurred under a high call load when calling the gc_MakeCall() function synchronously.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00094356	100	IP Media	The firmware crashes when invalid H.264 data is received.
Resolved	IPY00094163	100	MSML	The MSML Media Server is ignoring the http Content-Type (MIME) returned from the http server.
Resolved	IPY00093971	100	MSML	The MSML Media Server crashes when <createconference> requests are sent as payloads of SIP INFO messages with audio mix and no child element.
Resolved	IPY00093859	100	MSML	MSML play and record length values are not coming through MSML properly via play.amt and record.len terminations.
Resolved	IPY00094247	100	Voice	Calling the dx_mreciottdata() function after receiving a TDX_RECORD event can cause the channel to stop returning events.
Resolved	IPY00093988	94	Conferencing	Tone clamping in conferences does not completely remove digits from the conference.
Resolved	IPY00093910	94	Conferencing	A segmentation fault (core dump) occurs when the cnf_SetDTMFControl() function API is executed after the device is opened.
Resolved	IPY00093585	94	Conferencing	Active talker video switches excessively during an MCX conference.
Resolved	IPY00093944	94	Configuration	Updated the description for T1 Framing/Algorithm parameter (0x1624) in the configuration file to match the default setting.
Resolved	IPY00093101	94	Demo	A compatibility issue with the Mirial phone caused a video play back problem when attempting to play H.264 files in the MultiMediaDemo.
Resolved	IPY00093517	94	Firmware	ClusterPkg intermittently hangs during Dialogic® DNI2410AMCTEHMP AdvancedMC Module replacement or during a system restart using dstart.
Resolved	IPY00093633	94	Global Call IP	During a T.38 ReINVITE, HMP software does not use the remote media gateway address specified in the SDP. This results in T.38 data being sent to a different IP address.
Resolved	IPY00056659	94	Global Call IP (SIP)	The gc_InvokeXfer() function generates a core dump when makecallp is NULL on GCC 4.1.1 compiled code.
Resolved	IPY00056642	94	Global Call IP (SIP)	The application crashes when calling the gc_SetConfigData() function to set the EXTENSIONEVT_SIP_18X_RESPONSE bitmask value when in 3PCC mode.
Resolved	IPY00056610	94	Global Call IP (SIP)	HMP SIP stack sends a "400 Bad Request" during an unattended call transfer scenario.
Resolved	IPY00056632	94	OA&M	Using a secondary IP address (alias) for RTP does not work.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00093800	94	Voice	A TDX_VAD event is generated immediately after calling the dx_reciottdata() function. This event should be reported to application only when the caller has started talking.
Resolved	IPY00092358	87	IP Media	A fixed race condition between an NTC message thread and worker threads cause "PIO write" errors.
Resolved	IPY00093523	87	Licensing	IP Media fails to start on systems with a large number of cores.
Resolved	IPY00093555	86	Bridging	When a dt_listen() is issued on a DNI1210TEPE2HMP or DNI2410TEPE2HMP dti channel that was in a bridge with an HMP media device, the audio path from the existing bridge might not break. As a result, the dti channel audio is not from the intended new source, but from the existing audio source.
Resolved	IPY00093244	86	OA&M	NetRef fallback is not functional with DNI boards (clockdaemon is never notified of a DLGC_EVT_NETREF1_LINEBAD to initiate fallback).
Resolved	IPY00093283	85	Firmware	A firmware crash occurs when playing a 3GP file due to memory corruption.
Resolved	IPY00093409	85	Global Call IP (SIP)	When gc_AcceptModifyCall() is called to modify the RTP port with IPSET_RTP_ADDRESS/IPPARM_REMOTE or IPPARM_LOCAL, the API fails with a IPERR_BAD_PARAM error code.
Resolved	IPY00093399	85	Installation	Dialogic RPM packages do not list the shared libraries installed.
Resolved	IPY00093382	85	IP Media	Intermittent slow responses to IP Media API calls occur when setting RTCP enhance parameters using the ipm_SetParm() function.
Resolved	IPY00092850	82	Diagnostics	The LineAdmin utility does not display all trunk information in Alarm Status view.
Resolved	IPY00092944	82	Firmware	No MMEV_VIDEO_RECORD_STARTED event is reported to application after issuing the mm_Record() function on the multimedia resource.
Resolved	IPY00092840	82	Firmware	During runtime, the HMP service generates too many TCP source restart messages. These messages clutter the system message log causing it to overwrite.
Resolved	IPY00092868	82	Firmware	Firmware crashes when playing 0 bytes of PCM files.
Resolved	IPY00092965	82	MSML	The MSML Media Server service crashes and requires a manual restart.
Resolved	IPY00091574	82	SIP Call Control	HMP and IPT SDP session ID versions are incremented by 1 when sending a Session Timer INVITE.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00056289	82	SIP Call Control	The GCEV_TASKFAIL event is received after issuing <code>gc_AcceptModifyCall()</code> followed by <code>gc_ReqModifyCall()</code> to change the channel into an inactive state.
Resolved	IPY00092565	71	Conferencing	Adding or removing the coach from a conference results in a tone being played to all participants instead of just the pupil.
Resolved	IPY00092510	71	Conferencing	The MCX conference audio gain is significantly larger than the volume gain done by a CNF conference device or CT Bus route.
Resolved	IPY00091981	71	Fax	A TFX_FAXERROR event is not returned after by the <code>fx_stopch()</code> function.
Resolved	IPY00092519	71	Firmware	An IPMEV_GET_SESSION_INFOEX event is not returned by the <code>ipm_GetSessionInfoEx()</code> function.
Resolved	IPY00092468	71	IP Media	When the application receives an IPMEV_RTCP_NOTIFY_RECEIVED or IPMEV_RTCP_NOTIFY_SENT event, and then calls the <code>ipm_GetSessionInfoEx()</code> function, it does not receive an IPMEV_GET_SESSION_INFOEX event.
Resolved	IPY00092529	71	Licensing	The media server is unable to start when the license contains both MCX (HDVoice_Conferencing) and CNF.
Resolved	IPY00054971	65	MSML	The MSML server fails to read or write media files from remote drives or shares.
Resolved	IPY00092357	65	PSTN Call Control	The ISDN host library does not wait for LineAdmin utility to complete initialization and instead assumes the link is down.
Resolved	IPY00056036	65	SIP Call Control	SDP information is missing from the second INVITE after the stack authenticates the resend of the initial INVITE.
Resolved	IPY00056033	65	SIP Call Control	No 48x response to a SIP INVITE message is received after the licensed channel limit is reached.
Resolved	IPY00056010	65	SIP Call Control	When PRACK is sent to the Dialogic® HMP software with the RACK header missing the CSeq header, it incorrectly retransmits 183 and upon timeout sends 500 internal server.
Resolved	IPY00092285	61	Codec	The decoder occasionally fails to decode MPEG4 frames from the 3G endpoint, resulting in poor video quality.
Resolved	IPY00092278	61	Codec	HMP software does not consider available MPEG4 in-band DCI and instead relies on a user-provided value even though it is invalid and causing high CPU usage.
Resolved	IPY00092200	61	Codec	H263_1998 did not work despite being a valid transcode option with DML framework.
Resolved	IPY00091982	61	Configuration	A TDX_RECORD completion event is not received after calling the <code>dx_stopch()</code> function to terminate record activity.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00091502	61	Configuration	The use of the NCM API to restart the Dialogic® Service results in a potential hang during service shutdown.
Resolved	IPY00092243	61	CSP	CSP demo functionality fails on Red Hat Enterprise 5.x systems with the GCC 4.x compiler.
Resolved	IPY00091419	61	Device Management	Calling the dev_PortDisconnect () function stops two connections (RTP forking) when transcoding is enabled.
Resolved	IPY00092209	61	Fax	Inbound faxes are received with some of the pages cut off. The missing part appears blank when displayed with an image viewer.
Resolved	IPY00092289	61	Firmware	RC2833 Receive Only Mode does not take effect when set.
Resolved	IPY00091783	61	Firmware	A firmware crash results after setting the Signal Detector minimum energy parameter (0x070b).
Resolved	IPY00091959	61	Installation	During dlstop on a SUSE Linux Enterprise Server (SLES) 11 system, the DM3 script looks for logging entry /etc/syslog.conf file but the file does not exist on SLES 11 systems.
Resolved	IPY00092280	61	IP Media	Modify media tests result in a TASKFAIL.
Resolved	IPY00091910	56	Firmware	A "click" is heard when a party is added to a CNF conference.
Resolved	IPY00091792	56	Installation	A startup error occurs when scripts are run for modules which are not applicable to the system.
Resolved	IPY00081808	56	IP Media Session Control RTP	The RTP video buffer size for the IP Media device is preset to 1500 bytes, but should be higher for fragmented packets.
Resolved	IPY00091918	56	Licensing	The DM_GetDongleIDs library function fails due to memory-related issues.
Resolved	IPY00055853	56	Licensing	When generating an additive license file, the additive license generator places the SN keyword on the second to last line. The SN keyword must be on the last line.
Resolved	IPY00091478	53	Configuration	There is no audio when using a T1 clear channel configuration with the Dialogic® DNI2410TEPEHMP Board.
Resolved	IPY00091419	53	Firmware	The host library/firmware combination does not correctly implement duplicate connects and disconnects on the same pair of ports. In this case, the duplicate connection was between IPVSC and MDRSC components inside an IP Media device.
Resolved	IPY00091445	49	Firmware	The print level prints even though the Rx Coder 0x03 has not been reserved.
Resolved	IPY00091495	46	Codecs	H.263 transcode delays occur on M3G.
Resolved	IPY00091383	46	Dialogic® HMP Software	Dialogic® HMP software does not operate as expected after the operating system time is modified.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00090932	46	Fax	The customer implemented Fax on Demand to block incoming faxes, but the HMP application receives a call and transmits a fax.
Resolved	IPY00080125	46	Fax	A race condition in T.38 fax caused send and receive errors.
Resolved	IPY00080124	46	Fax	When the application receives more than 100 pages of fax, the page number restarts from zero (0).
Resolved	IPY00091536	46	Firmware	When one party unlistens in an application with multiple listeners, others lose audio.
Resolved	IPY00091422	46	Firmware	During download, there are various prints in /var/log/messages regarding the DM3KLOAD.
Resolved	IPY00091361	46	Global Call IP (SIP)	An application error (offset 000d959, librtfmt.dll) occurs.
Resolved	IPY00091348	46	Global Call IP (SIP)	Dialogic® HMP software rejects an incoming INVITE with "603 Decline". This occurs when the application issues the API functions gc_OpenEx() , gc_Close() , gc_WaitCall() , and gc_ResetLineDev() in particular order.
Resolved	IPY00091292	46	Global Call IP (SIP)	SIP UPDATE method is not included in the Allow header (unless Session Timers are enabled). There appears to be a missing check for UPDATE enabled mask when setting UPDATE in ALLOW header.
Resolved	IPY00091162	46	Global Call IP (SIP)	When a 200 OK response is received for an UPDATE message, the GCEV_EXTENSION event is blindly sent to the application.
Resolved	IPY00091091	46	Global Call IP (SIP)	When gc_MakeCall() is issued with timeout value =0, and one of the SIP stack timer "SIP_STACK_CFG.provisionalTimer" expires, HMP does not send the SIP CANCEL message. As a result, the call cannot be cleared at far end.
Resolved	IPY00055506	46	Global Call IP (SIP)	HMP transfer disconnects after 200OK is received for NOTIFY when attempting to make an outbound transfer using NOTIFY messages to track the status to the transferring party.
Resolved	IPY00091529	46	Installation	Files specific to the AdvancedTCA release still appear in the DATA directory (/usr/dialogic/data).
Resolved	IPY00091132	46	IP Media	Audio transcoding between G.711 to G.729 is not successful.
Resolved	IPY00055625	46	IP Media	Outbound 1PCC SIP call connects as G729AB but reports the receive GCEV_EXTENSION as G729A.
Resolved	IPY00091492	46	Multimedia	No Image or blurry image is displayed when attempting to play a JPEG image file using the mm_Play() function.
Resolved	IPY00079689	46	Voice	A DNI transaction record issue causes the dx_mreciottdata() function to not operate as expected.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00091475	42	3G-324M	MPEG4 profile_level_indication byte is set to 1 regardless of requested profile level. Note: Transcoded MPEG4 streams will always have the profile_level_indication byte set to 1, simple profile level 1, but the bitstream will otherwise be correct and match requested parameters.
Resolved	IPY00091441	42	Configuration	Transparent mode failures and higher audio failure rate occurs on one time slot.
Resolved	IPY00091439	42	Configuration	Intermittent download failures occur during configuration changes using the DNI2410AMCTEHMP board. Workaround: Change /usr/dialogic/data/gnetworkonly_hmposamc_8_e1cc.config parameter from 3 to 2, for example, SetParm=0x1601, 2.
Resolved	IPY00091140	42	Fax	Improper error handling - HMP fax stops processing faxes when it receives a BAD FCS HDLC message after a TSI and before the DCS frame.
Resolved	IPY00091139	42	Fax	HMP fax send/receive failures occur causing all fax channels to be busy.
Resolved	IPY00091138	42	Fax	The fax application locks when sending a fax.
Resolved	IPY00091093	42	Fax	A fax termination issue occurs while using T.30 FAX with ECM enabled (error correction mode) on HMP.
Resolved	IPY00090932	42	Fax	The customer implemented Fax on Demand to block incoming faxes, but the HMP application receives a call and transmits a fax.
Resolved	IPY00082125	42	Fax	The fax server freezes sporadically and the HMP fax application stops responding. Only a system reboot temporarily resolves the problem.
Resolved	IPY00081685	42	Firmware	A sporadic firmware crash occurs when the application runs overlays.
Resolved	IPY00091023	42	Global Call	The gc_DropCall() function fails and causes a hung port.
Resolved	IPY00055507	42	Global Call IP (SIP)	In 1PCC mode, SIP ReINVITE Codec Reporting appears to be wrong or corrupted.
Resolved	IPY00090994	42	Voice	The dt_listen() function locks. A mutex was used in the waiting message queue instead of a condition variable.
Resolved	IPY00090953	37	3G-324M	There is a two second delay before audio starts.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00082301	37	CSP	<p>During a CSP load, <code>ec_reciottdata()</code> was set to terminate in as many as 120 different lengths for MAX_SIL termination conditions. This caused the tone templates to be exceeded which caused the function to fail.</p> <p>Workaround: The following parameters can be added to the .config file to adjust the number of Tone Templates and Defines.</p> <pre>[sigDet] SetParm=0x0700,128 ! SD_ParmMaxTnTmplts (default=128) SetParm=0x0701,128 ! SD_ParmMaxTnDefs (default=128)</pre>
Resolved	IPY00091076	37	Firmware	A wave4 test fails with an "invalid parameter handle NULL" message.
Resolved	IPY00081664	37	Firmware	Firmware crashes when the input JPEG or YUV still image file is too large to be processed by the firmware.
Resolved	IPY00090750	37	Global Call IP (SIP)	No Response is sent to invalid INVITE.
Resolved	IPY00090646	37	Global Call IP (SIP)	In 1PCC mode, the SDP answer differs for 18x and 200_OK.
Resolved	IPY00090645	37	Global Call IP (SIP)	Dialogic® HMP Software sip stack incorrectly sends a "486 Busy Here" message in response to SIP PRACK message received.
Resolved	IPY00081732	37	Global Call IP (SIP)	A GCEV_REQ_MODIFY_CALL event is reported while a previous <code>gc_AcceptModifyCall()</code> is in progress.
Resolved	IPY00055439	37	Global Call IP (SIP)	Dialogic® HMP Software cannot issue a re-INVITE on a channel after rejecting a previous remote re-INVITE.
Resolved	IPY00081656	37	Multimedia	The system hangs after issuing the <code>mm_Play()</code> function with a specific high-resolution JPEG file.
Resolved	IPY00081676	32	Device Management	The <code>dev_GetResourceReservationInfo()</code> function returns two same DMEV_GET_RESOURCE_RESERVATIONINFO events.
Resolved	IPY00081136	32	Dialogic® HMP Software	MMDemo failed to compile on RH5 Update 2 due to PThread locking code demo.
Resolved	IPY00081135	32	Dialogic® HMP Software	3GDemo fails to compile on RH5 Update 2 due to PThreads issue with locking code.
Resolved	IPY00090656	32	Firmware	A KILLTASK was received when running still image and wave tests.
Resolved	IPY00082294	32	Firmware	The application runs for a while and then stops receiving calls and other API events.
Resolved	IPY00082163	32	Firmware	The <code>mm_Play()</code> function returns a play failure for 3GP play of helix generated files.
Resolved	IPY00081807	32	Firmware	The improper de-pack of H.263 by Dialogic® HMP software resulted in video quality issues.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00081664	32	Firmware	Firmware crashes when the input JPEG or YUV still image file is too large to be processed by the firmware.
Resolved	IPY00082165	32	Global Call IP (SIP)	Dialogic® HMP software failed to generate multiple PRACK responses.
Resolved	IPY00055290	32	Global Call IP (SIP)	When the application tries to de-register binding, which was previously registered with a particular IP-PBX, the gc_ReqService() function returns success, but the GCEV_SERVICERESP event indicates error.
Resolved	IPY00082137	32	IP Media	The ipm_StartMedia() function does not use the application specified DCI value so it is not able to start decoding incoming packets over RTP.
Resolved	IPY00082088	32	IP Media	The application does not receive GCEV_ALARM events (LAN DISCONNECT) for IPT devices, when DTI devices are opened before IPT devices during the initialization process.
Resolved	IPY00080472	32	IP Media	An issue with the RV stack causes incorrect handling of unexpected synchronous messages.
Resolved	IPY00082081	32	Licensing	Dialogic® HMP software fails with a DongleManager error in the Windows event log.
Resolved	IPY00081235	32	Licensing	License activation fails when using ethernet ports. Any subsequent attempt to start services fails with an "invalid host id " error message.
Resolved	IPY00082344	32	PSTN Call Control	The gc_GetCallInfo() function returns a redirecting number when none is present in the incoming IAM message.
Resolved	IPY00081807	25	Firmware	Improper Dialogic® HMP de-pack of H.263 caused video quality issues.
Resolved	IPY00081399	23	3G-324M	A segmentation fault occurs with video transcoding.
Resolved	IPY00054750	23	3G-324M	A 3G-324M MPEG4 signaling issue occurs against Dilithium. Dilithium advertises Simple Profile Level 2 in its MPEG4 Capabilities. The problem results in no matching video caps notifying the application, and no video channel being opened.
Resolved	IPY00081880	23	Conferencing	Multimedia conferencing fails to open an mcxB1 device if license does not contain an audio only (cnfB1) license.
Resolved	IPY00081826	23	Device Management	No audio or poor audio results when two IP media devices are connected via the dev_PortConnect() function with the transcode flag set to on.
Resolved	IPY00081661	23	Device Management	A call to the dev_PortDisconnect() function results in the application crashing with a segmentation fault.
Resolved	IPY00080694	23	Dialogic® HMP software	The <i>RvSdpLogFile.log</i> file continues to grow on a per call basis using up disk space.
Resolved	IPY00081284	23	Global Call IP (SIP)	Session Timer settings appear to be reset upon a re-INVITE to the Dialogic® HMP software.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00081264	23	Global Call IP (SIP)	UAS Session Timer programming does not survive the application's re-INVITE request.
Resolved	IPY00081142	23	Global Call IP (SIP)	The Proxy Bypass feature is not working properly.
Resolved	IPY00081132	23	Global Call IP (SIP)	The SIP hold/retrieve call scenario using the gc_ReqModifyCall() and gc_AcceptModifyCall() functions does not perform as expected.
Resolved	IPY00080800	23	Global Call IP (SIP)	A duplicate "a=inactive" line occurs in an SDP message when a re-INVITE is sent using the gc_ReqModifyCall() function.
Resolved	IPY00080772	23	Global Call IP (SIP)	183 messages cannot be PRACKed if they generate a GCEV_PROGRESSING event.
Resolved	IPY00080693	23	Global Call IP (SIP)	The application cannot add Session Description Protocol (SDP) to a SIP UPDATE message.
Resolved	IPY00080468	23	Global Call IP (SIP)	Dialogic® HMP software is unable to correctly register the application on a SIP Proxy.
Resolved	IPY00080308	23	Global Call IP (SIP)	The gc_ModifyCall() function fails with SIP Session Timers due to a SIP re-INVITE collision issue.
Resolved	IPY00080141	23	Global Call IP (SIP)	The Contact field has a different address in a re-INVITE message when it changes to T.38.
Resolved	IPY00080011	23	Global Call IP (SIP)	A GCEV_TASKFAIL event is generated upon receipt of a LowBitRate HOLD re-INVITE 200_OK message.
Resolved	IPY00080010	23	Global Call IP (SIP)	Dialogic® HMP rejects a SIP BYE from Snom with a "501 unimplemented" message.
Resolved	IPY00055020	23	Global Call IP (SIP)	Proxy-Authentication for registration and re-registration encounters problems with multi-user registrations.
Resolved	IPY00054970	23	Global Call IP (SIP)	Dialogic® HMP software responds incorrectly upon receipt of a 488 message from the UAS when issuing a Session Timer re-INVITE or UPDATE message. The re-INVITE 488 message response causes the session to drop immediately instead of the Session Timer functionality dropping the session.
Resolved	IPY00054969	23	Global Call IP (SIP)	A Call Transfer issue causes REFER to fail after receipt of an INVITE.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00081277	23	Installation	<p>The Uninstall process does not remove all RPM Package Managers (RPMs) when downgrading to an earlier build (for example, uninstall SU 227 and install SU 226).</p> <p>Workaround: Prior to installing earlier builds, you should first manually remove the RPMs that were not removed during the Uninstall process. Here is the procedure:</p> <ol style="list-style-type: none"> 1. Uninstall SU 227. 2. Log out and then log in. 3. Query the dialogic RPMs that are not removed from the SU 227 Uninstall process as follows: <code>"rpm -qa grep lsb-dialogic"</code> For example on a gcc3.4.3 system, <code>[root@pylnxlab9702 ~]# rpm -qa grep lsb-dialogic lsb-dialogic-hmp-sdk-3.1.0.0-283 lsb-dialogic-hmp-lic-3.1.0.0-225_gcc3.4</code> 4. Manually remove the above RPM as follows: <code>[root@pylnxlab9702 ~]# rpm -e lsb-dialogic-hmp-sdk-3.1.0.0-283</code> <code>[root@pylnxlab9702 ~]# rpm -e lsb-dialogic-hmp-lic-3.1.0.0-225_gcc3.4</code>
Resolved	IPY00081665	23	IP Media	The application is unable to set a custom port in the REGISTRAR ADDRESS when using proxy bypass because the port number is not extracted from the requested URI.
Resolved	IPY00080339	23	IP Media	Killing an IP media application while an IP media device is running causes the firmware to get stuck. If you run an application and <Ctrl>-C out of it while an IP media device is running, the firmware does not respond and the IP media device appears to keep running. If you then re-run the same application the next ipm_StartMedia() will fail.
Resolved	IPY00081571	23	IP Media Session Control RTP	RTP alarm events are missing when Dialogic® HMP acts as sender without a valid receiver.
Resolved	IPY00081489	23	IP Media Session Control RTP	The RTP alarm is on for approximately two minutes and then automatically resets when there is no active stream.
Resolved	IPY00081756	23	Multimedia	No MMEV_PLAY event was received when an audio/video file played out after calling the mm_Play() function via streaming.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Known (permanent)	IPY00100421		Conferencing	<p>When using Automatic Gain Control (AGC) with audio HMP conferencing and Dialogic® DNI boards, the audio levels resulting from AGC and output on the Dialogic® DNI boards are slightly different when compared to the AGC levels produced from embedded audio conferencing on legacy Dialogic® DMV boards in the latest System Release software. The former produces slightly higher audio levels than with Dialogic® DMV boards when the same input is applied, and under the same conditions.</p> <p>Workaround: In order for HMP conferencing to produce AGC levels equivalent to the embedded DMV boards, the following conferencing ([0x3b]) parameters should be used in an <i>Hmp.Uconfig</i> file and HMP should be restarted:</p> <pre>[0x3b] !<add> SetParm=0x3b1a,0x4189 !Memory size reset SetParm=0x3b1b,0x20C4 !Memory size max !</add></pre> <p>Refer to Section 3.6 Preserving Data in User Configuration Files in the <i>Dialogic® Host Media Processing Software for Linux Configuration Guide</i> for information about using <i>Hmp.Uconfig</i>.</p> <p>Note: These settings are global to the HMP system, and thus apply to any conference created and each of its participants, irrespective of the source of its input or output.</p> <p>In addition, these settings are only applicable to audio conferencing and to conference parties that are configured for AGC only; otherwise they are irrelevant. Settings remain in effect for as long as HMP is up and cannot be changed at run time.</p>
Known (permanent)	IPY00081246		Voice	Call progress positive voice detection may fail half the time when running on AMR coders. Positive voice detection/ PAMD does not work reliably on the following AMR bit rates: 5.9k, 6.7k, 7.4k, 7.95k
Known	IPY00081257		3G-324M	A system crash occurs while playing .3gp CIF files.
Known	IPY00080144		Global Call IP	The gc_SetConfigData() function fails when trying to set transparent mode on a channel. Although the software release does not require that gc_SetConfigData() be called to set transparent mode, the function should not fail and should return the proper events for backward compatibility.
Known	IPY00081262		Installation	Running the drvtrace utility causes the system to hang with a Linux kernel panic.
Known	IPY00081247		Installation	Boot errors and warnings occur throughout runtime.

Issues Sorted by Type, Dialogic® PowerMedia™ HMP for Linux Release 4.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Known	IPY00080915		Installation	The CLI commands ipmedia stop and ipmedia start do not return a confirmation event when the command has completed.
Known	IPY00080118		IP Media	IPMEV_TELEPHONY_EVENT is not generated if an RTP sequence number rollover happens in the middle of an RFC 2833 digit.
Known	IPY00081088		Multimedia	The mm_Record() function fails if no in-band DCI value is available.
Known	IPY00080004		Multimedia	The mm_Play() function intermittently returns with an MMEV_PLAY_ACK_FAIL event.
Known	IPY00079987		Voice	Positive voice detection (PVD) and/or Positive Answering Machine Detection (PAMD) intermittently fails with CAS due to test time outs.

The documentation updates for Dialogic® PowerMedia™ HMP for Linux Release 4.1 are divided into the following sections, which correspond to the top level categories used on the online documentation navigation page:

- [System Release Documentation](#) 252
- [Installation and Configuration Documentation](#) 253
- [OA&M Documentation](#) 254
- [Programming Libraries Documentation](#) 255
- [Remote Control Interfaces Documentation](#) 259
- [Demonstration Software Documentation](#) 259

3.1 System Release Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- [Dialogic® Host Media Processing Software Release 4.1LIN Release Guide](#)

3.1.1 Dialogic® Host Media Processing Software Release 4.1LIN Release Guide

With Service Update 87, update to Section 2.2, “Basic Software Requirements”
Add CentOS 5 Update 6 64 bit under the Supported Operating Systems heading.

With Service Update 165, update to Section 2.2, “Basic Software Requirements”
Replace the list of supported operating systems with the following (32-bit and 64-bit)
under the Supported Operating Systems heading.

32-bit operating system

- Red Hat Enterprise Linux Release 5 Update 2 or greater (AS/ES/WS)
- Red Hat Enterprise Linux Release 6 Update 2 or greater (AS/ES/WS)
- Community ENTERprise Operating System (CentOS) 5 Update 2 or greater
- Community ENTERprise Operating System (CentOS) 6 Update 2 or greater
- SUSE Linux Enterprise Server 11

64-bit operating system

- Red Hat Enterprise Linux Release 5 Update 4 or greater (AS/ES/WS)
- Red Hat Enterprise Linux Release 6 Update 2 or greater (AS/ES/WS)
- Community ENTERprise Operating System (CentOS) 5 Update 4 or greater
- Community ENTERprise Operating System (CentOS) 6 Update 2 or greater

With Service Update 213, update to section 2.1, “Basic Hardware Requirements”
Replace the minimum required memory with 8 GB of RAM.

3.2 Installation and Configuration Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- [Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide](#)
- [Dialogic® Host Media Processing Software for Linux Configuration Guide](#)
- [Dialogic® Global Call CDP Configuration Guide](#)

3.2.1 Dialogic® Host Media Processing Software Release 4.1LIN Software Installation Guide

Installation Package Policy

Dialogic® PowerMedia™ HMP for Linux Release 4.1 is an RPM-based installation packaged delivered as a g-zipped tar (.tgz) for installing HMP on an existing Linux system. It is recommended that users apply required updates in line with their applicable security policy/policies and to ensure that the updates are tested on a non-production HMP server prior to deployment. It is also recommended that a system backup and rollback procedure be put into place prior to deployment, in the event that any issues arise as a result of any updates being applied in production servers. Any issue(s) affecting the operation of HMP due to a security update should be reported to Dialogic.

With Service Update 86, a new version of this document is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.2.2 Dialogic® Host Media Processing Software for Linux Configuration Guide

Service Update 118 addresses a capitalization error for *Hmp.Uconfig*. (IPY00082353)
Hmp.Uconfig should replace *HMP.Uconfig* in five places in the document.

Service Update 108 adds the following new topic to Section 10.1.1 High Channel Density Recommendations: (IPY00099507)

Disk IO Performance

It is recommended that the disk subsystem's queue and read ahead buffers be increased for systems with more than 1000 channels. To do this, create files in the `/sys/block/` directory using the following command:

```
echo 1024 > /sys/block/hda/queue/nr_requests  
echo 256 > /sys/block/hda/queue/read_ahead_kb
```

Note: Replace `hda` with the name of the device where HMP will retrieve and store files, logs, etc.

3.2.3 Dialogic® Global Call CDP Configuration Guide

There are currently no updates to this document.

3.3 OA&M Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- [Dialogic® Host Media Processing Diagnostics Guide](#)
- [Dialogic® Standalone License Server User's Guide](#)

3.3.1 Dialogic® Host Media Processing Diagnostics Guide

There are currently no updates to this document.

3.3.2 Dialogic® Standalone License Server User's Guide

Service Update 129 adds this new document to the bookshelf.

3.4 Programming Libraries Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- [Dialogic® 3G-324M API Programming Guide and Library Reference](#)
- [Dialogic® Conferencing API Programming Guide and Library Reference](#)
- [Dialogic® Continuous Speech Processing API Library Reference](#)
- [Dialogic® Continuous Speech Processing API Programming Guide](#)
- [Dialogic® Device Management API Library Reference](#)
- [Dialogic® Digital Network Interface Library Reference](#)
- [Dialogic® Fax Software Reference](#)
- [Dialogic® Global Call API Library Reference](#)
- [Dialogic® Global Call API Programming Guide](#)
- [Dialogic® Global Call E1/T1 CAS/R2 Technology Guide](#)
- [Dialogic® Global Call IP Technology Guide](#)
- [Dialogic® Global Call ISDN Technology Guide](#)
- [Dialogic® Global Call SS7 Technology Guide](#)
- [Dialogic® IP Media Library API Programming Guide and Library Reference](#)
- [Dialogic® Media Toolkit API Library Reference](#)
- [Dialogic® Multimedia API Programming Guide and Library Reference](#)
- [Dialogic® Standard Runtime Library API Library Reference](#)
- [Dialogic® Standard Runtime Library API Programming Guide](#)
- [Dialogic® Voice API Library Reference](#)
- [Dialogic® Voice API Programming Guide](#)

3.4.1 Dialogic® 3G-324M API Programming Guide and Library Reference

With Service Update 86, a new version of this document is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.4.2 Dialogic® Conferencing API Programming Guide and Library Reference

Service Update 131 adds this new combined document to support the Conferencing API. Consult the document's Revision History for a list of changes. This document replaces the *Dialogic® Conferencing API Programming Guide* and *Dialogic® Conferencing API Library Reference* on the documentation bookshelf.

3.4.3 Dialogic® Continuous Speech Processing API Library Reference

There are currently no updates to this document.

3.4.4 Dialogic® Continuous Speech Processing API Programming Guide

Update to Chapter 8, Building Applications.

Added the following sentence to Section 8.2, Compiling and Linking: When compiling an application, you must list Dialogic libraries before all other libraries such as operating system libraries.

3.4.5 Dialogic® Device Management API Library Reference

There are currently no updates to this document.

3.4.6 Dialogic® Digital Network Interface Library Reference

There are currently no updates to this document.

3.4.7 Dialogic® Fax Software Reference

There are currently no updates to this document.

3.4.8 Dialogic® Global Call API Library Reference

There are currently no updates to this document.

3.4.9 Dialogic® Global Call API Programming Guide

Update to Chapter 12, Building Applications.

Added the following sentence to Section 12.1, Compiling and Linking: When compiling an application, you must list Dialogic libraries before all other libraries such as operating system libraries.

3.4.10 Dialogic® Global Call E1/T1 CAS/R2 Technology Guide

There are currently no updates to this document.

3.4.11 Dialogic® Global Call IP Technology Guide

With Service Update 131, update to **section 9.2.25**, “IPSET_SIP_MSGINFO” (IPY00100886)

The following row should be added to Table 68 “IPSET_SIP_MSGINFO Parameter Set” under the IPPARM_SIP_HDR row.

Parameter IDs	Data Type & Size	Description	SIP/ H.323
IPPARM_SIP_VIA_HDR_REPLACE	Type: NULL terminated String Size: string length	Used for enabling Via header replacement. The value is the desired Via header string.	SIP

Example:

```
char *pViaHeader = "Via: SIP/2.0/UDP black.com:5060;  
branch=z9hG4bK-e5cf7-381b26b2-7028bbc1";  
gc_util_insert_parm_ref_ex(&gcParmBlk,  
    IPSET_SIP_MSGINFO,  
    IPPARM_SIP_VIA_HDR_REPLACE,  
    (unsigned long) (strlen(pViaHeader) + 1),  
    pViaHeader);  
  
if gc_SetUserInfo(GCTGT_GCLIB_CRN, crn, parmbkp, GC_SINGLECALL);  
gc_util_delete_parm_blk(parmblkp);  
{
```

3.4.12 Dialogic® Global Call ISDN Technology Guide

There are currently no updates to this document.

3.4.13 Dialogic® Global Call SS7 Technology Guide

There are currently no updates to this document.

3.4.14 Dialogic® IP Media Library API Programming Guide and Library Reference

With Service Update 129, a new version of this document is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.4.15 Dialogic® Media Toolkit API Library Reference

With Service Update 94, a new version of this document is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.4.16 Dialogic® Multimedia API Programming Guide and Library Reference

With Service Update 131, a new version of this document is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.4.17 Dialogic® Standard Runtime Library API Library Reference

There are currently no updates to this document.

3.4.18 Dialogic® Standard Runtime Library API Programming Guide

In Service Update 82, update to Section 2.6, “Performance Considerations”

The following paragraphs should be added to the end of this section:

“On Linux systems, creation of a new process linked to Dialogic libraries using the `fork()` or `vfork()` functions may have adverse effects with Dialogic software. The `fork()` function creates a child process and duplicates the parent's page table. The `vfork()` function creates a new process without copying the page table of the parent process; in both cases the parent is suspended until the child process is started or one of the `exec()` family of functions is invoked. The heavy usage of process-based programs can cause scheduling delays, performance degradation, and problems while copying memory mapped regions in the parent process. Please refer to your Linux operating system documentation for more information about these functions. Adverse effects with Dialogic software include delays in executing Dialogic APIs, and RTF logging while parent execution suspension happens.

Dialogic recommends creating a program using POSIX threads (pthreads) instead of spawning new processes. Pthread-based programs are more optimized because they are created and managed with far less system overhead than running multiple processes. Also, communication between threads is achieved easier and more efficiently than communication between processes. Threads share the same memory space and are usually faster to start and terminate. Using pthreads is the preferred option when creating a program to interact with Dialogic host runtime libraries.”

3.4.19 Dialogic® Voice API Library Reference

There are currently no updates to this document.

3.4.20 Dialogic® Voice API Programming Guide

There are currently no updates to this document.

3.5 Remote Control Interfaces Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- [Dialogic® MSML Media Server Software User’s Guide](#)

3.5.1 Dialogic® MSML Media Server Software User’s Guide

With Service Update 118, a new version of this document is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.6 Demonstration Software Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- [Dialogic® 3G-324M Multimedia Gateway Demo Guide](#)
- [Dialogic® Global Call API Demo Guide](#)
- [Dialogic® Multimedia Demo Guide](#)

3.6.1 Dialogic® 3G-324M Multimedia Gateway Demo Guide

There are currently no updates to this document.

3.6.2 Dialogic® Global Call API Demo Guide

There are currently no updates to this document.

3.6.3 Dialogic® Multimedia Demo Guide

There are currently no updates to this document.