Dialogic® PowerMedia™ HMP for Windows Release 3.0

Release Update

March 2021

05-2508-121

Copyright and Legal Notice

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 Enghouse and its affiliates or subsidiaries ("Enghouse"). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Enghouse 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 ENGHOUSE, ENGHOUSE ASSUMES NO LIABILITY WHATSOEVER, AND ENGHOUSE DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF ENGHOUSE PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Enghouse products are not intended for use in certain safety-affecting situations.

Due to differing national regulations and approval requirements, certain Enghouse 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 Enghouse at legal.operations@enghouse.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 Enghouse may infringe one or more patents or other intellectual property rights owned by third parties. Enghouse does not provide any intellectual property licenses with the sale of Enghouse products other than a license to use such product in accordance with intellectual property owned or validly licensed by Enghouse and no such licenses are provided except pursuant to a signed agreement with Enghouse. More detailed information about such intellectual property is available from Enghouse's legal department at **80 Tiverton Court, Suite 800 Markham, Ontario L3R 0G4**.

Enghouse 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, DialogicOne, Dialogic Buzz, 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 Enghouse and its affiliates or subsidiaries. Enghouse's trademarks may be used publicly only with permission from Enghouse. Such permission may only be granted by Enghouse legal department at **80 Tiverton Court, Suite 800 Markham, Ontario L3R 0G4**. Any authorized use of Enghouse's trademarks will be subject to full respect of the trademark guidelines published by Enghouse from time to time and any use of Enghouse'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. Enghouse is not responsible for your decision to use open source in connection with Enghouse products (including without limitation those referred to herein), nor is Enghouse 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.

About This Publication

This section contains information about the following topics:

Purpose

This Release Update addresses issues associated with Dialogic® PowerMedia™ HMP for Windows Release 3.0 (formerly known as Dialogic® Host Media Processing Software Release 3.0WIN). 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 new issues, if any, that may arise after the release date.

Intended Audience

This Release Update is intended for users of Dialogic® PowerMedia™ HMP for Windows Release 3.0.

How to Use This Publication

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

- Document Revision History: This section summarizes 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 3.0WIN 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 each published version of the Release Update for Dialogic® PowerMedia™ HMP for Windows Release 3.0, which is a document that is subject to updates during the lifetime of the release.

Document Rev 121, March 2021

Additional updates for Service Update 525.
In the Post Release Developments chapter, added:

- Microsoft Hyper-V Support

Document Rev 120, November 2020

Additional updates for Service Update 525.
In the Post Release Developments chapter, added:

- Microsoft Azure Support

Document Rev 119, September 2020

Additional updates for Service Update 525.
In the Post Release Developments chapter, added:

- Amazon Web Services (AWS) Support

Document Rev 118, May 2020

Updates for Service Update 525.
In the Release Issues chapter:

- Added the following Resolved Defects: HMP-1109, HMP-1105, HMP-1012, HMP-794.

Document Rev 117, February 12, 2020

Updates for Service Update 520.
In the Post Release Developments chapter, added:

- Enhanced Voice Services (EVS) Codec Support.
- Controlled Introduction (CI) Support for OpenStack/KVM Environments.
- End of Support Notification for Various Windows Operating System Versions.
- End of Support for Dialogic® D/4PCIU Boards.

In the Release Issues chapter:

- Added the following Resolved Defects: HMP-801, HMP-842, HMP-908, HMP-964, HMP-997, HMP-1013, HMP-1016, HMP-1075.

Document Rev 116, September 17, 2019

Updates for Service Update 395.
In the Documentation Updates chapter:

- Added a note in the Dialogic® Host Media Processing Configuration Guide.

Document Rev 115, July 31, 2019

Updates for Service Update 395.
In the Release Issues chapter:

- Added the following Resolved Defects: HMP-759, HMP-760, HMP-776, HMP-834, HMP-850, HMP-859, HMP-878.

In the Documentation Updates chapter:

- Added a note in the Dialogic® IP Media Library API Programming Guide and Library Reference for **ipm_getLocalMediaInfo( )** and **ipm_StartMedia( )** functions

Document Rev 114, May 28, 2019

Updates for Service Update 393.
In the Post Release Developments chapter, added:

- Support for Windows Server 2019 Operating System.

Document Rev 113, January 15, 2019

Updates for Service Update 393.
In the Post Release Developments chapter, added:

- QScript Utilities Support Deprecated.

In the Release Issues chapter:

- Added the following Resolved Defects: HMP-793, HMP-800, HMP-825, HMP-829, HMP-835.

In the Documentation Updates chapter:

- Added an update on QScript utilities in the Dialogic® Host Media Processing Diagnostics Guide.

Document Rev 112, June 22, 2018

Updates for Service Update 387.
In the Release Issues chapter:

- Added the following Resolved Defects: HMP-787, HMP-788.

Document Rev 111, June 5, 2018

Updates for Service Update 385.

In the Release Issues chapter:

- Added the following Resolved Defects: HMP-741, HMP-758.

Document Rev 110, March 27, 2018

Updates for Service Update 382.
In the Post Release Developments chapter, added:

- Installation Package Policy.

In the Documentation Updates chapter:

- Added Installation Package Policy in the Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide.

Document Rev 109, February 22, 2018

Updates for Service Update 382.

In the Post Release Developments chapter, added:

- Updated Visual C++ Libraries Runtime Components to Visual C++ 2015 Redistributable.
- Additional 64-bit Runtime API Support.

In the Release Issues chapter:

- Added the following Resolved Defects: HMP-262, HMP-535, HMP-603, HMP-626, HMP-655, HMP-657, HMP-672, HMP-673, HMP-693, HMP-694.
- Added the following Known Issues (permanent): HMP-705.

Document Rev 108, July 13, 2017

Updates for Service Update 375.

In the Post Release Developments chapter, added:

- 64-bit Runtime API Support.
- SIP TLS Wildcard Server Certificate Support.

In the Release Issues chapter:

- Added the following Resolved Defects: HMP-595, HMP-619, HMP-632, IPY00118224, IPY00118315, IPY00118443, IPY00118455, IPY00118501, IPY00118578, IPY00118608, IPY00118609.

Document Rev 107, May 18, 2017

Updates for Service Update 372.

In the Post Release Developments chapter, added:

- Support for Windows 10 and Windows Server 2016 Operating Systems.

Document Rev 106, February 23, 2017

Updates for Service Update 372.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00118527.

In the Documentation Updates chapter:

- Added an update to the Dialogic® Global Call IP Technology Guide to address IPY00118472.

Document Rev 105, February 9, 2017

Updates for Service Update 371.

In the Post Release Developments chapter, added:

- SIP TLS Version 1.2 Support.
- Ability to Handle re-INVITE Hold in SIP REFER to Support for SIP REFER for Adding a Participant in a Conference (RFC 4579, § 5.6).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00117959, IPY00117965, IPY00118039, IPY00118159, IPY00118274, IPY00118333, IPY00118341, IPY00118343, IPY00118363, IPY00118367, IPY00118392, IPY00118416, IPY00118419.
- Added the following Known Issues (permanent): IPY00118380.

Document Rev 104, July 29, 2016

Updates for Service Update 367.

In the Post Release Developments chapter, added:

- Multimedia User I/O.
- Adaptive Multi-Rate Narrowband (AMR-NB) Audio Codec for RTP.
- G.722.2 Adaptive Multi-Rate Wideband (AMR-WB) Audio Codec for RTP.
- VMware ESXi 6.x Support.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00116675, IPY00117163, IPY00117192, IPY00117194, IPY00117252, IPY00117326, IPY00117370, IPY00117388, IPY00117394, IPY00117539, IPY00117540, IPY00117541, IPY00117560, IPY00117565, IPY00117598, IPY00117627, IPY00117658, IPY00117732, IPY00117850, IPY00117895, IPY00117899, IPY00117927, IPY00118005, IPY00118065.
- Added the following Known Issues: HMP-300.

In the Documentation Updates chapter:

- Added a note in the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide requiring Windows 7 and Windows Server 2008 R2 users to install Windows security updates 3033929 and 3035131 prior to HMP installation.

Document Rev 103, September 15, 2015

Updates for Service Update 361.

In the Post Release Developments chapter, added:

- SIP TLS Certificate Verification and Post-Connect Callback Functions.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00117258, IPY00117370, IPY00117494, IPY00117496, IPY00117513.

Document Rev 102, August 10, 2015

Updates for Service Update 360.

In the Post Release Developments chapter, added:

- Support for New SIP_STACK_CFG Data Structure User-Configurable Fields.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00116605, IPY00116671, IPY00116781, IPY00116918, IPY00116942, IPY00116943 (IPY00116663), IPY00117066, IPY00117135, IPY00117243, IPY00117287, IPY00117289, IPY00117299, IPY00117422 (IPY00117116), IPY00117452 (IPY00117433), IPY00117456.
- Added the following Known Issues: IPY00117074, HMP-190.

Document Rev 101, October 17, 2014

Updates for Service Update 357.

In the Post Release Developments chapter, added:

- Windows 2003 Operating System End of Support Notification.
- Note stating the ability to handle multiple REFER messages on the same call to Support for SIP REFER for Adding a Participant in a Conference (RFC 4579, § 5.6).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00116346, IPY00116351, IPY00116402, IPY00116484, IPY00116560, IPY00116613, IPY00116619, IPY00116696, IPY00116752.
- Added the following Known Issues: IPY00116723.

Document Rev 100, September 23, 2014

Updates for Service Update 354.

In the Post Release Developments chapter, added:

- VMware ESXi Virtualization Support with Windows Server 2012 R2.

Document Rev 99, July 14, 2014

Updates for Service Update 354.

In the Post Release Developments chapter, added:

- Windows XP Operating System No Longer Supported.
- Windows Server 2003 Operating System Restrictions.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00116305, IPY00116335.
- Added the following Known Issues: IPY00116249, IPY00116484, IPY00116501, IPY00116523.
- Added the following Known Issues (permanent): IPY00116250.

Document Rev 98, January 29, 2014

Updates for Service Update 349.

In the Post Release Developments chapter, added:

- Support for Multiple NICs for Audio Media Sessions in 1PCC Mode.
- Support for SIP REFER for Adding a Participant in a Conference (RFC 4579, § 5.6).
- Support for Dialogic® DSI SS7LDH4Q Network Interface Board.
- Configuring a Network Interface in Tristate or Line Monitor Modes.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00057362, IPY00057363, IPY00100042, IPY00102738, IPY00115269, IPY00115304, IPY00115565, IPY00115666, IPY00115808, IPY00115897.
- Added the following Known Issues: IPY00115905.
- Added a Known Issue (permanent) with no defect number.

Document Rev 97, published December 12, 2013

In the Post Release Developments chapter, added and updated:

- Support for Windows Server 2012 R2 Standard and Windows 8.1 Pro Operating Systems.
- VMware ESXi Virtualization Support Enhancements.

Document Rev 96, July 29, 2013

Updates for Service Update 347.

In the Post Release Developments chapter, added:

- Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards.
- Support for Multiple NICs in IP Media Sessions.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00057320, IPY00102451, IPY00102864.

Document Rev 95, May 31, 2013

Updates for Service Update 343.

In the Post Release Developments chapter, added:

- Support for Windows Server 2012 Operating Systems.
- Increase in Concurrent Sessions on a Single Server.
- Configuring the Line Law Encoding Mode.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00057293, IPY00094373, IPY00099638, IPY00101839, IPY00101970, IPY00102042, IPY00102048, IPY00102052, IPY00102110, IPY00102111, IPY00102219, IPY00102241, IPY00102354.
- Added the following Known Issues: IPY00102428.
- Added the following Known Issues (permanent): IPY00091575, IPY00100042, IPY00100100.

In the Documentation Updates chapter:

- Added an update to the Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide to address Microsoft Intelligent Platform Management Interface (IPMI) Compatibility Device in Windows Server 2012.

Document Rev 94, January 31, 2013

Updates for Service Update 338.

In the Post Release Developments chapter, added:

- Support for Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP Digital Network Interface Boards.
- Special SIP INVITE Request-URI Overwrite.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00057220, IPY00057256, IPY00099952, IPY00101250, IPY00101287, IPY00101299, IPY00101303, IPY00101443, IPY00101457, IPY00101863.
- Added the following Known Issue (with workaround): IPY00101008.

Document Rev 93, September 17, 2012

Updates for Service Update 328.

In the Post Release Developments chapter, added:

- MSML Server Software Support Removed.
- Support for TCP Connection Reuse in SIP.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00057170, IPY00057174, IPY00093730, IPY00098898, IPY00099145, IPY00099940, IPY00100127, IPY00100385, IPY00100449, IPY00100486, IPY00100534, IPY00100730, IPY00100775, IPY00100796, IPY00100886, IPY00100893.
- Changed the Issue Type for IPY00100857 from “Known” to “Resolved” and removed IPY00100100 as it is no longer applies to this release.
- Added the following Known issue: IPY00100540.

In the Documentation Updates chapter:

- Added an update to the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide to address DEP.
- Added an update to the Dialogic® Host Media Processing Configuration Guide to address IPY00100643.
- Added an update to the Dialogic® Global Call IP Technology Guide to address IPY00100886.
- Removed the Dialogic® MSML Media Server Software User’s Guide from the bookshelf since MSML Media Server Software support is no longer available.

Document Rev 92, August 2, 2012

Additional Update for Service Update 323.

In the Release Issues chapter:

- Added the following Known Issue: IPY00100857.

In the Documentation Updates chapter:

- Removed updates from the Dialogic® Global Call IP Technology Guide because a new version is available on the documentation bookshelf.

Document Rev 91, July 9, 2012

Update for Service Update 323.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093233, IPY00093988, IPY00098919, IPY00099495, IPY00099621, IPY00099719, IPY00099743, IPY00099842, IPY00099956, IPY00100032, IPY00100034, IPY00100045, IPY00100100, IPY00100141, IPY00100305.

Document Rev 90, May 4, 2012

Update for Service Update 320.

In the Post Release Developments chapter, added:

- Support for SIP 1xx Provisional Responses in a Ringing State.
- SIP Semi-Attended Call Transfer Support.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00093540, IPY00098933, IPY00099028, IPY00099630, IPY00099736, IPY00099807, IPY00099861, IPY00099980.

Document Rev 89, April 17, 2012

Update for Service Update 318.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00092928, IPY00093775.
- Removed IPY00099069 from the Release Issues table as it is not yet resolved. The issue, an access violation when issuing the gc_MakeCall( ) function on a SIP channel during a glare condition, was erroneously published as resolved in Service Update 317.

Document Rev 88, April 9, 2012

Update for Service Update 317.

In the Post Release Developments chapter, added:

- Increased G.729 Density.
- Overlap Send and Receive SIP Signaling Support.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056719, IPY00056766, IPY00056770, IPY00056779, IPY00056913. IPY00098827, IPY00098980, IPY00099067, IPY00099133, IPY00099149, IPY00099150, IPY00099177, IPY00099200,

IPY00099227, IPY00099307, IPY00099311, IPY00099319, IPY00099396, IPY00099572.

In the Documentation Updates chapter:

- Update to the **cnf_SetDTMFControl( )** description in the Dialogic® Conferencing API Library Reference.

Document Rev 87, published January 30, 2012

Update for Service Update 314.

In the Post Release Developments chapter, added:

- VMware ESXi Virtualization Support Enhancements.
- SIP “To tag” for Inbound Calls.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00094003, IPY00094176, IPY00098947, IPY00098952.

Document Rev 86, published December 21, 2011

Update for Service Update 313.

In the Post Release Developments chapter, added:

- IPMI Updates.
- Anti-virus Software Policy.
- User Account Control Recommendation.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056738, IPY00056756, IPY00056776, IPY00056843, IPY00056844, IPY00093233, IPY00093410, IPY00093416, IPY00093701, IPY00093823, IPY00093913, IPY00094030, IPY00094151, IPY00094154, IPY00094197, IPY00094290, IPY00094359, IPY00094589, IPY00094591, IPY00094606, IPY00094689, IPY00098894.

Document Rev 85, published September 27, 2011

Update for Service Update 310.

In the Post Release Developments chapter, added:

- Increase in Concurrent Sessions on a Single Server.
- New TDX_DRVNOMEM Event.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056539, IPY00056601, IPY00056642, IPY00056659, IPY00093453, IPY00093590, IPY00093815, IPY00093859, IPY00093902, IPY00093971 , IPY00094061, IPY00094125.

Document Rev 84, published July 29, 2011

Update for Service Update 307.

In the Post Release Developments chapter, added:

- PDK Support for Automatic Answer and Reject of Inbound Calls.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056458, IPY00056559, IPY00056584, IPY00056607, IPY00056610, IPY00056613, IPY00092371, IPY00093013, IPY00093232, IPY00093270, IPY00093288, IPY00093361, IPY00093375, IPY00093384, IPY00093385, IPY00093409, IPY00093501, IPY00093627, IPY00093672.

In the Documentation Updates chapter:

- Added a note to the io_bufp description on the DX_IOTT data structure reference page in the Dialogic® Voice API Library Reference. (IPY00093803)

Document Rev 83, published June 10, 2011

Update for Service Update 303.

In the Post Release Developments chapter, added:

- Service URN Support.
- Dialogic® D80PCIE-LS 32-bit Compatibility Support.
- Support for Initial Silence Termination.
- Updates to 64-bit Operating System Support in 32-bit Compatibility Mode.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056289, IPY00056482, IPY00056505, IPY00092139, IPY00093089, IPY00093097, IPY00093117, IPY00093128, IPY00093215, IPY00093246.

Document Rev 82, published March 29, 2011

Update for Service Update 299.

In the Post Release Developments chapter, added:

- Upgrading Intel Network Adapter Drivers on Windows Server 2008 and Windows 7.
- Increase in SUBSCRIBE and OPTIONS SIP Requests. (IPY00056119)

- SIP UPDATE Post-Connection Support. (IPY00091274)
- Added a note that all examples shown are pre-connection in Section 1.93.5, “Usage Scenarios”, on page 177.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056119, IPY00056234, IPY00056281, IPY00091574, IPY00091274, IPY00092725, IPY00092794, IPY00092835, IPY00092965, IPY00092995.

Document Rev 81, published March 1, 2011

Update for Service Update 298.

In the Post Release Developments chapter, added:

- 64-bit Operating System Support in 32-bit Compatibility Mode.
  - Note: Please refer to Known Issue, IPY00093028, in the Release Issues table for information about a video problem.

In the Release Issues chapter:

- Added the following Known Issue: IPY00093028.

Document Rev 80, published February 8, 2011

Update for Service Update 297.

In the Post Release Developments chapter, added:

- Session Timer Support SDP in ReINVITE.
- Improvement to Call Progress Analysis on Dialogic® HMP Thin Blades.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056096, IPY00056135, IPY00079902, IPY00080769, IPY00092242, IPY00092276, IPY00092511, IPY00092629.

Document Rev 79, published January 17, 2011

Update for Service Update 296.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00056010, IPY00056033, IPY00056036, IPY00092148.

Document Rev 78, published December 13, 2010

Additional Update for Service Update 291.

In the Post Release Developments chapter, added:

- Support for Dialogic® D/4PCIU Boards.

In the Release Issues chapter:

- Added the following Known Issue: IPY00092500.

In the Documentation Updates chapter:

- Removed updates from the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide because a new version is available on the documentation bookshelf.
- Added the Dialogic® Springware Architecture on Windows Configuration Guide to the documentation bookshelf for analog device support.
- Updated the **dev_Connect( )** function reference page in the Dialogic® Device Management API Library Reference for analog device support.
- Added the Dialogic® Global Call Analog Technology Guide to the documentation bookshelf for analog device support.
- Added the Dialogic® Learn Mode and Tone Set File API Software Reference to the documentation bookshelf for analog device support.
- A new version of the Dialogic® Voice API Library Reference is now available on the documentation bookshelf.
- A new version of the Dialogic® Voice API Programming Guide is now available on the documentation bookshelf.

Document Rev 77, published November 12, 2010

Updates for Service Update 291.

In the Post Release Developments chapter, added:

- Increase in Fax Channels Support.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055976, IPY00091862, IPY00092090, IPY00092209.

In the Documentation Updates chapter:

- Replaced text for Note 1 in the Introduction section of Chapter 6. “Displaying and Changing the Dialogic® HMP Software Default IP Address” in the Dialogic® Host Media Processing Administration Guide.

Document Rev 76, published October 13, 2010

Updates for Service Update 289.

In the Post Release Developments chapter, added:

- Increase in CSP Channels.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055503, IPY00055504, IPY00055887, IPY00055898, IPY00081381, IPY00091084, IPY00091103, IPY00091502, IPY00091783, IPY00091795, IPY00091823, IPY00091855, IPY00091867, IPY00091878, IPY00091932, IPY00091982.
- Added the following Known Issue: IPY00092165.

In the Documentation Updates chapter:

- Added a note about a restriction when assigning a license to a network interface card (NIC) in a system with more than five NICs to the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide and the Dialogic® Host Media Processing Administration Guide. (IPY00091711)
- Added that the Global Call SS7 binaries are now linked with the dynamic linked libraries in the Dialogic® SS7 DSI Development Package in the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide. (IPY00081381)
  - Note: This change requires that Global Call SS7 customers use the Dialogic® SS7 DSI Development Package version 5.0 or later. If a customer is using an earlier version, the Global Call SS7 server will not start during download.

Document Rev 75, did not publish

Document Rev 74, published September 8, 2010

Updates for Service Update 286.

In the Post Release Developments chapter, added:

- MSML Server Software Support.
- Monitor Mode Support for HMP Conferencing.

In the Release Issues chapter:

- Added the following Resolved Defects: Added a new topic, Avoiding UPD/RTP Port Conflicts, to Compatibility Notes to resolve IPY00091187.

In the Documentation Updates chapter:

- Removed documentation updates to the Dialogic® Host Media Processing Configuration Guide because a new version is available on the bookshelf.

Document Rev 73, published August 6, 2010

Updates for Service Update 283.

In the Post Release Developments chapter, added:

- Using IP Call Control (1PCC and 3PCC) Licenses without Media.
- Recording and Playback of G.729A Files.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00091219, IPY00091247, IPY00091278, IPY00091292, IPY00091303, IPY00091348, IPY00091360, IPY00091361.

In the Documentation Updates chapter:

- A new version of the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide is now available on the bookshelf. See the Revision History section for details.

Document Rev 72, published July 21, 2010

Additional Updates for Service Update 279.

In the Post Release Developments chapter, added:

- Support for the Dialogic® D/80PCIE-LS Media Board.

In the Release Issues chapter:

- Added the following Known Problem: IPY00081927 (IPY00081721, IPY00091533).
- Added a Intel Xeon E5500 Nehalem with QuickPath Interconnects subsection in Compatibility Notes. This is related to IPY00081927 (IPY00081721, IPY00091533).

In the Documentation Updates chapter:

- A new version of the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide is now available on the bookshelf to support the Dialogic® D/80PCIE-LS Media Board.

Document Rev 71, published July 13, 2010

Additional Update for Service Update 279.

In the Post Release Developments chapter, added:

- Virtualization Support.

Note: Feature documentation is added in this additional update. Feature support was available in Service Update 275.

Document Rev 70, published July 7, 2010

Additional Updates for Service Update 279.

In the Post Release Developments chapter, added:

- Overlap-Receive Support for Limited SIP-I Interworking Scenarios.
- Unspecified G.723.1 Bit Rate in Outgoing SIP Requests with SDP.
- Processing Multiple 18x Provisional Responses.

Note: Feature documentation is added in this additional update. Feature support was available in the initial release of Service Update 279.

Document Rev 69, published June 23, 2010

Updates for Service Update 279.

In the Post Release Developments chapter, added:

- TLS and SRTP Channel Support Increase.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00047709, IPY00047710, IPY00055506, IPY00055507, IPY00055624, IPY00055625, IPY00082125, IPY00090867, IPY00090932, IPY00091023, IPY00091050, IPY00091081, IPY00091091, IPY00091093, IPY00091132, IPY00091137, IPY00091138, IPY00091139, IPY00091140, IPY00091162.

Document Rev 68, published May 14, 2010

Updates for Service Update 275.

In the Post Release Developments chapter, added:

- Registering Authentication Data without Realm String.
- Support for a FAX Gateway.
- Handling Non-2xx Responses to T.38 Switch.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00055439, IPY00081518, IPY00081873, IPY00082301, IPY00090705, IPY00090969.

In the Documentation Updates chapter:

- Added Windows Server 2003 with Service Pack 2 to the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide.
- Added a note about time slot resource allocation limitations to the Dialogic® Conferencing API Programming Guide. (IPY00090909)

- Update to Section 3.4.9, "Dialogic® Fax Software Reference", on page 376 in support of a new fax gateway.

Document Rev 67, published April 13, 2010

Updates for Service Update 272.

In the Post Release Developments chapter, added:

- MIME Insertion in Outgoing ACK.
- Support for WaitCall Cancellation.
- Increase in TLS and SRTP Channels.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00080842, IPY00080914, IPY00081695, IPY00081732, IPY00082165, IPY00090645, IPY00090646, IPY00090750.

Document Rev 66, published March 30, 2010

Updates for Service Update 271.

In the Post Release Developments chapter:

- Removed Support for WaitCall Cancellation as support for this feature is not yet completed.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00081235, IPY00082081.

Document Rev 65, published March 19, 2010

Updates for Service Update 270.

In the Post Release Developments chapter:

- Added Support for Windows 7 and Windows Server 2008 Operating Systems.
- Added Channel Density Upgrade (G.711 Voice Only).
- Added 3PCC Support for Dynamic Selection of Outbound SIP Proxy.
- Updated 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: IPY00055020 (resolved in SU 267), IPY00055290, IPY00055436, IPY00081251, IPY00081529, IPY00082088, IPY00082126, IPY00090707.

In the Documentation Updates chapter:

- Removed the updates for the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide, because these updates have been incorporated into the revised document that is now on the online documentation bookshelf.
- Removed the updates for the Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide, because these updates have been incorporated into the revised document that is now on the online documentation bookshelf.

Document Rev 64, published February 18, 2010

Updates for Service Update 267.

In the Post Release Developments chapter:

- Added Windows XP SP3 Operating System Support.
- Added the ability to Defer the Sending of SIP Messages.
- Added a note regarding the Contact header field to Support for Dynamic Selection of Outbound SIP Proxy.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00054991 (resolved in SU 265), IPY00081665.

In the Documentation Updates chapter:

- In the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide, added Windows XP SP3 to the bulleted list under operating system support in Section 2.2, “Basic Software Requirements” and removed Section 9.2, “Dialogic® Digital Station Interface Products” because these products have been discontinued and are no longer supported.
- Added a note regarding the Contact header field to Section 4.1.1. “Setting a SIP Outbound Proxy” in the Dialogic® Global Call IP Technology Guide.

Document Rev 63, published December 11, 2009

Updates for Service Update 265.

In the Release Issues chapter:

- Added the following Resolved Defect: IPY00081465.

Document Rev 62, published November 20, 2009

Updates for Service Update 262.

In the Post Release Developments chapter:

- Added Intel Xeon 55xx Processor Support.
- Added Support for Dynamic Selection of Outbound SIP Proxy.

In the Release Issues chapter:

- IPY00054969, IPY00054970, IPY00080918, IPY00080944, IPY00081061, IPY00081096, IPY00081132, IPY00081142, IPY00081159, IPY00081264, IPY00081284.
- Updated the APIC Timer and Dialogic® HMP text in Section 2.2, "Compatibility Notes", on page 350.

In the Documentation Updates chapter:

- A new version of the Dialogic® Device Management API Library Reference is available on the documentation bookshelf.
- A new version of the Dialogic® IP Media Library API Programming Guide and Library Reference is available on the documentation bookshelf.
- A new version of the Dialogic® Multimedia API Programming Guide and Library Reference is available on the documentation bookshelf.

Document Rev 61, published September 16, 2009

Updates for Service Update 256.

In the Post Release Developments chapter, added:

- Retrieving SIP Inbound RFC 2833.
- Removed details from Support for NAT Traversal in SIP Media Session. The information was replaced by a reference to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for information about using this feature.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00080010, IPY00080011, IPY00080308, IPY00080396, IPY00080407, IPY00080468, IPY00080640, IPY00080693, IPY00080694, IPY00080772, IPY00080800, IPY00080860.

Document Rev 60, published August 28, 2009

Updates for Service Update 255.

In the Post Release Developments chapter, added:

- Runtime Support for Edge Selection in Digit Reporting
- Support for GSM Coder
- Additional Operating System Support

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00080124, IPY00080125.

In the Documentation Updates chapter:

- Added additional operating support to the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide.
- Removed an incorrect note from Section 9.2, “Input Volume Automatic Gain Control” in the Dialogic® Audio Conferencing API Programming Guide.
- Replaced a paragraph in Section 4.9.6, “Retrieving SIP Message Header Fields” in the Dialogic® Global Call IP Technology Guide. (IPY00080750)
- Updated text in Table 6, “Voice Encoding Methods” of the Dialogic® Voice API Programming Guide to show correct GSM support.
- Corrected the spelling of a field value in the Dialogic® Voice API Library Reference. (IPY00080715)
- Replaced contents of Table 8, “GSM Voice Coder Support Fields” in the Dialogic® Voice API Library Reference.

Document Rev 59, published May 28, 2009

Updates for Service Update 253.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00079937, IPY00080031, IPY00080086, IPY00080152.
- Added the following Known Issue: IPY00080175.

In the Documentation Updates chapter:

- Changes to the Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide regarding APIC Timer Compatibility.

Document Rev 58, published May 8, 2009

Updates for Service Update 251.

In the Post Release Developments chapter:

- Added Support for RFC 3326 SIP Header for BYE and CANCEL Messages.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00043527, IPY00044285, IPY00045481, IPY00078684, IPY00079186, IPY00079825, IPY00079866.

In the Documentation Updates chapter:

- Deleted the updates for the Dialogic® Voice API Programming Guide, because these updates have been incorporated into the revised document that is now on the online documentation bookshelf.
- Deleted the updates for the Dialogic® Voice API Library Reference, because these updates have been incorporated into the revised document that is now on the online documentation bookshelf.

Document Rev 57, published April 20, 2009

Updates for Service Update 248.

In the Post Release Developments chapter:

- Support for Proxy Bypass in SIP Register Requests.
- Increase in CSP Instances.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00045524, IPY00079353, IPY00079551, IPY00079764.

Document Rev 56, published April 9, 2009

Updates for Service Update 246.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00079435, IPY00079678, IPY00079763.

Document Rev 55, published March 20, 2009

Updates for Service Update 245.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00079601, IPY00079655.

In the Documentation Updates chapter:

- Added a note to the **gc_RejectModifyCall( )** function reference page in the Dialogic® Global Call IP Technology Guide. (IPY00037226)

Document Rev 54, published February 27, 2009

Updates for Service Update 241.

In the Post Release Developments chapter:

- Added Support for RFC 3311 UPDATE Message.

In the Release Issues chapter:

- IPY00033977 - Removed an inactive hyperlink in the Description column.

In the Documentation Updates chapter:

- A new version of the Dialogic® Fax Software Reference is now available on the documentation bookshelf.

Document Rev 53, published January 14, 2009

Updates for Service Update 237.

In the Post Release Developments chapter:

- Removed details from sections Native T.38 Hairpinning Support, Receive-only RFC 2833 Mode Available, and Selective Packet Filtration Method. The information was replaced by a reference to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for information about using these features.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00044782, IPY00079068, IPY00079108, IPY00079116, IPY00079123, IPY00079125, IPY00079160, IPY00079161, IPY00079177, IPY00079251, IPY00079254.

In the Documentation Updates chapter:

- Updates to the **NCM_StartSystem( )**, **NCM_StartDlgSrv( )**, **NCM_StopDlgSrv( )**, and **NCM_StopSystem( )** function reference pages in the Dialogic® Native Configuration Manager API Library Reference.
- A new version of the Dialogic® IP Media Library API Programming Guide and Library Reference is now available on the documentation bookshelf.

Document Rev 52, published December 19, 2008

Updates for Service Update 233.

In the Post Release Developments chapter:

- Added support for Exposing Raw RFC 2833 Data Using Global Call API.

In the Documentation Updates chapter:

- Updates to sections 4.6.1. “Enabling and Disabling Unsolicited Notification Events" and 4.16.2. “Getting Notification of DTMF Detection" in the Dialogic® Global Call IP Technology Guide.
- A new version of the Dialogic® Global Call E1/T1 CAS/R2 Technology Guide is now available on the documentation bookshelf.
- A new version of the Dialogic® Global Call ISDN Technology Guide is now available on the documentation bookshelf.
- A new version of the Dialogic® Global Call SS7 Technology Guide is now available on the documentation bookshelf.
- A new version of the Dialogic® Multimedia API Programming Guide and Library Reference is now available on the documentation bookshelf. This document replaces the former Multimedia API Programming Guide and the Multimedia API Library Reference.

Document Rev 51, published December 10, 2008

Updates for Service Update 232.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00040428, IPY00045369, IPY00078720, IPY00078754.

Document Rev 50, published November 20, 2008

Updates for Service Update 229.

In the Post Release Developments chapter:

- Added Support for the Samsung IAP PBX E1 ISDN Protocol.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00043492, IPY00045058, IPY00078626, IPY00078639.

Document Rev 49, published November 13, 2008

Updates for Service Update 228.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00078334, IPY00078742, IPY00078606, IPY00078827.

Document Rev 48, published November 4, 2008

Updates for Service Update 226.

In the Post Release Developments chapter:

- Made minor updates to code examples in the Section 1.96, “Support for NAT Traversal in SIP Media Session”, on page 188.

In the Documentation Updates chapter:

- Updates to Table 2. eIPM_PARM Parameters and Values in the Dialogic® IP Media Library API Programming Guide and Library Reference. (IPY00078708)

Document Rev 47, published October 23, 2008

Updates for Service Update 225.

In the Post Release Developments chapter:

- Added Support for NAT Traversal in SIP Media Session.
- Added Support for Egress 183 Session Progress Message.

In the Documentation Updates chapter:

- Updates to Section 8.3.17.3, “Forming a Destination Address String - Variance for H.323” and Section 8.3.17.4, “Destination Address Interpretation” in the Dialogic® Global Call IP Technology Guide. (IPY00045281)
- Replaced code snippet in Section 4.15.1, “Sending SUBSCRIBE Requests” in the Dialogic® Global Call IP Technology Guide. (IPY00045426)

Document Rev 46, published September 26, 2008

Updates for Service Update 221.

In the Release Issues chapter:

- Added the following Resolved Defect: IPY00044630.

Document Rev 45, published September 3, 2008

Updates for Service Update 218.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00044404, IPY00044633, IPY00044700, IPY00044705, IPY00044822, IPY00044976.

Document Rev 44, published August 6, 2008

Updates for Service Update 214.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00043787, IPY00044145, IPY00044219, IPY00044273, IPY00044101, IPY00044366, IPY00044437, IPY00044488, IPY00044516, IPY00044578, IPY00044684.

Document Rev 43, published July 9, 2008

Updates for Service Update 210.

In the Post Release Developments chapter:

- Added Added Global Call API Access to New H.323/Q.931 Message IEs.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00043399, IPY00043716, IPY00043860, IPY00044310.

In the Documentation Updates chapter:

- Updated the link to the Multimedia File Conversion Utilities in the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide. (IPY00044494)
- Added an update to the Troubleshooting chapter regarding TDM Bus Messages to ignore in Dialogic® Host Media Processing Administration Guide. (IPY00044223)
- Added an update to the Troubleshooting chapter regarding possible errors when using third-party software with FlexNET licensing in Dialogic® Host Media Processing Administration Guide. (IPY00044247)
- Updated the example code for the **gc_util_insert_parm_val( )** function in the Dialogic® Global Call API Library Reference**.**
- Added an update to indicate that Bearer Independent Call Control (BICC) signaling protocol is supported in the Dialogic® Global Call SS7 Technology Guide.
- Added an update to the Using Dual Resilient SIU Configurations section of the Dialogic® Global Call SS7 Technology Guide.
- Removed a note from the Variances for SS7 section of the Dialogic® Global Call SS7 Technology Guide.

Document Rev 42, published June 5, 2008

Updates for Service Update 202.

In the Post Release Developments chapter:

- Added Support for Ingress 183 Session Progress Informational Response Containing SDP.
- Added Support for DM_PORT_CONNECT_INFO Data Structure.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00043149, IPY00043209, IPY00043543, IPY00043595.

In the Documentation Updates chapter:

- Corrected the procedure for preserving data in user configuration files in the Dialogic® Host Media Processing Configuration Guide.
- Added a link to FLEXnet documentation in the Dialogic® Host Media Processing Administration Guide.

Document Rev 41, published May 29, 2008

Updates for Service Update 201.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00043210, IPY00043320, IPY00043536, IPY00043620.

In the Documentation Updates chapter:

- A new version of the Dialogic® Host Media Processing Administration Guide is now available on the documentation bookshelf.

Document Rev 40, published May 6, 2008

Updates for Service Update 199.

In the Post Release Developments chapter:

- Added Support for NCM Library Tracing within RTF Logging.
- Added Native T.38 Hairpinning Support (added in SU 195).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00042651, IPY00042654, IPY00042671, IPY00042960, IPY00042993, IPY00043035, IPY00043043, IPY00043261.

In the Documentation Updates chapter:

- Added a procedure for preserving data in user configuration files in the Dialogic® Host Media Processing Configuration Guide.
- Documentation corrections to the **ec_getblkinfo( )** reference page in the Dialogic® CSP API Library Reference. (IPY00043040)
- Corrected a typographical error found in Section 4.2.4. in the Dialogic® Global Call IP Technology Guide.
- Updated Table 1. “High-Level Feature Support by Platform” in the Dialogic® MSML Media Server Software User’s Guide.

Document Rev 39, published April 9, 2008

Updates for Service Update 195.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00042235, IPY00042805, IPY00042893.

Document Rev 38, published April 2, 2008

Updates for Service Update 192.

In the Post Release Developments chapter:

- Added AMD Support for PSTN.
- Added Channel Density Upgrade (G.711, Voice & Conferencing Only).
- Documentation correction for AMD Machine Support for IP-only Configurations. Replaced the dual socket dual Core Athlon 8220 2.8GHz with the dual socket dual core Opteron 8220 2.8GHz. (IPY00042925)

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00041686, IPY00041789, IPY00041876, IPY00041894, IPY00042204, IPY00042246, IPY00042300, IPY00042329, IPY00042407, IPY00042424, IPY00042461, IPY00042550, IPY00042737.
- Removed IPY00040101 as a Known Issue.

Document Rev 37, published March 31, 2008

Additional Updates for Service Update 182.

In the Post Release Developments chapter:

- Added Support for Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards.
- Added the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards to the list of supported boards in Section 1.119, “Enabling Echo Cancellation with Non-Linear Processing”, on page 219.

Document Rev 36, published March 13, 2008

Updates for Service Update 182.

In the Post Release Developments chapter:

- Added Service Update numbers to each feature description to identify the release where the feature was introduced.
- Added Support for 720 Channels.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00040358, IPY00041014, IPY00041047, IPY00042011, IPY00042072, IPY00042397.

In the Documentation Updates chapter:

- Update to Section 4.15.5. “Sending NOTIFY Requests” on the Dialogic® Global Call IP Technology Guide. (IPY00042101)

Document Rev 35, published February 6, 2008

Updates for Service Update 176.

In the Post Release Developments chapter:

- Cisco Call Manager 6.0 Support.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00041847, IPY00042016.

In the Documentation Updates chapter:

- Update to Table 2. in the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide to show support for Cisco Call Manager 6.0.
- Update the code examples in Section 4.13.4, “Responding to an INFO Message” in the Dialogic® Global Call IP Technology Guide. (IPY00041957)
- A new version of the Dialogic® Global Call SS7 Technology Guide is now available on the documentation bookshelf.
- A new version of the Dialogic® Voice API Library Reference is now available on the documentation bookshelf.

Document Rev 34, published January 22, 2008

Updates for Service Update 175.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00041112, IPY00041160, IPY00041570, IPY00041674.

Document Rev 33, published January 16, 2008

Updates for Service Update 174.

In the Post Release Developments chapter:

- Signaling Licenses Increase to 5000
- Configuring SIP Stack Parameters with Global Call

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00041294, IPY00041300, IPY00041404, IPY00041405, IPY00041582, IPY00041801.

Document Rev 32, published January 9, 2008

Updates for Service Update 172.

In the Post Release Developments chapter:

- Added support for Specifying Reliable or Non-reliable SIP.
- Updated the code example used to set or get parameters using the IP Media API for the Receive-only RFC 2833 Mode Available feature.

In the Release Issues chapter:

- Added the following Resolved Defect: IPY00040866.

In the Documentation Updates chapter:

- A new version of the Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide is now available on the documentation bookshelf.

Document Rev 31, published January 3, 2008

Updates for Service Update 171.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00041157, IPY00041254.

Document Rev 30, published December 20, 2007

Updates for Service Update 170.

In the Post Release Developments chapter:

- Added AMD Machine Support for IP-only Configurations.
- Receive-only RFC 2833 Mode Available.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00040029, IPY00041076, IPY00041268.

In the Documentation Updates chapter:

- Update to Chapter 8, “CONFIG File Parameter Reference for HMP Software” in the Dialogic® Host Media Processing Configuration Guide.
- Updated the paragraph about “references to different technologies” in the Dialogic® Global Call API Library Reference.
- Update to Chapter 3, “IP Call Scenarios” in the Dialogic® Global Call IP Technology Guide. (IPY00040320)
- A new version of the Dialogic® MSML Media Server Software User’s Guide is now available on the documentation bookshelf.

Document Rev 29, published November 28, 2007

Updates for Service Update 165.

In the Post Release Developments chapter:

- Added the Selective Packet Filtration Method.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00040630, IPY00040826, IPY00040859.
- Added the following Known Issue: IPY00036934.

In the Documentation Updates chapter:

- Added a new IP Media parameter to Chapter 8, “CONFIG File Parameter Reference for HMP Software,” in the Dialogic® Host Media Processing Configuration Guide.

Document Rev 28, published November 14, 2007

Updates for Service Update 162.

In the Post Release Developments chapter:

- Using Global Call with SS7: Updated the Capacity of SIUs table. Removed references to Dialogic® products SIU131, SIU231, and SIU520 which are no longer supported.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00039536, IPY00039823, IPY00040419, IPY00040440, IPY00040895, IPY00040961, IPY00041234.

In the Documentation Updates chapter:

- Removed all but one of the documentation updates to the Dialogic® Audio Conferencing API Library Reference. A new version of the document is now available on the documentation bookshelf.
- Removed the documentation updates to the Section 3.1.1, “Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide”, on page 357. A new version of the Release Guide is now available on the documentation bookshelf.

Document Rev 27, published October 16, 2007

Update for Service Update 157.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00040031, IPY00040071, IPY00040082, IPY00040382.

Document Rev 26, published October 5, 2007

Update for Service Update 155.

In the Post Release Developments chapter:

- Dialogic® DNI601TEPHMP boardAdded Native Streaming Support for Audio and Video.
- Added additional License Verification.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00039325, IPY00039707, IPY00039822, IPY00039964, IPY00039965.
- Added the following Known Issue: IPY00040101.

In the Documentation Updates chapter:

- Added an update to Section 2.2. “Basic Hardware Requirements” of the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide.
- Added a note to Section 3.1. “Requirements” of the Dialogic® Host Media Processing Administration Guide.

Document Rev 25, published September 25, 2007

Update for Service Update 153.

In the Release Issues chapter:

- Added the following Known Defects to the Issues Table: IPY00037336, IPY00039308, IPY00039701, IPY00039855.,

Document Rev 24, published September 6, 2007

Update for Service Update 152.

In the Documentation Updates chapter:

- Added an update to the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide about nonpaged Kernel Memory.

Document Rev 23, published August 21, 2007

Update for Service Update 150.

In the Release Issues chapter:

- Added the following Defects to the Issues Table: IPY00037841, IPY00038998, IPY00039488.

Document Rev 22, published August 16, 2007

Additional Update for Service Update 144.

In the Post Release Developments chapter:

- Added information about Enabling Echo Cancellation with Non-Linear Processing.
- Added Support for Dialogic® DNI2410TEPEHMP Digital Network Interface Board.

In the Release Issues chapter:

- Added the following Known Defects to the Issues Table: IPY00033243, IPY00038972, IPY00039309.

Document Rev 21, published July 31, 2007

Update for Service Update 144.

In the Post Release Developments chapter:

- Added information about Switching Connections from Full to Half Duplex and from Half to Full Duplex.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00039232, IPY00039321.

Document Rev 20, published July 23, 2007

Update for Service Update 142.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00038802, IPY00039033, IPY00039036, IPY00039170, IPY00039206.

Document Rev 19, published July 2, 2007

Update for Service Update 137.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00037654, IPY00038827, IPY00038848, IPY00038856, IPY00038868, IPY00038869, IPY00038992, IPY00038997.

Document Rev 18, published June 20, 2007

Update for Service Update 134.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00038288, IPY00038696, IPY00038732, IPY00038747, IPY00038208 (resolved in Service Update 130).

Document Rev 17, published June 12, 2007

Update for Service Update 133.

In the Post Release Developments chapter:

- Added information about how to Configure Persistent SIP Header Operations.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00038218, IPY00038313, IPY00038474, IPY00038475, IPY00038513, IPY00038549, IPY00038555, IPY00038572, IPY00038580, IPY00038610, IPY00038638.

Document Rev 16, published June 6, 2007

Update for Service Update 130.

In the Post Release Developments chapter:

- Added SIGTRAN Signaling Support for Global Call SS7.
- Added New Processor Support.

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00037789, IPY00038470, IPY00038476, IPY00038240, IPY00038342, IPY00038343, IPY00038365.

In the Documentation Updates chapter:

- Added updates to the Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide for the DHCP Windows Client Service.
- Added update to the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide to show support for Windows 2003 Server R2 with Service Pack 2.

Document Rev 15, published May 17, 2007

Update for Service Update 128.

In the Post Release Developments chapter:

- Added SIP PRACK Handling Support (RFC 3262).

In the Release Issues chapter:

- Added the following Resolved Defects: IPY00033127, IPY00037659, IPY00037756, IPY00037918, IPY00038049, IPY00038050, IPY00038060, IPY00038092, IPY00038150, IPY00038217, IPY00038233.
- Changed “Issue Type” to Known for IPY00037739.

In the Documentation Updates chapter:

- Added updates to the Dialogic® Global Call API Library Reference, to support SIP PRACK Handling.
- Added updates to the Dialogic® Global Call IP Technology Guide, to support SIP PRACK Handling.

Document Rev 14, published April 24, 2007

Update for Service Update 123.

In the Post Release Developments chapter:

- Added Aastra Telecom IP Phone Model 480i Support.
- Added support for the SIP Session Timer.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00034254, IPY00035675, IPY00036779, IPY00037358, IPY00037422, IPY00037699, IPY00037704, IPY00037733, IPY00037737, IPY00037776, IPY00037777, IPY00037778, IPY00037795, IPY00037888.
- Added the following Known Defect to the Issues Table: IPY00038049.

In the Documentation Updates chapter:

- Updated the **ipm_GetCapabilities( )** code example in the Dialogic® IP Media Library API Programming Guide and Library Reference to correct errors (IPY00038085).

Document Rev 13, published April 5, 2007

Update for Service Update 117.

In the Post Release Developments chapter:

- Added Command Line Interface to the its_sysinfo Utility.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00036634, IPY00037632, IPY00037686, IPY00037708, IPY00037739, IPY00037746.

In the Documentation Updates chapter:

- Updated the Troubleshooting chapter in the Dialogic® Host Media Processing Administration Guide to address errors received when using HMP with RealSpeak 4.0 (IPY00035708).
- Added information about binary log files in the Dialogic® Host Media Processing Diagnostics Guide (IPY00037518).

Document Rev 12, published March 28, 2007

Update for Service Update 115.

In the Post Release Developments chapter:

- Added SIP re-INVITE Support for MSML (available in Service Update 112).
- Added HTTP Play and Record Support for MSML (available in Service Update 112).
- Added Specify SIP Informational Response on a Per Call Level (available in Service Update 103).

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00037432, IPY00037532, IPY00037541, IPY00037603, IPY00037633, IPY00037655, IPY00037656, and IPY00036658 (resolved in Service Update 112).

In the Documentation Updates chapter:

- Added updates to the Dialogic® MSML Media Server Software User's Guide, to support SIP re-INVITE and HTTP Play and Record for MSML.
- Added updates to the Dialogic® Global Call IP Technology Guide, for SIP Informational Response on a per call level.

Document Rev 11, published March 15, 2007

Update for Service Update 112.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00037222, IPY00037338, IPY00037481, IPY00037618.

In the Documentation Updates chapter:

- Added updates to the Dialogic® Native Configuration Manager API Library Reference.

Document Rev 10, published March 7, 2007

Update for Service Update 110.

In the Post Release Developments chapter:

- Added Access the Transport Layer IP Address of an Inbound SIP Call.
- Added Access ALERTING Progress Indicator Information Element.
- Added Support for SS7 Functionality.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00033753, IPY00036292, IPY00036867, IPY00036930, IPY00036943, IPY00037170, IPY00037225, IPY00037361, IPY00037386.

In the Documentation Updates chapter:

- Added updates to the Dialogic® Global Call IP Technology Guide, to support access to the Transport Layer IP address of an inbound SIP call.
- Added updates to the Dialogic® Global Call IP Technology Guide, to support access to PIIE in an ALERTING message.
- Added a new book to the Dialogic® HMP 3.0 bookshelf, Programming Libraries Documentation Updates, called the Dialogic® Global Call SS7 Technology Guide. Also added updates to the book to support SS7 functionality.
- Added updates to Dialogic® IP Media Library API Programming Guide and Library Reference, Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide, Dialogic® Host Media Processing Configuration Guide, and Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide to support SS7 functionality.

Document Rev 09, published February 22, 2007

Update for Service Update 103.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00035405, IPY00035685, IPY00035767, IPY00035821, IPY00036086, IPY00036209, IPY00036515, IPY00036618, IPY00036790, IPY00036914, IPY00037181.

Document Rev 08, published February 8, 2007

Update for Service Update 99.

In the Post Release Developments chapter:

- Added Enhancements to CNF and DCB Notification Tone.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00034315, IPY00035374, IPY00036075, IPY00036283, IPY00036332.

In the Documentation Updates chapter:

- Added information for MOML Feature Support Matrix to Section 3.5.1, "Dialogic® MSML Media Server Software User's Guide", on page 384.
- Added new parameters to Section 3.2.2, "Dialogic® Host Media Processing Configuration Guide", on page 360.
- Added new functions to Section 3.4.2, "Dialogic® Audio Conferencing API Library Reference", on page 366.
- Added new parameter to Section 3.4.4, "Dialogic® Conferencing API Library Reference", on page 368.

Document Rev 07, published January 29, 2007

Update for Service Update 97.

In the Post Release Developments chapter:

- Added dx_reciottdata( ) Enhancements and Support for SCR.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00036236, IPY00036312, IPY00036451.

In the Documentation Updates chapter:

- Added updates to Dialogic® CSP API Programming Guide and Dialogic® CSP API Library Reference because **ec_getxmitslot( )** and **ec_rearm( )** are not supported.
- Added a new version of the Dialogic® Global Call CDP Configuration Guide to the bookshelf.
- Added updates to CONFIG file to Section 3.2.2, "Dialogic® Host Media Processing Configuration Guide", on page 360.
- Added updates to APIs in the Section 3.4.23, "Dialogic® Voice API Library Reference", on page 384.

Document Rev 06, published January 4, 2007

Update for Service Update 91.

In the Release Issues chapter:

- Added the following Resolved Defect to the Issues Table: IPY00035843.

In the Documentation Updates chapter:

- Added IPY00036321 under Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide.

Document Rev 05, published December 19, 2006

Update for Service Update 90.

In the Post Release Developments chapter:

- Added Support for Advanced Network Services.
- Added Set the MIME Message in SIP 200 OK Response.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00035507, IPY00035795, IPY00035822.

In the Documentation Updates chapter:

- Added new **cnf_ResetDevices( )** function to Section 3.4.4, "Dialogic® Conferencing API Library Reference", on page 368 to address defect IPY00035507.
- Added IPY00036235 under Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide.
- Added information for MIME message in SIP 200 OK response to Dialogic® Global Call IP Technology Guide.

Document Rev 04, published November 27, 2006

Update for Service Update 87.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00034679, IPY00035760.

Document Rev 03, published November 17, 2006

Update for Service Update 85.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00035371, IPY00035584, IPY00035589. Changed the following resolved defect IPY00034049 - resolved in Service Update 85 not Service Update 81.

In the Documentation Updates chapter:

- Added update to section 12.2, "Generating Encryption Keys" to Section 3.4.17, “Dialogic® IP Media Library API Programming Guide and Library Reference”, on page 382.
- Added update to chapter 2, Function Information, to Section 3.4.19, “Dialogic® Multimedia API Programming Guide and Library Reference”, on page 383.
- Added update to chapter 5, “Data Structure Reference” to Section 3.4.19, “Dialogic® Multimedia API Programming Guide and Library Reference”, on page 383.

Document Rev 02, published November 2, 2006

Update for Service Update 81.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00034049, IPY00034435, IPY00035489.
- In addition, the following known problems have been resolved: IPY00033894, IPY00034064.

In the Documentation Updates chapter:

- Added new **cnf_OpenEx( )** function to Section 3.4.4, “Dialogic® Conferencing API Library Reference”, on page 368 to address defect IPY00035489.

Document Rev 01, published September 2006

Initial Version of document.

Post Release Developments 1

This section describes significant changes to the Dialogic® PowerMedia™ HMP for Windows Release 3.0 subsequent to the general availability release.

1.0 Service Update

This Service Update for Dialogic® PowerMedia™ HMP for Windows Release 3.0 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.

Controlled Introduction Features

In addition to general availability of new features and functionality, Dialogic® PowerMedia™ HMP for Windows Release 3.0 Service Update releases also include functionality in a controlled introduction (CI). These are features that are under development or have a limited scope before being made generally available. These features are available for approved customers that are looking to perform Proof of Concept (PoC) with the listed functionality. CI features have not completed Dialogic's Quality Assurance ("QA") testing and are not recommended for production deployments without approval from Dialogic. Customers interested in these features should contact their Dialogic Sales Representative or Technical Support Service Engineer for further information on usage.

1.1 Microsoft Hyper-V Support

With Service Update 525, Dialogic® PowerMedia™ HMP for Windows Release 3.0 adds support for Microsoft Hyper-V Server 2019.

1.2 Microsoft Azure Support

With Service Update 525, Dialogic® PowerMedia™ HMP for Windows Release 3.0 adds support for Microsoft Azure instances running Microsoft Windows Server 2012 R2 or later.

Note: Depending on use case, an application may need to configure SIP headers ("Contact", "From") using the external IP address of the instance.

1.3 Amazon Web Services (AWS) Support

With Service Update 525, Dialogic® PowerMedia™ HMP for Windows Release 3.0 adds support for Amazon Web Services (AWS) instances running Microsoft Windows Server 2012 R2 or later. Recommend c4.2xlarge or larger instances depending on HMP and application performance requirements.

Note: Depending on use case, an application may need to configure SIP headers ("Contact", "From") using the external IP address of the instance. See https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html for steps to determine the external IP address for an instance.

1.4 Enhanced Voice Services (EVS) Codec Support

With Service Update 520, Dialogic® PowerMedia™ HMP for Windows Release 3.0 adds support for the EVS codec, the next-generation audio codec defined by the 3rd Generation Partnership Project (3GPP) for Enhanced Voice Services (EVS) in LTE networks*.

*The EVS codec is specified by the 3GPP documents TS 26.441 through TS 26.451, along with IMS Multimedia telephony media handling and interaction for EVS defined in TS 26.445 and TS 26.114.

The EVS codec is a conversational codec that offers up to 20kHz audio bandwidth that can be utilized for Narrowband (NB), Wideband (WB), Super-wideband (SWB) and Full-Band (FB) voice frequency ranges. It is optimized to provide high quality and efficiency across a wide range of voice services for telephony, conferencing and audio-visual application scenarios. In 3GPP IMS networks, the EVS codec is the successor of the current mobile HD Voice codec, AMR-WB, and includes an enhanced AMR-WB Inter-Operational mode (AMR-WB IO mode) compatible with the AMR-WB codec over all nine operational bitrates. In this AMR-WB IO mode, the EVS codec can be used as an alternative implementation of the AMR-WB codec.

EVS codec support includes enabling support for call establishment using the EVS codec and providing MRF functionality to those endpoints. The EVS codec has been integrated in HMP for RTP streaming, announcement play, call record, HD Voice conferencing and audio transcoding.

Highlights of the EVS codec implementation include:

- Compliant EVS implementation of 3GPP specifications (TS 26.441-26.451).
- Compliance to 3GPP specifications (TS 26.445 and TS 26.114) mandatory requirements for SDP negotiation with IMS EVS endpoints.
- Support for both EVS Primary and AMR-WB IO modes.
- Support for compact and header-full packetization formats.
- Support for negotiation of all EVS RTP bitrates (NB, WB, SWB, FB).
- Support for NB and WB voice range for media transcoding and conferencing.
- Addition of EVS (.evs) file format container for play/record, including native EVS record.
- Support for transcoding between EVS and other system supported codecs and file formats.

1.4.1 ipm_SetAttr( )

The **ipm_SetAttr( )** function is used to configure an IPM device for EVS.

Attribute List for EVS Codec

Name	Value	Reference
payload-type	96 to 127	3GP TS 26.445, 3GPP TS 26.114
rtp-clock-frequency	16000 (default)	HMP limit
sample-rate	16000 (default)	HMP limit
ptime	20, 40, 60, 80	3GP TS 26.445, 3GPP TS 26.114, RFC 4566
maxptime	20 to 240	3GP TS 26.445, 3GPP TS 26.114, RFC 4566
dtx	0 or 1	3GP TS 26.445, 3GPP TS 26.114
dtx-recv	0 or 1	3GP TS 26.445, 3GPP TS 26.114
hf-only	0 or 1	3GP TS 26.445, 3GPP TS 26.114
evs-mode-switch	0 or 1	3GP TS 26.445, 3GPP TS 26.114
cmr	-1, 0, 1	3GP TS 26.445, 3GPP TS 26.114
max-red	0 to 65535	3GP TS 26.445, 3GPP TS 26.114, RFC 4867

Name	Value	Reference
br	5900, 7200, 8000, 9600, 13200, 16400, 24400, 32000, 48000, 64000, 96000, 128000	3GP TS 26.445, 3GPP TS 26.114
br-send	5900, 7200, 8000, 9600, 13200, 16400, 24400, 32000, 48000, 64000, 96000, 128000	3GP TS 26.445, 3GPP TS 26.114
br-recv	5900, 7200, 8000, 9600, 13200, 16400, 24400, 32000, 48000, 64000, 96000, 128000	3GP TS 26.445, 3GPP TS 26.114
bw	nb, wb, swb, fb, nb-wb, nb-swb, nb-fb	3GP TS 26.445, 3GPP TS 26.114
bw-send	nb, wb, swb, fb, nb-wb, nb-swb, nb-fb	3GP TS 26.445, 3GPP TS 26.114
bw-recv	nb, wb, swb, fb, nb-wb, nb-swb, nb-fb	3GP TS 26.445, 3GPP TS 26.114
ch-send	1 or 2	3GP TS 26.445, 3GPP TS 26.114
ch-recv	1 or 2	3GP TS 26.445, 3GPP TS 26.114
ch-aw-recv	-1, 0, 2, 3, 5, 7	3GP TS 26.445, 3GPP TS 26.114
mode-set	0 to 7	3GP TS 26.445, 3GPP TS 26.114
mode-change-period	1 or 2	3GP TS 26.445, 3GPP TS 26.114, RFC 4867
mode-change-capability	1 or 2	3GP TS 26.445, 3GPP TS 26.114, RFC 4867
mode-change-neighbor	0 or 1	3GP TS 26.445, 3GPP TS 26.114, RFC 4867
native	0 or 1 (default: 0)	HMP specific

The **ipm_StartMedia( )** function is used to configure an IPM device to transmit and receive EVS RTP streams. When setting up the necessary parameters for the IPM_MEDIA structure, set only the following:

- The eIPM_MEDIA_TYPE value to one of the audio based enumerated datatypes: MEDIATYPE_(LOCAL/REMOTE)_AUDIO_(RTP/RTCP)_INFO.
- The port information in the IPM_PORT_INFO structure that is also referenced in the IPM_MEDIA structure.

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

Note: Setting of all the codec related information is done through the Attribute List for EVS Codec table when building the attribute list via the **ipm_SetAttr( )** API call. In this case, it is not necessary to set the actual IPM_AUDIO_CODER_INFO as with other codecs.

mm_record ( ) / mm_play ( )

```
typedef enum
{
     MM_EVS_BW_NOT_PRESENT
   , MM_EVS_BW_NB
   , MM_EVS_BW_WB
   , MM_EVS_BW_SWB
   , MM_EVS_BW_FB
   , MM_EVS_BW_NB_WB
   , MM_EVS_BW_NB_SWB
   , MM_EVS_BW_NB_FB
} eMM_EVS_BW;typedef enum
{
     MM_CODER_KEYS_EVS_DTX  = 0
   , MM_CODER_KEYS_EVS_DTX_RECV
   , MM_CODER_KEYS_EVS_HF_ONLY
   , MM_CODER_KEYS_EVS_MODE_SWITCH
   , MM_CODER_KEYS_EVS_SAMPLERATE
   , MM_CODER_KEYS_EVS_PTIME
   , MM_CODER_KEYS_EVS_MAX_PTIME
   , MM_CODER_KEYS_EVS_CMR
   , MM_CODER_KEYS_EVS_MAX_RED
   , MM_CODER_KEYS_EVS_BR1
   , MM_CODER_KEYS_EVS_BR2
   , MM_CODER_KEYS_EVS_BR1_SEND
   , MM_CODER_KEYS_EVS_BR2_SEND
   , MM_CODER_KEYS_EVS_BR1_RECV
   , MM_CODER_KEYS_EVS_BR2_RECV
   , MM_CODER_KEYS_EVS_BW
   , MM_CODER_KEYS_EVS_BW_SEND
   , MM_CODER_KEYS_EVS_BW_RECV
   , MM_CODER_KEYS_EVS_CH_SEND
   , MM_CODER_KEYS_EVS_CH_RECV
   , MM_CODER_KEYS_EVS_CH_AW_RECV
   , MM_CODER_KEYS_EVS_MODE_SET
   , MM_CODER_KEYS_EVS_MODE_CHANGE_PERIOD
   , MM_CODER_KEYS_EVS_MODE_CHANGE_CAPABILITY
   , MM_CODER_KEYS_EVS_MODE_CHANGE_NEIGHBOR
} eMM_CODER_KEYS_EVS;
```

Sample Code

The following is sample code to show how to setup an EVS connection and to play/record to .evs container:

```
void setEVSprofile(int idx) {
        switch (PROFILE) {
        //payload-type=97, sample-rate=16000, ptime=20, maxptime=240, dtx=0, hf-only=0,
evs-mode-switch=0, cmr=0, max-red=220, br=5.9-24.4, bw=nb-wb, native=0
               case 1:
                       if (ipm_SetAttr(port[idx].ipmh,"rtp-1","codec-type","evs"))
                               { printf("Error in ipm_SetAttr(codec-type):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                       }
                       if (ipm_SetAttr(port[idx].ipmh,"rtp-1","direction","receive"))
                               { printf("Error in ipm_SetAttr(direction):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                       }
                       if (ipm_SetAttr(port[idx].ipmh,"rtp-1","payload-type","97"))
                               { printf("Error in ipm_SetAttr(payload-type):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                       }
                       if (ipm_SetAttr(port[idx].ipmh,"rtp-1","sample-rate","16000")) {
```

```
                                printf("Error in ipm_SetAttr():
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","ptime","20")) {
                                printf("Error in ipm_SetAttr(ptime):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","maxptime","240"))
                                { printf("Error in ipm_SetAttr(maxptime):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","dtx","0")) {
                                printf("Error in ipm_SetAttr(dtx):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","hf-only","0")) {
                                printf("Error in ipm_SetAttr(hf-only):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","evs-mode-switch","0"))
                                { printf("Error in ipm_SetAttr(evs-mode-switch):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","max-red","220"))
                                { printf("Error in ipm_SetAttr(max-red):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","br","5.9-24.4"))
                                { printf("Error in ipm_SetAttr(br):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","bw","nb-wb")) {
                                printf("Error in ipm_SetAttr(bw):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","native","0")) {
                                printf("Error in ipm_SetAttr(native):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-1","cmr","0")) {
                                printf("Error in ipm_SetAttr(cmr):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","codec-type","evs"))
                                 { printf("Error in ipm_SetAttr(codec-type):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","direction","transmit"))
                                 { printf("Error in ipm_SetAttr(direction):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","payload-type","97"))
                                 { printf("Error in ipm_SetAttr(payload-type):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","sample-rate","16000"))
                                { printf("Error in ipm_SetAttr():
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","ptime","20")) {
                                printf("Error in ipm_SetAttr(ptime):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","maxptime","240"))
                                { printf("Error in ipm_SetAttr(maxptime):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
```

```
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","dtx","0")) {
                                printf("Error in ipm_SetAttr(dtx):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","hf-only","0")) {
                                printf("Error in ipm_SetAttr(hf-only):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","evs-mode-switch","0"))
                                { printf("Error in ipm_SetAttr(evs-mode-switch):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","max-red","220"))
                                { printf("Error in ipm_SetAttr(max-red):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","br","5.9-24.4"))
                                { printf("Error in ipm_SetAttr(br):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","bw","nb-wb")) {
                                printf("Error in ipm_SetAttr(bw):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","native","0")) {
                                printf("Error in ipm_SetAttr(native):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        if (ipm_SetAttr(port[idx].ipmh,"rtp-2","cmr","0")) {
                                printf("Error in ipm_SetAttr(cmr):
%s",ATDV_ERRMSGP(port[idx].ipmh));
                        }
                        break;
                default:
                        printf("** INVALID PROFILE selected: %d **\n",PROFILE);
            break;
        }
}

void play_file(char *a_fileName, int idx) {
        MM_PLAY_INFO        playInfo;
        MM_PLAY_RECORD_LIST playRecordList;
        MM_MEDIA_ITEM_LIST  audioMediaList;
        MM_AUDIO_CODEC      audioCodecPlay;
        MM_RUNTIME_CONTROL  rtcP;

        INIT_MM_AUDIO_CODEC(&audioCodecPlay);
        INIT_MM_MEDIA_AUDIO(&audioMediaList.item.audio);
        INIT_MM_MEDIA_ITEM_LIST(&audioMediaList);
        INIT_MM_PLAY_RECORD_LIST(&playRecordList);
        INIT_MM_PLAY_INFO(&playInfo);
        INIT_MM_RUNTIME_CONTROL(&rtcP);

        rtcP.Reason = EMM_TERM_NORTC;
        rtcP.Action = EMM_TA_AUDIO_STOP;
        rtcP.unValue = 1;

        switch (FORMAT) {
               case 5: //EVS file
                        audioCodecPlay.unCoding = MM_DATA_FORMAT_EVS;
                        audioCodecPlay.unSampleRate = 0;
                        audioCodecPlay.unBitsPerSample = 0;
        //payload-type=97, sample-rate=16000, ptime=20, maxptime=240, dtx=0, hf-only=0,
evs-mode-switch=0, cmr=0, max-red=220, br=5.9-24.4, bw=nb-wb, native=0
                        audioCodecPlay.unCount = 9;
                        audioCodecPlay.KeyValues[0].unKey = MM_CODER_KEYS_EVS_SAMPLERATE;
                        audioCodecPlay.KeyValues[0].unValue = 16000;
```

```
                        audioCodecPlay.KeyValues[1].unKey = MM_CODER_KEYS_EVS_PTIME;
                        audioCodecPlay.KeyValues[1].unValue = 20;
                        audioCodecPlay.KeyValues[2].unKey = MM_CODER_KEYS_EVS_MAX_PTIME;
                        audioCodecPlay.KeyValues[2].unValue = 240;
                        audioCodecPlay.KeyValues[3].unKey = MM_CODER_KEYS_EVS_DTX;
                        audioCodecPlay.KeyValues[3].unValue = 0;
                        audioCodecPlay.KeyValues[4].unKey = MM_CODER_KEYS_EVS_MODE_SWITCH;
                        audioCodecPlay.KeyValues[4].unValue = 0;
                        audioCodecPlay.KeyValues[5].unKey = MM_CODER_KEYS_EVS_MAX_RED;
                        audioCodecPlay.KeyValues[5].unValue = 220;
                        audioCodecPlay.KeyValues[6].unKey = MM_CODER_KEYS_EVS_BR1;
                        audioCodecPlay.KeyValues[6].unValue = 24400;
                        audioCodecPlay.KeyValues[7].unKey = MM_CODER_KEYS_EVS_BW;
                        audioCodecPlay.KeyValues[7].unValue = MM_EVS_BW_NB_WB;
                        audioCodecPlay.KeyValues[8].unKey = MM_CODER_KEYS_EVS_CMR;
                        audioCodecPlay.KeyValues[8].unValue = 0;

                        audioMediaList.item.audio.eFileFormat =
                        EMM_FILE_FORMAT_EVS; break;

                }

                audioMediaList.item.audio.codec = audioCodecPlay;
                audioMediaList.item.audio.unMode = 0;
                audioMediaList.item.audio.unOffset = 0;
                audioMediaList.item.audio.szFileName = a_fileName;
                audioMediaList.item.audio.unAccessMode = MM_MEDIA_ACCESS_MODE_FILE;

                playRecordList.ItemType = EMM_MEDIA_TYPE_AUDIO;
                playRecordList.ItemChain = EMM_ITEM_EOT;
                playRecordList.list = &audioMediaList;

                playInfo.list = &playRecordList;

                printf("Playing file: %s\n", audioMediaList.item.audio.szFileName);
                if (mm_Play(port[idx].mmh,&playInfo,&rtcP, (void*)&port[idx]) != EMM_SUCCESS)
                {
                        GetMMErrorInfo("mm_Play()",port[idx].mmh);
                }
}
```

1.4 Controlled Introduction (CI) Support for OpenStack/KVM Environments

New clocking changes have been incorporated into Service Update 520 which will allow Dialogic® PowerMedia™ HMP for Windows Release 3.0 to start and run in cloud environments. With Service Update 520, Dialogic® PowerMedia™ HMP for Windows Release 3.0 will now use a derived system clock when the recommended High Precision Event Timer (HPET) is not available for HMP clocking. Additional architectural changes are planned for upcoming Dialogic® PowerMedia™ HMP for Windows Release 3.0 Service Update releases that will improve audio and tone detection/generation performance in these environments. Refer to the Controlled Introduction Features section for more details on controlled introduction (CI).

1.5 End of Support Notification for Various Windows Operating System Versions

With Service Update 520, Dialogic® PowerMedia™ HMP for Windows Release 3.0 drops support for support Windows 7 (both 32-bit and 64-bit versions), Windows Server 2008 R1 (32-bit version only), and Windows 10 (32-bit version only). Note that 32-bit HMP applications are still supported in this release. See PCN D000311-00 for further details.

1.6 End of Support for Dialogic® D/4PCIU Boards

With Service Update 520, Dialogic® PowerMedia™ HMP for Windows Release 3.0 removes support for Dialogic® D/4PCIUFEQ and Dialogic® D/4PCIU4SEQ Springware boards. See PCN D000304-00 for further details.

1.7 Support for Windows Server 2019 Operating System

Service Update 393 introduces support for Windows 2019 operating system. For information about installing and using the Dialogic® PowerMedia™ HMP for Windows Release 3.0, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide*.

1.8 Installation Package Policy

Dialogic® PowerMedia™ HMP for Windows Release 3.0 is an InstallShield-based installation package delivered as a zipped .zip file for installation on an existing Microsoft Windows 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.

1.9 Updated Visual C++ Libraries Runtime Components to Visual C++ 2015 Redistributable

With Service Update 382 on Dialogic® PowerMedia™ HMP for Windows Release 3.0, Visual C++ libraries runtime components have been upgraded to latest Visual C++ 2015 Redistributable. This removes the dependencies on outdated versions of Microsoft C Runtime (MSVCx80 runtime) from HMP.

In addition, the HMP installation setup has been updated to detect the presence of the appropriate Visual C++ 2015 Redistributable runtime (either 32-bit and/or 64-bit) version, and if required, prompt for installing the Visual C++ 2015 Redistributable update.

The updated Visual C++ libraries runtime provides the proper dynamic linking to Dialogic software components; it does not affect existing customer applications which will run unmodified on Service Update 382 in spite of the version of the Visual Studio it was originally built with, all other things (runtime libraries) being equal.

Note: Due to the fact the HMP installation setup detection and prompting of the Visual C++ libraries runtime redistributable is not part of the silent install, customers can choose to install it themselves prior to running the setup in silent install mode. The latest update can be obtained from: http://www.microsoft.com/en-us/download/details.aspx?id=52685.
For 32-bit applications, the *vc_redist.x86.exe* should be installed.
For 64-bit applications, the *vc_redist.x64.exe* should also be installed, in addition to the former.

Refer to the "Performing a Silent Install" section in the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide* for further information on silent install.

1.10 Additional 64-bit Runtime API Support

Service Update 382 includes additional 64-bit API support for HMP applications.

This release provides 64-bit API support for the following HMP host libraries:

- **Global Call API Library** – gc_

  Note: Limited to Global Call IP.
- **Device Management API Library** – dev_
- **Multimedia API Library** – mm_
- **Voice API Library** – dx_
- **Continuous Speech Processing API Library** – csp_
- **Conferencing API Library** – cnf_
- **Audio Conferencing API Library** – dcb_
- **IP Media API Library** – ipml_
- **Standard Runtime Library API Library** – sr_

Applications must link with the newly-provided Dialogic x64 export libraries; they are named with the "64" suffix and in the same order as the above list:

- libgc64.lib
- libdevmgmt64.lib
- libmml64.lib
- libdxxmt64.lib
- libecmt64.lib
- libcnf64.lib
- libdm3dcb64.lib
- libipm64.lib
- libsrlmt64.lib

Alternatively, applications can link dynamically (**LoadLibrary( )** function) with their DLL counterparts. Refer to the 64-bit Runtime API Support section for more details on 64-bit Dynamic Linking.

1.11 64-bit Runtime API Support

Overview

Service Update 375 contains the complete Dialogic® PowerMedia™ HMP for Windows Release 3.0 installation for 32-bit applications. Existing 32-bit applications will continue to operate unmodified with this release.

In addition, Service Update 375 includes new import libraries and DLLs allowing 64-bit applications to directly link to HMP libraries on a 64-bit environment. The 64-bit DLLs/LIBs have a different name to their 32-bit counterpart. The convention used is to add "64" to the root of the filename. For example, the 32-bit import library *libsrlmt.lib* would become *libsrlmt64.lib*; the 32-bit library *libdxxmt.dll* would become *libdxxmt64.dll*.

This allows easy mixing of 32-bit and 64-bit DLLs in the same 64-bit system, so at runtime the system will automatically load the appropriate DLL that corresponds to each import library.

Header files are common to both platforms.

For 64-bit applications that use dynamic linking, via **GetProcAddress( )** or similar method, the Dialogic DLL name must reflect this convention in order to load the appropriate dynamic library that matches the application target.

This release provides 64-bit API support for the following HMP host libraries:

- **Voice API Library** – dx_
- **Continuous Speech Processing API Library** – csp_
- **Conferencing API Library** – cnf_

- **IP Media API Library** – ipml_

Additional 64-bit API support will be rolled out in future Service Updates.

Installation Instructions

- Install Service Update 375.
- Download and install 64-bit Visual C++ Redistributable for Visual Studio 2015 from the following link: http://www.microsoft.com/en-us/download/details.aspx?id=48145

  Note: This is only required if the target application is 64-bit; 32-bit targeted applications use an earlier Visual C++ Redistributable installed automatically by the setup.

Compiling and Linking

Change the platform settings in Visual Studio to generate 64-bit application and update the dependencies list to include 64-bit versions of the HMP import libraries.

1.12 SIP TLS Wildcard Server Certificate Support

With Service Update 375, Dialogic® PowerMedia™ HMP for Windows Release 3.0 has added support for wildcard character "*" in the DNS name in a Transport Layer Security (TLS) Server Certificate when HMP acts as TLS client during regular TLS handshake.

Feature Description

There may be two X.509 attributes of a TLS Server certificate that are normally used for matching the Server "hostname", as follows:

- *CommonName* (**CN**)
- *SubjectAltName* (**SAN**)

As such, any or both of those attributes can contain the wildcard character "*".

This feature support is limited to a wildcard in the **CN-ID** (*CN RDN* of certificate). Only if full Server "hostname" verification fails in the **SAN** (*SubjectAltName* with *DNSName* entry in the certificate), will wildcard matching in the **CN-ID** (this feature) take effect. To further reduce the risk of vouching for rogue servers, **CN-ID** wildcard support is limited to a DNS identifier in which the wildcard character is in its left-most label.

The feature is available by default.

1.13 Support for Windows 10 and Windows Server 2016 Operating Systems

Service Update 372 introduces support for Windows 10 and Windows Server 2016 operating systems. For information about installing and using the Dialogic® PowerMedia™ HMP for Windows Release 3.0, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Installation Guide*.

Note: Secure boot should be disabled in the UEFI BIOS prior to installing Service Update 372 on new Windows 10 installations, version 1607 or later.

1.14 SIP TLS Version 1.2 Support

With Service Update 371, Dialogic® PowerMedia™ HMP for Windows Release 3.0 has added support for Transport Layer Security (TLS) version up to 1.2.

Enabling the Feature

In order to take advantage of the feature, new enumerations to the **sip_tls_method** in the **SIP_TLS_ENGINE** data structure have been added as such:

- ENUM_TLS_METHOD_TLS_V1_1 – use TLS ver. 1.1
- ENUM_TLS_METHOD_TLS_V1_2 – use TLS ver. 1.2

In order to take advantage of the feature, a TLS application simply sets the aforementioned field to one of the enumeration value for the desired TLS version. No other changes are required.

Note: The default TLS version continues being 1.0:

- ENUM_TLS_METHOD_TLS_V1 – use TLS ver. 1 (Default value)

1.15 Multimedia User I/O

With Service Update 367, Dialogic® PowerMedia™ HMP for Windows Release 3.0 has expanded support for HD audio and Multimedia User I/O API. For details, refer to the *Dialogic® Multimedia API Programming Guide and Library Reference*.

1.16 Adaptive Multi-Rate Narrowband (AMR-NB) Audio Codec for RTP

Dialogic® PowerMedia™ HMP for Windows Release 3.0 adds support for the Adaptive Multi-Rate Narrowband (AMR-NB) Audio Codec. AMR-NB frame size is 20 ms at 1 to 33 frames per packet.

Note: Voice activity detection (VAD) is disabled by firmware when frames per packet is greater than 10. This cannot be changed by the application.

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

1.17 G.722.2 Adaptive Multi-Rate Wideband (AMR-WB) Audio Codec for RTP

Service Update 367 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 to 7000 Hz instead of 200 to 3400 Hz for wideband applications. The increased bandwidth improves the intelligibility and naturalness of speech significantly.

1.18 VMware ESXi 6.x Support

VMware ESXI 6.x is supported by Dialogic® PowerMedia™ HMP for Windows Release 3.0. Refer to the Virtualization Support section for HMP configuration and other relevant information. For VMware documentation, refer to http://www.vmware.com/support/pubs.

1.19 SIP TLS Certificate Verification and Post-Connect Callback Functions

With Service Update 361, Dialogic® PowerMedia™ HMP for Windows Release 3.0 provides two callback functions that allow for application fine tuning of the SIP TLS handshake process when HMP acts as the client side or in the case of server side if client certificate is enabled. In particular, a certificate-verification callback function provides the application with the raw X.509 certificate(s), and the post-connect callback function provides connection-specific information along with X.509 certificate attributes information.

Both callback functions allow for the overriding of two default HMP SIP stack decisions during the TLS handshake: namely if a failure either in the X.509 certificate verification, or in the TLS handshake post-connect, occurs.

In that case, HMP will consult the application by invoking one or both callback functions, thus providing the application the ability to override HMP's default decision during the TLS handshaking process.

Enabling the Feature

In order to take advantage of the feature, a TLS application must declare and implement at least one of two newly introduced TLS callback functions to HMP passing them to the **IP_SIP_TLS_ENGINE** structure.

Interface Changes

The **IP_SIP_TLS_ENGINE** structure offers two specific fields (only the two feature-specific fields are listed for simplicity) as follows:

```
typedef struct
{
     …
     VerifyCertificateCallback verify_cert_CB; /* CB to a server certificate callback
     */ PostConnectCallback post_connect_CB; /* CB to a post-connect callback */
…
}SIP_TLS_ENGINE;
```

The **verify_cert_CB** field expects a callback function with typedef as follows:

```
#ifdef WIN32
typedef int (__stdcall *VerifyCertificateCallback)(const char *const, int, int
); #else
typedef int (*VerifyCertificateCallback)(const char *const, int, int
); #endif
```

The **post_connect_CB** field expects a callback function declared as follows:

```
#ifdef WIN32
typedef int (__stdcall *PostConnectCallback) (const GC_PARM_BLK
*const); #else
typedef int (*PostConnectCallback) (const GC_PARM_BLK
*const); #endif
```

The enum field, eTransportConnType, has been added to the existing **IP_SIP_TRANSPORT_ADDR** structure as follows:

```
typedef struct {
   unsigned long               version;
   unsigned short              remTAport;
   EnumSipTransport            eremTransportType;
   EnumSipTransportAddressType eremAddrType;
   char                        remTA[CCLIB_SIP_TRANSPORT_IPSTRING_LEN];
   EnumSipTransportConnType    eTransportConnType;
}IP_SIP_TRANSPORT_ADDR;
```

The eTransportConnType enumeration only applies to TCP connections; it indicates if the connection was created by the remote UA or by the HMP application. The possible enumerations are as follows:

```
typedef enum
{
   eSIP_TRANSPORT_CONN_TYPE_UNDEFINED,
   eSIP_TRANSPORT_CONN_TYPE_CLIENT,
   eSIP_TRANSPORT_CONN_TYPE_SERVER,
   eSIP_TRANSPORT_CONN_TYPE_MULTISERVER
}EnumSipTransportConnType;
```

The TLS-specific data structure, **IP_TLS_NAME_INFO**, has been introduced as follows:

```
typedef struct IP_TLS_NAME_INFO{
   unsigned long       version;
   EnumTlsCertNameAttr eNameAttr;
   int                 nameLen;
   union {
      unsigned char    *szNameUtf8;
      char             *szHostname;
   } szName_u;
} IP_TLS_NAME_INFO;
```

This structure provides X.509 certificate name information as well as TCP-connection host name, as indicated below:

- version is used by the SIP library and should not be altered by the application. Use **INIT_IP_TLS_NAME_INFO( )** to initialize its value along with the rest of the fields.
- eNameAttr is an enumeration which indicates what type of name string is passed into the embedded szName_u union (see below), if any. It can be one of these values:

```
typedef enum
{
   eSIP_TLS_CERT_ATTR_UNDEFINED = -1,       /* undefined */
   eSIP_TLS_CERT_ATTR_SUBJECT_CN,           /* Subject attr; CN subset */
   eSIP_TLS_CERT_ATTR_SUBJECT_ALT_NAME_DNS, /* SubjectAltName attr; DNS subset */
   eSIP_TLS_POST_CONNECT_HOSTNAME           /* Post-connect hostname  */
} EnumTlsCertNameAttr;
```

- **szName_u** is an embedded union; it contains two possible formats for the remote host name of the connection. It can be used in the post-connection CB to compare against the X.509 certificate attributes in order to help in whether to override the SIP stack decision, if a post-connection failure in the TLS handshake occurs:
  - **szNameUtf8** - raw (unsigned char*) string which length is indicated in the nameLen field. Note that this might not include the terminating character (\0) thus the application must always obtain its length prior to manipulating it.
  - **szHostname** - containing a C-style, null-terminated string in ASCII format.

Note: The **szHostname** might not necessarily be a name string in the strict sense; it can also be an IP address in the form of a string.

New Callback Functions

The **VerifyCertificateCallback**, if defined, and passed to HMP in the **verify_cert_CB** field, enables the application to examine incoming certificates and analyze data on these certificate(s) and override a fail decision from the HMP SIP stack on the X.509 certificate(s) verification. In this callback, the application is allowed to retrieve data; however, it's not allowed to alter the certificate in any manner.

The callback applies whenever HMP acts as the Client side for the purposes of the TLS handshake, or else if Server side and client certificate is requested (via the **E_client_cert_require** field).

Notes:1. Only if a SIP stack certificate verification occured will HMP invoke the **VerifyCertificateCallback** callback function, providing application access to the raw X.509 certificate.

2. It's possible that the callback function is invoked more than once in short order if the certificate received from the far end during TLS handshake is chained.

3. The context for the callback is global, in the sense that it will be invoked for every applicable TLS handshake, irrespective of the channels for which the handshake is taking place. Furthermore, there will be no indication as to what SIP channel the function is invoked for.

4. The arguments to the function are in the following order:

- IN: **const char *const server_cert_base64**: base64 encoded certificate. Neither the pointer nor its content can be altered by the application.
- IN: **int server_cert_len**: length of the base64-encoded certificate in bytes.
- IN: **int error**: non-zero value indicates a SIP stack failure in certificate verification.

5. Return value: The application can either stick to the SIP stack decision about the verification by returning the same error value as its function return value or override it by returning **GC_SUCCESS** (zero) from the function. In the latter case, it will instruct the SIP stack to continue with the handshaking process irrespective of its previous decision.

Example using OpenSSL Functions to Examine the X.509 Certificates:

```
#include <openssl/x509.h>
…
int serverCertVerificationCB(const char *const server_cert_base64, int server_cert_len, int
error)
{
   X509   *pCert = NULL;
   X509   *pszCert = server_cert_base64;
   char   bVerify = 0;
   char   szTmpData[1024] = {'\0'}; // 1024 might not be sufficient for all sorts of X.509
                                    // certificates. Consider using dynamic allocation instead
   // converting the certificate to Open SSL format
   pCert = d2i_X509(NULL,(const unsigned char**)&pszCert,
   server_cert_len); if (pCert != NULL) {
            X509_NAME_oneline(X509_get_subject_name(pCert),szTmpData,1024);
            // freeing the certificate
            X509_free(pCert);
            return error; // return same value as the error argument
   }
   // Perform your own certificate processing to verify it; set bVerify on if
   so X509_free(pCert);
   if (bVerify)
            return 0; // pass it!
   else
            return error; // return same value as the error argument
}
```

The **PostConnectCallback** is used to override a TLS handshake post-connection assertion of the SIP Stack. Once a connection has completed the handshake, the SIP stack needs to make sure that the certificate presented was issued for the address for which the connection was made. This is known as the post-connection assertion which is performed automatically by the stack. If the application would like to override a failed assertion, it can implement and pass the address of the function callback to HMP in the **post_connect_CB** field.

One of the parameters of the post-connection handshake is the host name of the TCP connection, against which the SIP stack will compare with the certificate CN attribute RDN of the subject field or alternative with the subjectAltName field, in order to prevent a malicious UA to impersonate a host whose X.509 server certificate has been compromised.

The application can, for instance, keep a list of predefined known host names to check against if a SIP-stack assertion occurs and thus make its own determination. With this new functionality, the application is given the option to decide whether the connection should proceed and bypass a failed post-connection assertion.

Notes:1. Only if a SIP stack post-connect TLS handshake assertion occur will HMP invoke the **PostConnectCallback** callback function.

2. The CB context is global, in the sense that it will be invoked for every applicable post-connect SIP-stack failed assertion, irrespective of the SIP channels for which the handshake is occurring. Furthermore, there will be no indication as to what channel or handshake the function is invoked for.

3. The argument to the function is a GC_PARM_BLK structure that can be examined by the application:

IN: **const GC_PARM_BLK *const pCertParmblk**: It will contain two parm_ID under the IPSET_CALLINFO set_ID:

- **IPPARM_SIP_TRANSPORT_ADDR**
- **IPPARM_SIP_TLS_NAME_INFO**

4. Return value: The application can override the SIP stack post-connect assertion status by returning **GC_SUCCESS** (zero) from the function. It can also return a non-zero value to keep the assertion and thus fail the handshake. In either case the decision will be final in that it will instruct the SIP stack to either pass or fail the TLS post-connection stage.

- Alternatively the application can return **IPERR_REMOTE_PROXY_ADDR**; in this case the HMP stack will continue processing the assertion and compare the host name of the connection against the outbound proxy name (**outbound_proxy_hostname**), and outbound proxy address (**outbound_proxy_IP**), and if a match exists with either of them, the assertion will also be overridden completing the handshaking process accordingly.

5. HMP SIP stack allocates storage for the **GC_PARM_BLK** passed in the **pCertParmblk** constant pointer function argument, the application is not allowed to manipulate its address in any way, nor should it alter its content. The storage will be deleted by the HMP SIP stack upon the CB returning control to HMP.

Example:

```
int postconnHostnameCB(const GC_PARM_BLK *const pCertParmblk)
{
    GC_PARM_DATA_EXT parm_data;
    IP_TLS_NAME_INFO *pSipTlsName;
    int len=0;
    char  bPass = 0;

    INIT_GC_PARM_DATA_EXT(&parm_data);

    while (gc_util_next_parm_ex(pCertParmblk,&parm_data) == GC_SUCCESS)
    {
        switch (parm_data.set_ID)
        {
            case IPSET_CALLINFO:
            {
               switch (parm_data.parm_ID)
               {
                   case IPPARM_SIP_TRANSPORT_ADDR:
                   {
                          if(parm_data.data_size == sizeof(IP_SIP_TRANSPORT_ADDR))
                                        { IP_SIP_TRANSPORT_ADDR *pSipTA;
                                        pSipTA = (IP_SIP_TRANSPORT_ADDR
                                        *)parm_data.pData;
                                        if (pSipTA->eremTransportType ==
                                        eSIP_TRANSPORT_TLS) {
                                        // Do something with the transport type
                                        }
                                 }
                                 }
                                 case IPPARM_SIP_TLS_NAME_INFO:
                                 {
                                        if(parm_data.data_size == sizeof(IP_TLS_NAME_INFO)) {
                                        pSipTlsName = (IP_TLS_NAME_INFO
                                        *)parm_data.pData;
                                        len = pSipTlsName->nameLen;
                                        if (pSipTlsName->eNameAttr ==
                                        eSIP_TLS_CERT_ATTR_SUBJECT_CN) {
                                        // Do something with the Subject CN
                                        }
                                        if (pSipTlsName->eNameAttr ==
                                        eSIP_TLS_CERT_ATTR_SUBJECT_ALT_NAME_DNS) {
                                        // Do something with the AltSubjectName
                                        }
                                        if (pSipTlsName->eNameAttr ==
                                        eSIP_TLS_POST_CONNECT_HOSTNAME) {
                                        // Do something with the host name of the
                                        connection
                                        }
                                 }
                                 }
                           }
                     }
               }
        }
        // Perform your own processing in order to decide to override default
        assertion; set bPass on if so
        if (bPass)
               return 0; // pass it!
        else
               return -1; // keep the assertion and fail the handshake
}
```

1.20 Support for New SIP_STACK_CFG Data Structure User-Configurable Fields

With Service Update 360, the SIP_STACK_CFG member structure which is referenced by the IP_VIRTBOARD data structure, contains two new user-configurable fields as follows:

- int forked1xxTimerTimeout;
- EnumSIP_Enabled removeOldAuth;

To take advantage of these new fields, the application must be recompiled; furthermore and irrespective of these two new fields, the **INIT_SIP_STACK_CFG( )** function is used to initialize the structure with the correct version number and initial field values before setting the appropriate values.

The forked1xxTimerTimeout field sets the timeout value for the forked-1xx-timer which is set by a forked call resource (call-leg) after receiving the first 1xx response. This timer is released when the call-leg receives a 2xx response. If the timer expires before 2xx reception, the call-leg is terminated. This timeout value defines how long the call-leg will wait for a 2xx response before termination and its units are in milliseconds. The default is 32000.

Note: The call-leg forking can occur if multiple 1xx responses arrive from more than one UA due to a proxy forking an INVITE out of HMP. Set the forked1xxTimerTimeout field to zero (“0”) to avoid call-leg forking altogether.

The removeOldAuth is an enum used for Digest authentication; it determines whether to remove old authentication challenges that were received from the same realm.

If a new challenge is received from the server (401/407) and this field is enabled, the Stack will check whether an old challenge from the same realm exists. If so, the old challenge will be removed.

Possible values are ENUM_Disabled (default), or ENUM_Enabled. By default, old authentication challenges will NOT be removed.

1.21 Windows 2003 Operating System End of Support Notification

Microsoft has announced that Windows Server 2003 Product Lifecycle Mainstream Support has ended and the Extended Support End Date has been announced for July 14, 2015.

Dialogic® PowerMedia™ HMP for Windows Release 3.0 software support for Windows Server 2003 and Windows Server 2003 R2 operating systems December 21, 2014 (“End of Software Support date”) and ending technical support July 15, 2015 (“End of Technical Service Support date”).

Although Windows Server 2003 and Windows Server 2003 R2 functionality will remain in Dialogic® PowerMedia™ HMP for Windows Release 3.0 until the End of Technical Service Support date, please note that after the End of Software Support date, no further fixes will be made for issues that relate specifically to interoperability with Windows Server 2003 or Windows Server 2003 R2.

Customers are strongly encouraged to start the migration toward newer, supported versions of the Windows operating system, such as Windows Server 2008. Consult the following link for more details:
http://support2.microsoft.com/lifecycle/search/default.aspx?sort=PN&alpha=Windows+Server+2003

1.22 VMware ESXi Virtualization Support with Windows Server 2012 R2

With Service Update 354, Dialogic® PowerMedia™ HMP for Windows Release 3.0 adds supports for VMware ESXi virtual machine with Windows Server 2012 R2. This support applies to VMware ESXi versions 4.1 through 5.5, and the following of Windows guest operating systems:

- Windows Server 2008 R2 with Service Pack 1
- Windows Server 2008 with Service Pack 2 (32-bit and 64-bit versions)
- Windows 7 with Service Pack 1 (32-bit and 64-bit versions)
- Windows Server 2003 (Standard or Enterprise Edition) with Service Pack 2 (32-bit version)
- Windows Server 2003 R2 (Standard or Enterprise Edition) with Service Pack 2 (32-bit version)
- Windows Server 2012 R2 (Support added in September 2014)

Refer to Section 1.43, “VMware ESXi Virtualization Support Enhancements”, on page 132 for more details.

1.23 Windows XP Operating System No Longer Supported

With Service Update 354, Dialogic® PowerMedia™ HMP for Windows Release 3.0 drops support for any previously-supported versions of Windows XP operating system, such as Windows XP Professional and its Service Packs.

1.24 Windows Server 2003 Operating System Restrictions

Starting with Service Update 354, certain restrictions on the supported version of the Windows Server 2003 operating system apply:

- This Service Update 354 reduces IP RTP media channel density to four hundred (400) channels on Windows Server 2003 systems due to higher non-pageable system memory requirements introduced in this release.

  Customers are encouraged to start the migration toward newer, supported versions of the Windows operating system, such as Windows Server 2008.

Note: Windows Server 2003 Product Lifecycle Mainstream Support has ended, and Extended Support End Date has been announced for July 14, 2015. Consult the following link for more details:
http://support2.microsoft.com/lifecycle/search/default.aspx?sort=PN&alpha=Windows+Server+2003

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

With Service Update 349, Dialogic® PowerMedia™ HMP for Windows Release 3.0 introduces support for using multiple local IP interfaces for audio sessions in Global Call SIP 1PCC mode.

1.25.1 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 runtime 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 in IP Media Sessions. 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.

1.25.2 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 in IP Media Sessions.

  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.

1.25.3 Runtime Implementation

An application uses the Global Call library functions to control Audio Sessions within a dialog established in the1PCC mode. The following functionality is available with this feature.

1.25.3.1 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 set ID **IPSET_RTP_ADDRESS**, and parameter ID **IPPARM_LOCAL_ENUM**

The function will return an array of **RTP_ADDRESS** structures which contain the available RTP addresses.

In Windows, only IPv4 is available.

1.25.3.2 Example

```
int GetAvailableRTPAddresses(int nLineHandle)
{
        int             nError = GC_SUCCESS;
        long            nRequestID = 0;
        GC_PARM_BLKP    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(&parmblkpRTP, IPSET_RTP_ADDRESS,
IPPARM_LOCAL_ENUM, sizeof(rtpA), &rtpA);

        nError = gc_GetUserInfo(GCTGT_GCLIB_CHAN,nBoardHandle, &parmblkpRTP,
                                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(parmblkpRTP, &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;
                                int cbSize = sizeof(SOCKADDR);
                                sockaddr_in address = {0};
                                char localAddress[1024];
                                DWORD dwAddressSize = 1024;
                                address.sin_addr.S_un.S_addr = pRTPAddress->u_ipaddr.ipv4;
                                address.sin_family = AF_INET;
                                WSAAddressToString((SOCKADDR*)&address, cbSize, NULL,
localAddress, &dwAddressSize);

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

                        }
                 }
        }

        gc_util_delete_parm_blk(parmblkpRTP);

        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.

1.25.4 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 RTP address can only be an IPv4 address.

1.25.4.1 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 set ID **IPSET_RTP_ADDRESS**, and parameter ID **IPPARM_LOCAL**

1.25.4.2 Example

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

        rtpA.ip_ver = IPVER4;

        int     cbSize = sizeof(SOCKADDR);
        sockaddr_in address = {0};

        char    localAddress[1024];
        strcpy_s(localAddress, _countof(localAddress), sIPAddress);
        if (WSAStringToAddress(localAddress, AF_INET, NULL, (SOCKADDR*)&address, &cbSize) ==
0)
                rtpA.u_ipaddr.ipv4 = address.sin_addr.S_un.S_addr;

        /*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;
}
```

1.25.4.3 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 set ID **IPSET_RTP_ADDRESS**, and parameter ID **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.

1.25.4.4 Example

```
int SetLineRTPAddress(int nLineHandle, const char* sIPAddress)
{
        int                     nError = GC_SUCCESS;
        GC_PARM_BLKPparmblkpRTP = NULL;
        RTP_ADDR         rtpA = {0};

        rtpA.ip_ver = IPVER4;

        int     cbSize = sizeof(SOCKADDR);
        sockaddr_in address = {0};

        char    localAddress[1024];

        strcpy_s(localAddress, _countof(localAddress), sIPAddress);
        if (WSAStringToAddress(localAddress, AF_INET, NULL, (SOCKADDR*)&address, &cbSize) ==
0)
                rtpA.u_ipaddr.ipv4 = address.sin_addr.S_un.S_addr;

        /*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;
}
```

Note: 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.

1.25.4.5 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.

1.25.4.6 Example

```
void Set_GCLIBMAKECALLBLK_RTPAddress(GCLIB_MAKECALL_BLKP pGCMKB, const char* sIPAddress)
{
        GC_PARM_BLKP parmblkpRTP = pGCMKB->ext_datap;
        RTP_ADDR         rtpA = {0};

        rtpA.ip_ver = IPVER4;

        int     cbSize = sizeof(SOCKADDR);
        sockaddr_in address = {0};

        char    localAddress[1024];
        strcpy_s(localAddress, _countof(localAddress), sIPAddress);

        if (WSAStringToAddress(localAddress, AF_INET, NULL, (SOCKADDR*)&address, &cbSize) ==
0)
                rtpA.u_ipaddr.ipv4 = address.sin_addr.S_un.S_addr;

        /*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 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( ).

Note: This is irrespective of whether or not a previous Call on the same channel had used the Call-specific override method.

1.26 Support for SIP REFER for Adding a Participant in a Conference (RFC 4579, § 5.6)

With Service Update 349, Global Call now supports a new parameter ID which is used with set ID value of **IPSET_CONFIG** and can be configured with the help of the **gc_SetConfigData( )** at the IPT board device level, as follows:

Parameter ID	Data Type & Size	Description	SIP/H.323
IPPARM_RFC_4579_REFERER_REQ	Type: int, Size: sizeof(int)	This parameter is used to modify the behavior for handling an incoming **SIP REFER** and associated outgoing **NOTIFY** requests.	SIP

According to **RFC 4579 § 5.6**, a user agent (UA) wishing to add a new participant to a conference should send a **REFER** request to the participant with a **Refer-To** header containing the conference URI.

Using this parameter, the application can change the behavior for HMP's handling to an incoming **SIP REFER** request. The application may use the Global Call function **gc_AcceptXfer( )** in order to accept this **REFER** request outside of a Call Transfer scenario. Rejecting the request can be done with or **gc_RejectXfer( )** instead.

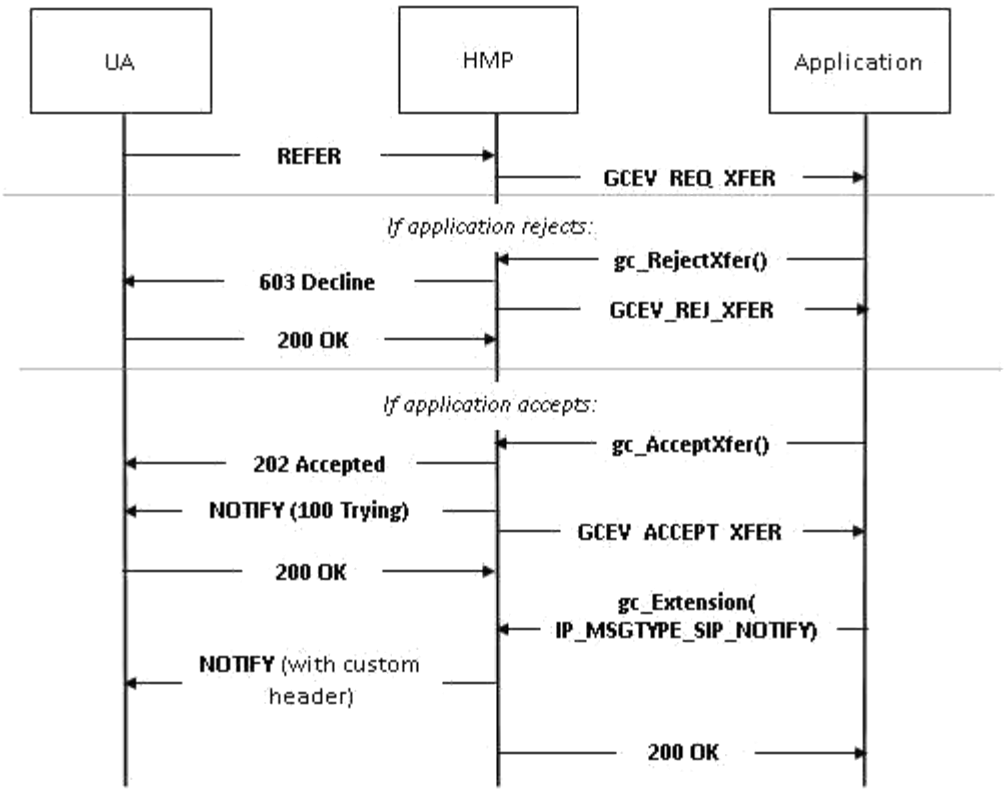
Note: Service Update 357 adds the ability to handle multiple REFER messages on the same call.

In addition, the **RFC 4579 § 5.6** scenario may require the sending of one or several **NOTIFY** messages with custom SIP headers; the requests can be sent inside of a **SIP** dialog. With this new parameter, Global Call allows the sending of **SIP NOTIFY** requests using the function **gc_Extension( )** and the specific Call's CRN.

Note that the SIP Stack automatically increments the **CSeq** number for each subsequent **SIP NOTIFY** message going out the network so long it is sent inside of a SIP dialog.

Note: Once the desired **NOTIFY** custom header fields are set up, the application can send the message within a dialog: **gc_Extension**(GCTGT_GCLIB_CRN, crn, IPEXTID_SENDMSG, parmblkp, &retblkp, EV_ASYNC).

This diagram expands on the usage of **SIP REFER** for this scenario, and describes the GC events expected to be received by the application along their handling:



Example

The below example shows the setting of the new parameter **IPPARM_RFC_4579_REFERER_REQ** in the application to set limited **RFC 4579 § 5.6** scenario support:

```
long request_id = 0;
GC_PARM_BLKP parmblkp = NULL;
gc_util_insert_parm_val(&parmblkp, IPSET_CONFIG,
        IPPARM_RFC_4579_REFERER_REQ, sizeof(int), IP_ENABLE);

if (gc_SetConfigData(GCTGT_CCLIB_NETIF, bdev, parmblkp, 0,
        GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC) != GC_SUCCESS)
{
        // error handling
}
```

Note: The "bdev" in this context refers to the Global Call **gc_OpenEx( )** LineDevice from the IPT board-level device.

This example shows how configuring a custom header to be sent in (in-dialog) **NOTIFY** message to the UA:

```
char *pTo = "From: sipp <sip:sipp@192.168.86.9:5062>";
char *pFrom = "To: sut <sip:1234567890@192.168.130.199:5062>";
char *pContact = "Contact: <sip:WSPCDEV11@172.16.14.5>;isfocus";
char *pEvent = "refer;id=3"; char *pContentType = "Content-Type: message/sipfrag";
char *pBody = "SIP/2.0 200 OK";
char *pBlankBody = "";

GC_PARM_BLKP parmblkp = NULL;
GC_PARM_BLKP parmblkbody = NULL;
GC_PARM_BLKP retblkp = NULL;

gc_util_insert_parm_val(&parmblkp, IPSET_MSG_SIP, IPPARM_MSGTYPE,
        sizeof(int), IP_MSGTYPE_SIP_NOTIFY);
gc_util_insert_parm_ref_ex(&parmblkp, IPSET_SIP_MSGINFO,
        IPPARM_SIP_HDR, (unsigned long)(strlen(pTo) + 1), pTo);
gc_util_insert_parm_ref_ex(&parmblkp, IPSET_SIP_MSGINFO,
        IPPARM_SIP_HDR, (unsigned long)(strlen(pFrom) +1 ), pFrom);
gc_util_insert_parm_ref_ex(&parmblkp, IPSET_SIP_MSGINFO, IPPARM_SIP_HDR,
        (unsigned long)(strlen(pContact) + 1), pContact);
gc_util_insert_parm_ref_ex(&parmblkp, IPSET_SIP_MSGINFO,
        IPPARM_EVENT_HDR, (unsigned long)(strlen(pEvent) + 1), pEvent);
                gc_util_insert_parm_ref_ex(&parmblkbody, IPSET_MIME,
           IPPARM_MIME_PART_TYPE, (unsigned long)(strlen(pContentType) +1 ),
            pContentType); gc_util_insert_parm_val(&parmblkbody, IPSET_MIME,
                              IPPARM_MIME_PART_BODY_SIZE,
        sizeof(unsigned long), strlen(pBody));
gc_util_insert_parm_val(&parmblkbody, IPSET_MIME, IPPARM_MIME_PART_BODY,
        sizeof(unsigned long), (unsigned long)pBody);
gc_util_insert_parm_val(&parmblkp, IPSET_MIME, IPPARM_MIME_PART,
        sizeof(unsigned long), (unsigned long)parmblkbody);

if (gc_Extension(GCTGT_GCLIB_CRN, crn, IPEXTID_SENDMSG, parmblkp, &retblkp, EV_ASYNC))
        != GC_SUCCESS)
{
        // error handling
}
```

1.26.1 Ability to Handle re-INVITE Hold in SIP REFER

With Service Update 371, Global Call now supports a new parameter value, for the **IPSET_MSG_SIP** set ID and **IPPARM_MSGTYPE** parameter ID, as follows:

Set ID	Parameter ID	Data Type & Size	Description	SIP/H.323
IPSET_MSG_SIP	IPPARM_MSGTYPE	Type: int, Size: sizeof(int)	Sets type of supported SIP message to send using **gc_Extension( )** and the **IPEXTID_SENDMSG** extension ID. Defined values are: • IP_MSGTYPE_SIP_REINVITE_ACCEPT • IP_MSGTYPE_SIP_REINVITE_REJECT The set ID **IPSET_CONFIG** and parameter ID **IPPARM_RFC_4579_REFERER_REQ** must be enabled globally on the IPT board device.	SIP

The new **IPPARM_MSGTYPE** SIP IP messages are used in the context of this RFC scenario when the application receives a **GCEV_REQ_MODIFY_CALL** event, indicating a re-INVITE from the SIP agent for the purposes of modifying the media attributes, in particular for stopping or resuming the RTP stream.

When the application has set the set ID **IPSET_CONFIG**/ parameter ID **IPPARM_RFC_4579_REFERER_REQ** to **IP_ENABLED** globally on the IPT board device, a re-INVITE is allowed in the context of the RFC 4579 Conference scenario as described above.

Under this condition, the application can accept or reject the request for session change via the set ID/parameter ID and either **IP_MSGTYPE_SIP_REINVITE_ACCEPT** or **IP_MSGTYPE_SIP_REINVITE_REJECT** message, respectively.

Note:

- The **gc_AcceptModifyCall( )** and **gc_RejectModifyCall( )** are not supported and will fail in the context of an RFC 4579 Conference.
- The **GCEV_EXTENSIONCMPLT** should not be expected in this case; either a **GCEV_ACCEPT_MODIFY_CALL** or **GCEV_REJECT_MODIFY_CALL**.

The **IP_MSGTYPE_SIP_REINVITE_ACCEPT** message functions like the **gc_AcceptModifyCall( )** in that the message carries the media attributes of the call that is being accepted in an additional GC parameter block.

The application adds the set ID **IPSET_MSG_SIP**, parameter ID **IPPARM_MSGTYPE**, and Msg value **IP_MSGTYPE_SIP_REINVITE_ACCEPT** as first block in a **GC_PARM_BLK** structure with the **gc_util_insert_parm_val( )**.

Next each media attribute that are being accepted are added one after the other one using the appropriate set ID/parameter ID exactly as it would've been done in a **gc_AcceptModifyCall( )**, using the appropriate GC Utility function(s).

The **IP_MSGTYPE_SIP_REINVITE_REJECT** message functions like the **gc_RejectModifyCall( )** in that the message carries the reason for rejecting request to change call attribute.

The application adds the set ID **IPSET_MSG_SIP**, parameter ID **IPPARM_MSGTYPE**, and Msg value **IP_MSGTYPE_SIP_REINVITE_REJECT** as first block in a **GC_PARM_BLK** structure with the **gc_util_insert_parm_val( )**.

A reason value for rejecting the request to modify the media attributes is added next as an additional GC block in the same **GC_PARM_BLK** structure, also using the **gc_util_insert_parm_val( )**. In this case the already existing set ID **IPSET_MSG_SIP**, parameter ID **IPPARM_MSG_SIP_RESPONSE_CODE**, and the desired response code as the value, as deemed appropriate (e.g., **IPEC_SIPReasonStatus406NotAcceptable**)

The application can now send the previously configured **GC_PARM_BLK** structure via a call to **gc_Extension( )** API.

As described, the Extension ID is **IPEXTID_SENDMSG**, and the Call's CRN as the Target ID with EV_ASYNC as its mode.

If the message is successfully sent, the application will receive a **GCEV_ACCEPT_MODIFY_CALL** or **GCEV_REJECT_MODIFY_CALL** respectively, which can be processed as usual. If a failure occurs a **GCEV_TASKFAIL** will be received instead.

Example

Snippet of code for the new Parm ID values:

- IP_MSGTYPE_SIP_REINVITE_ACCEPT
- IP_MSGTYPE_SIP_REINVITE_REJECT

```
static void     gc_respond_to_req_modify_call(struct channel *pline, GC_PARM_BLK
*a_pParmBlk, char bAccept)
{
        int     count = 0, callindex;
        GC_PARM_BLK *lpgcParmBlk = NULL;
        GC_PARM_BLK *retblkp = NULL;
        GC_PARM_DATAP curParm = NULL;

        /* Retrieve CRN  */
        callindex = ....;
...
        if (bAccept) {
                // Insert SIP message type

gc_util_insert_parm_val(&lpgcParmBlk,IPSET_MSG_SIP,IPPARM_MSGTYPE,sizeof(int),

IP_MSGTYPE_SIP_REINVITE_ACCEPT);

                // Extract GCSET_CHAN_CAPABILITY ParmIDs passed in a_pParmBlk
obtained from re-INVITE, in a loop
                while ((NULL != a_pParmBlk) && ( curParm = gc_util_next_parm
( a_pParmBlk, curParm ) ) != NULL ){
                        //If it is NULL, then just use the default information
                        if ( curParm->set_ID == GCSET_CHAN_CAPABILITY &&
curParm->parm_ID == IPPARM_LOCAL_CAPABILITY ){
                                IP_CAPABILITY *pCap = ( IP_CAPABILITY * )curParm-
>value_buf;
                                IP_CAPABILITY t_Capability;

                                memset( &t_Capability, 0, sizeof ( t_Capability ) );
                                t_Capability.capability = pCap->capability;
                                        t_Capability.type = pCap->type;
                                        t_Capability.direction = pCap->direction;
                                        t_Capability.payload_type = pCap-
>payload_type;
                                        t_Capability.extra.audio.frames_per_pkt =
pCap->extra.audio.frames_per_pkt;
```

```
                                        t_Capability.extra.audio.VAD = pCap-
>extra.audio.VAD;

                                if ((t_Capability.direction ==
IP_CAP_DIR_LCLRTPRTCPINACTIVE) ||
                                                (t_Capability.direction ==
IP_CAP_DIR_LCLRTPINACTIVE))
                                {
                                        // this is to respond HOLD CALL request from
remote end.
                                        // we only need one IP_CAPABILITY parameter
for gc_AcceptModifyCall().

                                        if (count == 0) {  // try to add only one of
entry.
                                                if (
gc_util_insert_parm_ref(&lpgcParmBlk,GCSET_CHAN_CAPABILITY,IPPARM_LOCAL_CAPABILITY
,
                                                                        sizeof ( IP_CAPABILITY
),&t_Capability) != GC_SUCCESS ){
                                                        // Process error
                                                }
                                                count ++ ;     // one time only.
                                        }
                                }
                                else
                                {
                                        if(t_Capability.direction ==
IP_CAP_DIR_LCLSENDONLY) {
                                                        t_Capability.direction =
IP_CAP_DIR_LCLRECVONLY;
                                        }
                                        else if(t_Capability.direction ==
IP_CAP_DIR_LCLRECVONLY) {
                                                        t_Capability.direction =
IP_CAP_DIR_LCLSENDONLY;
                                        }
                                        // this is to respond RETRIEVE Call request
from remote end.
                                        // we need two IP_CAPABILITY parameters (TX
and RX) for gc_AcceptModifyCall().

                                        if (
  gc_util_insert_parm_ref(&lpgcParmBlk,GCSET_CHAN_CAPABILITY,IPPARM_LOCAL_CAPABILITY
,
                                                        sizeof ( IP_CAPABILITY
),&t_Capability) != GC_SUCCESS ){
                                                // Process error
                                        }
                                }
                        }
                }
        }
        else // Reject
```

```
        {
                // Insert SIP message type

gc_util_insert_parm_val(&lpgcParmBlk,IPSET_MSG_SIP,IPPARM_MSGTYPE,sizeof(int),

IP_MSGTYPE_SIP_REINVITE_REJECT);
                // Insert Reason code in payload block
                const ULONG cause = IPEC_SIPReasonStatus406NotAcceptable;
                //const ULONG cause = IPEC_SIPReasonStatus500ServerInternalError;

                if (
gc_util_insert_parm_val(&lpgcParmBlk,IPSET_MSG_SIP,IPPARM_MSG_SIP_RESPONSE_CODE,
                                                        sizeof(ULONG),cause) !=
GC_SUCCESS) {
                        // Process error
                }
        }

        if (gc_Extension(GCTGT_GCLIB_CRN,pline-
>call[callindex].crn,IPEXTID_SENDMSG,lpgcParmBlk,&retblkp,EV_ASYNC) != GC_SUCCESS)
{
                // Process error
        }
        gc_util_delete_parm_blk(lpgcParmBlk);
        lpgcParmBlk = NULL;

}  /* End of function gc_respond_to_req_modify_call() */
```

1.27 Support for Dialogic® DSI SS7LDH4Q Network Interface Board

With Service Update 349, the PCI Express half-size, full-profile Dialogic® DSI SS7LDH4Q Digital Network Interface Board is supported by HMP and can take advantage of Global Call SS7 functionality.

This Network Interface DSI SS7 board can be configured and can operate in stand-alone mode or along DNIxxxxTEPE2HMP 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.

1.27.1 Feature Implementation

With this feature the SS7LD is integrated in the HMP OA&M allowing the board to be started and stopped within the context of the Dialogic System Service.

Applications can take full advantage of the Dialogic® Global Call SS7 functionality which provides support for ISUP and/or TUP SS7 protocols on DSI SS7-enabled boards.

To take advantage of this new feature, the Dialogic® DSI Development Package for Windows version 6.4.2 or later (DPK 6.4.2) must also be installed. A separate DSI SW license is also required. Refer to the "Dialogic® Distributed Signaling Interface Network Interface Boards" page at:

http://www.dialogic.com/products/signaling-and-ss7-components/download/dsi-network-interface-boards.aspx

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 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.

1.27.2 Feature Limitations

- Refer to the *Dialogic® DSI Development Package Release Notes* (*http://resource.dialogic.com/telecom/support/ss7/cd/GenericInfo/DevelopmentPackages/Windows/rn001dpk.pdf*) for up-to-date information regarding functionality and limitations with the SS7LD network interface board.
- HMP media streaming is not possible with the SS7LD board.
- The SS7LD does not have a CT bus; however it provides connectivity for the Dialogic® SyncRoute bus which has limited H.100 capability. The SyncRoute bus supports the clock/frame synchronization and clock fall back features of the CT bus (TDM bus). These are some important considerations and limitations related to SS7LD Clocking configuration in HMP:
  - The SS7LD can get attached to the HMP TDM bus; therefore TDM configuration for the board is available via the HMP TDM configuration procedures as indicated in the "TDM Bus Configuration of Dialogic® SS7 Boards" section in *Dialogic® Global Call SS7 Technology Guide*.

  Note: The **flags** parameter in the **SS7_BOARD** command (*config.txt* file) is used to configure the SS7LD's TDM settings; the parameter is parsed in order to determine its initial clocking configuration. Only if this parameter configured for the board to be a Master FRU, will its clock configuration be relevant; otherwise the board can be a Slave FRU or possibly, be completely isolated from the CT Bus.

  For more information on the SS7LD clocking capabilities and settings, and in particular of this parameter, refer to the *Dialogic® DSI SS7LD Network Interface Boards - Programmer's Manual* (*http://www.dialogic.com/~/media/manuals/ss7/cd/ProductSpecific/SS7LD/Programmers_Manual/SS7LD-PM.pdf*).

  The document provides details of the **flags** parameter in the **MGT_MSG_CONFIG0** message, specific to this board.

- Only when the SS7LD is in the system cabled to other CT bus capable boards via the SyncRoute, whether of the same type or with a DNIxxx board, its CT bus clocking configuration should be setup to interface with the CT bus clock (i.e., configured as a Clock Master FRU to drive CT bus clock set A or B).

  Note: While the **flags** parameter configuration allows an SS7LD to drive the CT_NETREF1, its configuration isn't supported in HMP; thus it is recommended that (Bit 13) it is set to disabled.

  When the SS7LD is cabled via the SyncRoute, care must be taken to properly configure the bus termination. Refer to the *Dialogic® DSI SS7LD Installation Guide* (*http://www.dialogic.com/~/media/manuals/ss7/cd/ProductSpecific/SS7LD/InstallationGuide/SS7LD-IG.pdf*).

- When the board isn't cabled to other (one or more) SS7LD or DNIxxx board(s) in the same system, the SS7LD CT bus interface must be disabled so the board is electrically isolated from the other boards in the TDM bus. To do that, the **flags** parameter in the **SS7_BOARD** command must be set to disable the CT bus interface.

  Note: This parameter also allows its clock configuration to either use its internal oscillator or else recover the clock from one of its network interfaces.

  Since the SS7LD provides no HMP streaming capabilities, providing the board is not interconnected to other boards in the system via its network interfaces, it is usually more practical to disable the board's CT bus interface and have it recover the TDM clock from one of its line interfaces.

1.27.3 Board Configuration and Startup

The Dialogic System Service can detect the SS7LD providing it is properly configured in the DSI subsystem. Once that is accomplished the board is visible within the DCM as part of an SS7 Family of boards, with the name displayed as "SS7LD #N" (i.e., N=0, 1, 2, etc.).

- The "ConfigFile" Parameter in the "System" tab for the board entity in DCM reflects the name and path for the GC SS7 configuration file (*gcss7.cfg*); this value cannot be changed from the DCM or NCM.
- The "Misc" and "TDM Bus Configuration" tabs display various relevant Parameters; however none of them can be modified from the utility.

In general the GC SS7 configuration and startup information applies to the SS7LD, except when indicated otherwise in this document.

1.27.4 SS7 Configuration

The DSI SS7 configuration for the SS7LD is as documented in the *Dialogic® DSI Software Environment Programmer's Manual*. This includes the DSI Parameters in the System and Protocol Configuration Files.

1.27.4.1 GC SS7 Configuration File

- 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"
```

- It's recommended leaving the **Module ID** unchanged; otherwise SS7 System Configuration needs changes accordingly.

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

- The **MtpLink** source and parameters are not relevant and not used for this board.
- The **ClearGrp** parameter is neither used nor relevant in this environment since the board does not support CT bus routing and cannot be used in Clear Channel mode either.
- Global Call SS7 Circuit Group can be configured for either ISUP or TUP User-Part protocols and follow these guidelines:
  - The **<trunk_name>** should be set to "dkBx".
  - The CGrp **<gid>** must match the group id of 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 determination of what underlying LIU timeslots are used for the CIC and MTP link(s) is determined by the GC SS7 SW by fetching the **<cic_mask>** entry per **<gid>** in each **XXX_CFG_CCTGRP** command in the Protocol configuration file. Only those physical time slots that are associated with voice circuits are enumerated as channel devices for each CGrp.
  - The optional [**<base_TS>**] can also be used as documented.

For more information, refer to the “Dialogic® Global Call SS7 Software Configuration (gcss7.cfg)” section in *Dialogic® Global Call SS7 Technology Guide*.

Note: When the SS7LD is configured along DNIXXXXTEPE2HMP board(s) with SS7 combined functionality, the DNI configuration still follows the guidelines as per the Feature Configuration section in Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards; this is irrespective of SS7LD board presence.

1.27.5 SS7 Startup

Refer to the “Starting a Dialogic® SS7 Board System” section in *Dialogic® Global Call SS7 Technology Guide*.

Note: When the SS7LD is configured along DNIXXXXTEPE2HMP board(s) with SS7 combined functionality, the DNI board startup in Atomatic mode still requires the GC SS7 Service (DlgcS7Srv) to first be installed. This is a one-time configuration that can be done with the "DlgcS7Srv.exe Install" from a DOS cmd prompt. Refer to the Automatic DSI SS7 Start and Stop within Dialogic System Service section in Support for Combined DSI SS7LD Stack and Media Streaming on Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards; failure to do so might render the entire SS7 functionality to malfunction. This is irrespective of SS7LD board presence.

1.27.6 Sample Configuration

In this case one SS7LD and one DNI2410TEPE2HMP are configured in the same system; each board carries separate MTP links.

Each LIU is connected back to back to another LIU on the same board for the purposes of this sample.

In this case the SS7LD and the DNI board are connected with the SyncRoute cable; the DNI board is the Primary TDM Clock source and the SS7LD is configured to be a Slave; the SS7LD is also configured **not** to drive the NETREF 1 line.

The SS7LD is configured with four (4) MTP links using physical tslots 15 & 16 on the first and 2nd LIU (as they are connected back to back); the DNI2410TEPE2HMP is configured with two (2) MTP links, by using tslot 16 on first and 2nd LIU (as they are connected back to back).

For the DSI configuration files, only the relevant parameters are shown here.

```
******************************************************************************* *

* Example System Configuration File (system.txt)
*
*******************************************************************************
* The following lines identify the various modules, running on this system
*
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
*
* 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 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.
*
* Processes started with "-t" are running in trial mode - they will
* automatically halt after 10 hours and must be restarted
*
FORK_PROCESS ./ssdl -d -Lp -o3 -a0,3
FORK_PROCESS ./tim
FORK_PROCESS ./tick
FORK_PROCESS ./s7_mgt -d -i0x4d
FORK_PROCESS ./s7_log -fss7.log -m0xef
*

******************************************************************************* *

* Example Protocol Configuration File (config.txt)
*
*******************************************************************************
* Configure individual boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
*
* CTBus Slave (bits 6 & 7 on)
SS7_BOARD 0 SS7LD 0x0060 ./DC/ss7.dc7 ISUP
* DNI2410TEPE2HMP SS7 Combined
SS7_BOARD 1 DNI2410 0x0000 "C:\Program Files\Dialogic\HMP\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 for the SS7LD (only)
* LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format>
*           <crc_mode> [<build_out>]
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>
MTP_LINKSET 0 900 3 0x0800 959 0x0008
MTP_LINKSET 1 959 3 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>
*
```

```
* SS7LD Links
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 0 0x2  0 15 0x0006
MTP_LINK 0x3  1 1 1 0 0x3  1 15 0x0006
* DNI2410TEPE2HMP combined Links
MTP_LINK 0x4  0 2 2 1 0x0  0 16 0x0006
MTP_LINK 0x5  1 2 2 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>
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
* 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.
* The base CIC (and the range of CICs) will need to be unique for each group between each two
* If more than one group configured for a specific DPC the CICs must be unique to each.
* The base CID is the logical identifier passed over the API so must be unique to the system not

* just to any given DPC.
* CIC_mask - if bit is 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>
* SS7LD Circuits - Exclude ts 16 and 15
ISUP_CFG_CCTGRP   0  900 0x01  0x00   0x7fff3fff 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   1  959 0x01  0x40   0x7fff3fff 0x001c  0  0x4d  900  0x8  0 0x00
ISUP_CFG_CCTGRP   3  959 0x21  0x60   0x7fffffff 0x001c   0  0x4d   900  0x8   0 0x00

* DNI2410TEPE2HMP combined Circuits - Exclude ts 16
ISUP_CFG_CCTGRP   4  900 0x41  0x80   0x7fff7fff 0x001c  0  0x4d  959  0x8  0 0x00
ISUP_CFG_CCTGRP   6  900 0x61  0xa0   0x7fffffff 0x001c  0  0x4d  959  0x8  0 0x00
ISUP_CFG_CCTGRP   8  900 0x81  0xc0   0x7fffffff 0x001c  0  0x4d  959  0x8  0 0x00
ISUP_CFG_CCTGRP   10 900 0xa1  0xe0   0x7fffffff 0x001c  0  0x4d  959  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
```

Examination of the GC SS7 log file (*DlgcS7.log*) shows the enumeration of **dkB1** channel devices; since the **<cic_mask>** excludes tslots 15 and 16 which are used for the MTP links, those two tslots are not enumerated; the tslots are bypassed and the devices are sequentially enumerated from **dkB1T1** through **dkB1T29**. The rest of the circuits are not shown.

```
INFO: dkB1 ISUP_CFG_CCTGRP mappings:
INFO: base_ts=0x1, gid=0x0, board_id=0x0, liu_id=0x0
INFO: base_cid=0x0, cic_mask=0x7fff3fff, clear_chan:0, ts_mask=0x7fff3fff
```

```
  DEBUG: Circuit NB:1, Name:dkB1T1, CID=0x0, TS=1
  DEBUG: Circuit NB:2, Name:dkB1T2, CID=0x1, TS=2
  DEBUG: Circuit NB:3, Name:dkB1T3, CID=0x2, TS=3
  DEBUG: Circuit NB:4, Name:dkB1T4, CID=0x3, TS=4
  DEBUG: Circuit NB:5, Name:dkB1T5, CID=0x4, TS=5
  DEBUG: Circuit NB:6, Name:dkB1T6, CID=0x5, TS=6
  DEBUG: Circuit NB:7, Name:dkB1T7, CID=0x6, TS=7
  DEBUG: Circuit NB:8, Name:dkB1T8, CID=0x7, TS=8
  DEBUG: Circuit NB:9, Name:dkB1T9, CID=0x8, TS=9
  DEBUG: Circuit NB:10, Name:dkB1T10, CID=0x9, TS=10
  DEBUG: Circuit NB:11, Name:dkB1T11, CID=0xa, TS=11
  DEBUG: Circuit NB:12, Name:dkB1T12, CID=0xb, TS=12
  DEBUG: Circuit NB:13, Name:dkB1T13, CID=0xc, TS=13
  DEBUG: Circuit NB:14, Name:dkB1T14, CID=0xd, TS=14
  DEBUG: Circuit NB:15, Name:dkB1T15, CID=0x10, TS=17
  DEBUG: Circuit NB:16, Name:dkB1T16, CID=0x11, TS=18
  DEBUG: Circuit NB:17, Name:dkB1T17, CID=0x12, TS=19
  DEBUG: Circuit NB:18, Name:dkB1T18, CID=0x13, TS=20
  DEBUG: Circuit NB:19, Name:dkB1T19, CID=0x14, TS=21
  DEBUG: Circuit NB:20, Name:dkB1T20, CID=0x15, TS=22
  DEBUG: Circuit NB:21, Name:dkB1T21, CID=0x16, TS=23
  DEBUG: Circuit NB:22, Name:dkB1T22, CID=0x17, TS=24
  DEBUG: Circuit NB:23, Name:dkB1T23, CID=0x18, TS=25
  DEBUG: Circuit NB:24, Name:dkB1T24, CID=0x19, TS=26
  DEBUG: Circuit NB:25, Name:dkB1T25, CID=0x1a, TS=27
  DEBUG: Circuit NB:26, Name:dkB1T26, CID=0x1b, TS=28
  DEBUG: Circuit NB:27, Name:dkB1T27, CID=0x1c, TS=29
  DEBUG: Circuit NB:28, Name:dkB1T28, CID=0x1d, TS=30
  DEBUG: Circuit NB:29, Name:dkB1T29, CID=0x1e, TS=31


##################################@@@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 ciruit 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/deamon #
################################################

# Logging Level for the service (DlgcS7.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"
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 mechanisme
# Format: Integer; Default: 7; 0 means the mechanisme is off (recommended for
Windows) #Service.WatchDogMaxTime = 0


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

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

# 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
```

1.28 Configuring a Network Interface in Tristate or Line Monitor Modes

Service Update 349 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.

1.28.1 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.

1.28.2 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).

1.28.3 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 DNIxxxxTEPE2HMP; 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 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).

1.29 Support for Windows Server 2012 R2 Standard and Windows 8.1 Pro Operating Systems

Service Update 347 introduces support for Windows Server 2012 R2 Standard and Windows 8.1 Pro operating systems. For information about installing and using the Dialogic® PowerMedia™ HMP for Windows Release 3.0, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Installation Guide*.

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

With Service Update 347, the PCI Express half-size, full-profile, Dialogic® DNIxxxxTEPE2HMP Digital Network Interface Boards, add SS7LD support combined with full Dialogic® PowerMedia™ HMP for Windows Release 3.0 media streaming.

1.30.1 Feature Implementation

The Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP Digital Network Interface Boards, when operating along the Dialogic® DSI Windows Development Package (DPK) version 6.4.2 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 Windows version 6.4.2 or later (DPK 6.4.2) must also be installed. Separate SW licenses for HMP media and SS7 signaling are required.

1.30.2 Feature Limitations

Refer to the *Dialogic® DSI Development Package Release Notes* (*http://resource.dialogic.com/telecom/support/ss7/cd/GenericInfo/DevelopmentPackages/Windows/rn001dpk.pdf*) for up-to-date information regarding signaling support on the DNIxxxxTEPE2HMP Digital Network Interface Boards.

While the DNIxxxxTEPE2HMP 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.

While the SS7 stack can be started manually, the automatic mode is recommended; refer to the Automatic DSI SS7 Start and Stop within Dialogic System Service section for more information.

Whether in automatic or manual mode, the Dialogic® Distributed Signaling SW environment cannot be restarted without re-starting the Dialogic System Service.

HMP product licenses do not provide SS7 protocol licensing. Separate SS7 protocol licenses are required; since the DNIxxxxTEPE2HMP is functionally equivalent to the SSLD 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.

1.30.3 Feature Configuration

The HMP and DPK SW releases are installed separately and can be done in any particular order; however since both the DNIxxxxTEPE2HMP, 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, or it will cause the DNIxxxxTEPE2HMP in HMP.

Refer to the *Dialogic® DSI Software Environment Programmer's Manual* for software installation instructions for Windows. The DPK installation procedure will ask you to select the driver to be installed; it is important NOT to install the SS7LD device driver.

Note: Installing the SS7LD device driver will cause the DNIxxxxTEPE2HMP board to malfunction in ways that might not be obvious.

1.30.3.1 Board Configuration and Startup

For SS7 configurations, the DNIXXXXTEPE2HMP must be set to Clear Channel, either T1CC or E1CC.

SS7 LIU configuration is provisioned by the DNI Line Administration (LineAdmin) Parameters, which is normally done at system configuration time using the board's .config/.fcd file pair.

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

Complete HMP / SS7 startup is a two-step process; first the HMP and DNIXXXXTEPE2HMP 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 DNIXXXXTEPE2HMP board as SS7 functionality equivalent to an SS7LD board.

Automatic DSI SS7 Start and Stop within Dialogic System Service

The Dialogic System Service can be configured to start both subsystems at once.

This is done by means of the Global Call SS7 configuration (*gcss7.cfg*) file, which can be set up to start the DSI subsystem automatically by means of the GC SS7 Service SW (DlgcS7Srv) "Service.xxxx" parameters; refer to the *Global Call SS7 Technology Guide* for more information about these parameters.

In order to take advantage of this mode, the GC SS7 Service (DlgcS7Srv) must first be installed. This is a one-time configuration that can be done with the "DlgcS7Srv.exe Install" from a DOS cmd prompt. If the Service.GCTLOAD_Control is set to "Yes" the Service will start automatically within the Dialogic System Service on subsequent start ups; it will be stopped automatically whenever the Dialogic System Service is stopped.

Note: To remove this setting one can type: "DlgcS7Srv.exe UnInstall" from a DOS cmd prompt.

Furthermore, the GC SS7 DlgcS7Srv will also use the setting of "Service.GCTLOAD_Control" in gcss7.cfg configuration file and make an attempt to start the DSI GCTLOAD component, if set to "Yes"; therefore starting the DSI environment if it was configured properly. By the same token, the GC SS7 DlgcS7Srv will make an attempt to stop the DSI environment when the former is stopped.

With automatic mode setting, the Dialogic System Service should yield full Start/Stop control on the GC SS7 SW DlgcS7Srv and DSI subsystem, readying the system for operation without the need for further initialization required. This 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 failure.

Manual DSI Start and Stop within HMP

In manual mode, the GC SS7 DlgcS7Srv is not installed and the user is responsible for invoking it upon the Dialogic System Service and DSI environment startup.

1.30.4 SS7 Configuration

The SS7 SW stack supported on the DNIXXXXTEPE2HMP interface boards provides the same capabilities 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* (*http://www.dialogic.com/~/media/manuals/ss7/cd/GenericInfo/GeneralDocumentation/U10SSS-SwEnv-PM.pdf*).

Note: For ISUP-based systems, runtime licenses which bundle ISUP, MTP3, and MTP2 functionality that run within the SSDL binary, are available.

Two modes are still possible:

- If the Service.GCTLOAD_Control / GCTLOAD_Path are still configured for automatic GCTLOAD control (Start/Stop); in which case the DlgcS7Srv will still make an attempt to start the GCTLOAD. Similarly for stopping it.

  Note: In this case restarting the DlgcS7Srv is possible without restarting the DSI and the Dialogic System Service.

- If the "Service.GCTLOAD_Control = No" the DlgcS7Srv will not attempt to start/stop the GCTLOAD. The user will first start the GCTLOAD and then proceed to start the GC SS7 DlgcS7Srv. 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 DlgcS7Srv is not possible without restarting the Dialogic System Service and DSI SW first.

Once both subsystems are ready, the GC SS7 can be started manually by means of:

"DlgcS7Srv.exe Start"

If successful "…OK" is printed in the console; if not an error is returned.

One should stop it first prior to DSI gctload and Dialogic System Service shutdown:

"DlgcS7Srv.exe Stop"

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

1.30.4.1 DSI System Configuration File

These are the minimum System Configuration File settings required for proper SS7 operation; it this does not provide a comprehensive list and 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 program 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 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
```

1.30.4.2 DSI Protocol Configuration File

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 Command, is used to configure the DNIXXXXTEPE2HMP board in the system.

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

   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 by default under:
   C:\ProgramData\Dialogic\data\hmp2_mixed.bin
   The <board_type> 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 relative order of boards is loosely associated either with the HMP InstanceNumber or the PhysicalSlotNumber of the Physical Property Sheet in the DCM (refer to the *Dialogic® Host Media Processing Configuration Guide*), depending on which <addressing mode> is used in the ssdl module options. This association is discovered by the GC SS7 SW in order to properly map MTP links and voice circuits with the board's bearer channels; refer to the GC SS7 Configuration File section for more information.

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

```
SS7_BOARD 0 DNI2410 0x0000 c:\ProgramData\Dialogic\data\hmp2_mixed.bin
ISUP
SS7_BOARD 1 DNI2410 0x0000 c:\ProgramData\Dialogic\data\hmp2_mixed.bin
ISUP
```

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 DNIXXXXTEPE2HMP combined environment since all LIU settings are taken care by the DNI Line Administration settings in it configuration file.

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

1.30.4.3 GC SS7 Configuration File

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

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

2. It's 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 combined environment.

4. The ClearGrp parameter is neither used nor relevant in this combined environment since the board supports clear-channel circuits within itself.

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").

- 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 of 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.

1.30.5 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 an 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 BChannelId 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 BChannelId 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 Lines 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 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's 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.

1.30.6 Sample Configuration

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 Ids <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, the corresponding 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 which by means of the corresponding CGrp entry in the gcss7 config file for that <gid>:

```
CGrp 0 dtiB1

******************************************************************************** *

* Example Protocol Configuration File (example_config.txt)
*
********************************************************************************
*
* Configure individual boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
*
SS7_BOARD 0 DNI2410 0x0000 c:\ProgramData\Dialogic\data\hmp2_mixed.bin ISUP
SS7_BOARD 1 DNI2410 0x0000 c:\ProgramData\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_ccts> 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
********************************************************************************



####################################@@@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 ciruit 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/deamon #
################################################

# Logging Level for the service (DlgcS7.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 mechanisme
# Format: Integer; Default: 7; 0 means the mechanisme 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 transfered 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
#
```

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, is bypassed with the <base_TS>=2.

Also, note that the <cic_mask> does not have a TS16 exclusion even though we are carrying the MTP links in LIU one and two. 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, thus MTP3 and TUP are running with the trial-mode license (-t).

```
*************************************************************************** *

* 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           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 (example_config.txt)
*
*******************************************************************************
* 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_ccts>
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
```

1.31 Support for Multiple NICs in IP Media Sessions

With Service Update 347, Dialogic® PowerMedia™ HMP for Windows Release 3.0 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.

1.31.1 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 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 runtime 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 allows 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 multi-homed 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.

1.31.2 Runtime Implementation

An application uses the IP Media library functions to control media sessions within a media session 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.

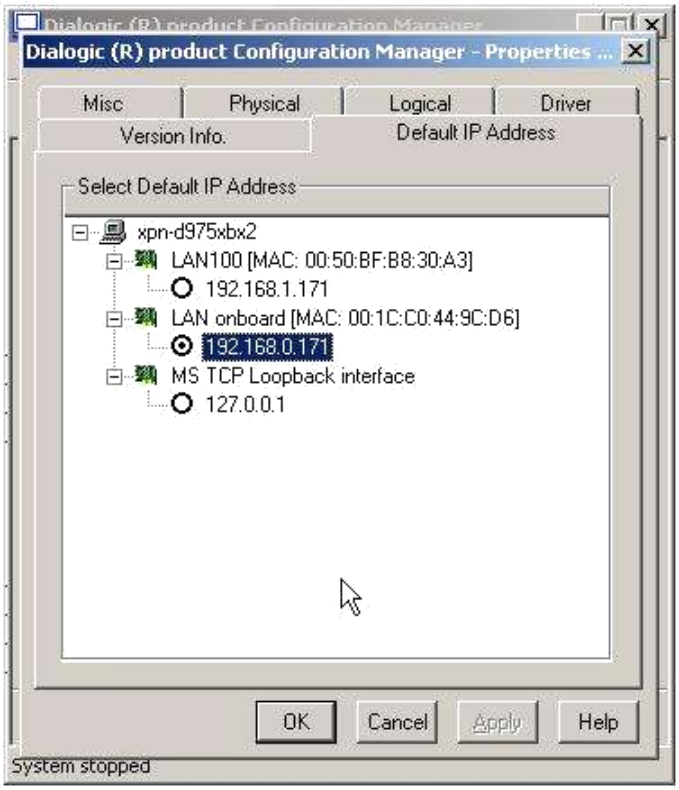
Media session attributes and values in the IP Media library are controlled by the settings in the IPM_MEDIA data structure. This structure contains information about RTP/RTCP ports, coders, security information, etc. It is a parent structure of the IPM_PORT_INFO and contains UDP port properties for audio 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;
```

1.31.3 Initialize the Interfaces Table

Initialization is accomplished when Dialogic System Service is started. During the download phase, the list of available IP Addresses is sent to the firmware.

To select the default IP Address, start the “Dialogic® Configuration Manager (DCM)” and open the “Default IP Address” tab.



Here, the DCM will display all detected IP Addresses in the multi-homed system. The selected IP Address is the default address. The system will set the first IP address to be the default, but the user can change it to be any other of the detected IP addresses.

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 = BASE_PORT + (channel * 2). The same applies for the video port that has a different base.

Note: HMP currently supports a maximum of sixteen (16) IP addresses. If there are more than 16 IP addresses in a system, you need to disable several of them from the operating system, or else the HMP will not work properly.

1.31.4 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

The information returned is the list of the available IP addresses in the system. Each returned info is in a IPM_PORT_INFO structure. 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, and do not include with any other media type.

Example

```
#define NOERROR 0
#define ERROR 1

int GetInterfaceInfo(int nDeviceHandle)
{
    IPM_MEDIA_INFO MediaInfo;
    MediaInfo.unCount = 1;
    MediaInfo.MediaData[0].eMediaType = MEDIATYPE_LOCAL_RTP_INFO_ENUM;

    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 is an example of the 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_getevttype();
            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 switch
```

```
        }// 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 int i;
    char cAddr6[48]={0};
    IPM_PORT_INFO*pInfo;

    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;

            default:
                printf("MediaType[%d]=0x%x\n", i,pMediaInfo-
                >MediaData[i].eMediaType); break;
        }//end switch
    }//end for
}
```

1.31.5 Set IP Address for Channel

To set the IP address of an interface, use the **ipm_StartMedia( )** function with the mediatype MEDIATYPE_LOCAL_RTP_INFO, and fill the structure IPM_PORT_INFO 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 initialized at Dialogic System Service start time) is provided.

Example

```
// If desired the app can now select an appropriate interface from
// the list of available ones obtained from MEDIATYPE_LOCAL_RTP_INFO_ENUM
// 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)
{
    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
    {
        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;
    }

    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.

1.32 Support for Windows Server 2012 Operating Systems

Service Update 343 introduces support for the Standard Server edition of Windows 2012 operating system. For information about installing and using the Dialogic® PowerMedia™ HMP for Windows Release 3.0, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide*.

Notes:1. Currently supported Dialogic® boards operate in Windows 2012 with no specific configuration changes, with the exception of D/4PCIUFEQ and D/4PCIU4SEQ which are not supported (refer to IPY00102428 in the Release Issues chapter).

2. Applications must be compiled as 32-bit binaries.

3. The Microsoft Intelligent Platform Management Interface (IPMI) Compatibility Device may cause audio degradation. For more details, refer to the Section 3.2.1, “Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide”, on page 358.

1.33 Increase in Concurrent Sessions on a Single Server

Service Update 343 supports up to 2,000 IP Call Control (SIP) sessions, 2,000 RTP G.711 channels, and 2,000 concurrent standard Voice resources with Windows Server 64-bit versions.

Notes:1. This feature requires either Windows Server 2008 64-bit version or Windows Server 2012. Earlier Windows versions do not support this feature.

2. Data Execution Prevention (DEP) must be set to "OptIn"; refer to the Section 3.1.1, “Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide”, on page 357 for more information on DEP considerations.

3. The maximum licensable densities are as following link; this does not mean the features can be run all simultaneously.

http://www.dialogic.com/products/media-server-software/hmp-software/hmp-density-table.aspx

1.33.1 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 then 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
- Windows Server 2008 64-bit versions

1.34 Configuring the Line Law Encoding Mode

Service Update 343 provides the ability to change the line (network interface) law encoding mode irrespective of the line type (T1 or E1) setting. The feature is valid irrespective of what protocol the line is configured with, so long as the protocol belongs to Group 1.

The feature is specific to the Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP PCIe Digital Network Interface Boards.

1.34.1 Feature Description

By default, network interface boards enforce a specific line law encoding conversion based on the line type (T1 or E1). The line type is automatically determined and set based on the protocol selected through the trunk configuration utility. Law conversion on each line bearer channel (timeslot) is then applied based on the line encoding and the Soft Bus encoding settings.

This feature introduces a new LineAdmin parameter called LineEncoding. It provides a way to set the line law encoding preference to its default (native to line type), A-law, Mu-law, or transparent (no encoding) on the line, irrespective of its line type. Once the LineEncoding parameter is set during configuration, it remains in effect until a new board initialization is performed.

Notes:1. The LineEncoding parameter and desired value must be present in the LineAdmin section(s) of the board's configuration file and the fcdgen utility invoked prior to the board initialization for it to take effect; otherwise the default line law encoding for the line type applies.

2. The idle pattern transmitted by the network interface adjusts automatically to the proper value for the line encoding based on this parameter setting.

1.34.2 Configuration Changes

The new parameter, LineEncoding, allows to override the line interface default encoding. This parameter is entered and configured through the generated CONFIG file.

The parameter has three possible values as shown below:

Value Parameter Description

Parameter Name	Value	Parameter Setting Names	Value	Description
LineEncoding	0x1627	LineEncoding_Default	0	This value keeps the default (native) encoding and idle patterns for the line type (Mu-law for T1 and A-law for E1).
		LineEncoding_Alaw	1	This value forces the encoding on the line to be of A-law type. The idle patterns on the line channels toward the network will be set to an acceptable A-law value and law conversion will be applied on each bearer channel based on this setting and the Soft Bus law setting.
		LineEncoding_Mulaw	2	This value forces the encoding on the line to be of Mu-law type. The idle patterns on the each bearer channel toward the network will be set to an acceptable Mu-law value and law conversion will be applied on each bearer channel based on this setting and the Soft Bus law setting.
		LineEncoding_Txparent	4	This value forces removal of any encoding on the line (i.e., sets it in transparent mode). The idle pattern on the each bearer channel toward the network will be set to 0x00 and no law conversion will be applied on any bearer channel, so data is passed transparently to and from the Soft Bus and for hairpinning between two lines configured as such.

1.34.3 Configuring the LineEncoding Parameter

The LineEncoding parameter 0x1627 must be added in any place after the LineType parameter in the [lineAdmin] section for each line as desired.

For example:

```
[lineAdmin.1]
...
SetParm=0x1627,0 ! LineEncoding (Default=0, Alaw=1, Mulaw=2, Txparent=4)

[lineAdmin.2]
...
SetParm=0x1627,1 ! LineEncoding (Default=0, Alaw=1, Mulaw=2, Txparent=4)
```

Notes:1. Setting the LineEncoding to an incorrect value may cause audio corruption and call setup failures on most CAS protocols due to tone corruption.

2. Setting the LineEncoding to LineEncoding_Default has the same practical effect as not having it set at all.

3. The LineEncoding parameter must not be entered after the LineType parameter in any LineAdmin section. If multiple entries exist per LineAdmin section, the last one will take effect.

4. The LineEncoding setting can be different for each LineAdmin section. Each line interface is allowed its own line law encoding setting independent of any other on the board.

5. The LineEncoding parameter does not apply to ISDN British Telecom group (DPNSS and DASS2) protocols (Group 2).

6. 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 Configuration Guide* for details on how to do this.

7. The LineEncoding parameter is only available on the PCIe Network Interface Boards indicated.

1.35 Support for Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP Digital Network Interface Boards

Service Update 338 introduces Windows 7 and Windows Server 2008 support for the Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP Digital Network Interface boards. They have eight, four, two, and one T1/E1 network interfaces respectively, in a half-length by one (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.

1.35.1 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. None of these boards will function properly, unless installed on HMP supported revisions of Windows 7 and Windows Server 2008.

2. If the board is not configured and installed properly, it will not be detected in the system.

3. The boards also require a system (chipset) and BIOS that support 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.

1.35.2 Installing the Dialogic® HMP Software

The Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP boards require the use of a Dialogic® PowerMedia™ HMP for Windows Release 3.0 Service Update 338 or later.

Note: If the HMP Software has already been installed without the Circuit Connectivity Runtime Package, follow the instructions in the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide* to properly install it, before the boards can be used in the system.

1.35.3 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® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, or DNI310TEPE2HMP; however 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® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, or DNI310TEPE2HMP 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 of these boards 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 3.0WIN Software Installation Guide*.

1.35.4 Configuring the Boards

After the Dialogic® HMP Software is installed and the appropriate licensing is obtained, you can begin the configuration process. The *Dialogic® Host Media Processing Configuration Guide* provides detailed instructions for configuring HMP Software and digital network interface boards in the system.

This section provides supplementary information that is applicable to the boards in question:

- Global Call CDP Configuration for PDK Protocols
- Guidelines for Assigning Protocols
- FCD/PCD File Names
- Bridge Configuration

1.35.4.1 Global Call CDP Configuration for PDK Protocols

Refer to the *Dialogic® Global Call CDP Configuration Guide* for complete information on CDP configuration; only supplementary information is provided here.

pdk.cfg File location

Because the boards are supported under Windows 7 and Windows Server 2008, the location of the *pdk.cfg* file is in the "cfg" subdirectory of the User-Modifiable File Destination Location (normally C:\ProgramData\Dialogic).

Note: C:\ProgramData is a hidden directory.

pdk.cfg File Examples

Note: The variant (v) value will differ depending on the CDP file you are using.

If T1 CAS PDK is being used with a Dialogic® DNI310TEPE2HMP (single span) board with logical board ID 0 and Dialogic® HMP Software board with logical board ID 1:

```
b 0  v  pdk_us_mf_io.cdp  p  gnetworkonly_hmpsshle_1_cas.pcd  f
gnetworkonly_hmpsshle_1_cas.fcd m hmp2_mixed.mlm.sym r 1
```

If T1 CAS PDK is being used with a Dialogic® DNI610TEPE2HMP (dual span) board with logical board ID 0 and Dialogic® HMP Software board with logical board ID 1:

```
b 0  v  pdk_us_mf_io.cdp  p  gnetworkonly_hmpdshle_2_cas.pcd  f
gnetworkonly_hmpdshle_2_cas.fcd m hmp2_mixed.mlm.sym r 1
```

If T1 CAS PDK is being used with a Dialogic® DNI1210TEPE2HMP (quad span) board with logical ID 0 and Dialogic® HMP Software board with logical ID 1:

```
b 0 v pdk_us_mf_io.cdp p gnetworkonly_hmpqshle_4_cas.pcd f
gnetworkonly_hmpqshle_4_cas.fcd m hmp2_mixed.mlm.sym r 1
```

If a board is mixing CAS and ISDN protocols, the *pdk.cfg* file should indicate the trunks that are configured for CAS. For example, if a Dialogic® DNI2410TEPE2HMP (octal span) board is configured to have four CAS interfaces and four 4ESS interfaces, the *pdk.cfg* file should contain:

```
b 0 line {1 2 3 4} v pdk_us_mf_io.cdp p gnetworkonly_hmposhle_4_cas_4_4ess.pcd
f gnetworkonly_hmposhle_4_cas_4_4ess.fcd m hmp2_mixed.mlm.sym r 1
```

Note: R2MF is not supported on the DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, or DNI310TEPE2HMP board.

You may assign the same protocol or different protocols to each network interface on the board, with some limitations. Refer to the Guidelines for Assigning Protocols section below.

1.35.4.2 Guidelines for Assigning Protocols

Refer to the *Dialogic® Host Media Processing Configuration Guide* for complete details; this section only provides variances for the boards in question.

The protocols supported 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 interface on the board, but all of the protocols on a given board must belong to the same group. In addition, the DNI1210TEPE2HMP, DNI610TEPE2HMP, or DNI310TEPE2HMP board and interfaces 1 - 4 on the DNI2410TEPE2HMP board must have the same line type (T1 or E1), and all of the protocols on interfaces 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 interfaces using the Trunk Configuration property sheet in the DCM.

Refer to the FCD/PCD File Names section below.

1.35.4.3 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® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, or DNI310TEPE2HMP board are initially assigned. After you select the protocols by using the Trunk Configuration property sheet in the DCM, PCD/FCD 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_hmposhle_1_net5_7_qsige1.pcd* is a generated PCD file for a DNI2410TEPE2HMP board using NET5 protocol on one interface and QSIGE1 protocol on the other seven trunks. The generated PCD file, *gnetworkonly_hmpqsbe_1_net5_3_qsige1.pcd*, would be for a DNI1210TEPE2HMP board using NET5 protocol on one interface and QSIGE1 protocol on the other three ones.

A user-configurable file with the extension CONFIG is also created. If the default settings in the FCD files are not appropriate for your configuration, you can modify the FCD file parameters using the CONFIG file and the *fcdgen* utility.

Refer to the *Dialogic® Host Media Processing Configuration Guide* for more details.

1.35.4.4 Bridge Configuration

The Dialogic® DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP 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).

Refer to the “Bridge Device Parameters” section of *Dialogic® Host Media Processing Configuration Guide* for more information.

Note: It is neither required nor recommended changing any of the Bridge Device parameters on bridges established for the DNI2410TEPE2HMP, DNI1210TEPE2HMP, DNI610TEPE2HMP, and DNI310TEPE2HMP boards.

1.36 Special SIP INVITE Request-URI Overwrite

Service Update 338 provides the ability to set Destination Address in outgoing INVITE SIP requests different than the one derived from the R-URI entry.

The resolved SIP destination address of outgoing INVITEs is derived from the Request-URI in the INVITE line. There are several methods to manipulate the Request-URI in HMP though; refer to the *Dialogic® Global Call IP Technology Guide* for details on how to do so.

Though HMP allows to overwrite the Request-URI so that to be different from the "To" field, remote address resolution is normally derived from the former.

This feature changes the Request-URI behavior in that it allows a user application to overwrite it in the outgoing INVITE after address resolution is achieved (i.e., not using the Request-URI for destination address resolution).

To take advantage of this feature, the existing set ID IPSET_CONFIG with the new parameter ID, IPPARM_RURI_OVERWRITE are passed via the **gc_SetConfigData( )** function to enable and disable this feature on a virtual board basis.

By default, this feature is disabled.

The following table shows the new parameter ID in the IPSET_CONFIG parameter set.

Parameter ID	Data Type & Size	Description	SIP/H.323
IPPARM_RURI_OVERWRITE	Type: unsigned short, Size: sizeof(unsigned short)	This parameter is used for overwriting the outgoing SIP INVITE R-URI after remote destination address resolution is complete. Possible values are: • IP_ENABLE • IP_DISABLE – Default	SIP

This new parameter should be used in conjunction with the existing IPPARM_REQUEST_URI to achieve the desired results. When the new IPPARM_RURI_OVERWRITE is used, HMP uses the value in the IPPARM_REQUEST_URI to overwrite the Request-URI in the outgoing INVITE after address resolution is achieved. This way the remote address is determined by the GC_PARM_BLKP destination address passed onto the **gc_MakeCall( )**, instead.

Note: Even though the IPPARM_SIP_HEADER is the generic and preferred way for manipulating all standard and non-standard SIP headers, given the feature's limited scope, IPPARM_REQUEST_URI is hereby supported instead.

Example

Use the **gc_SetConfigData( )** API on the board device (here bdev) to overwrite the SIP request URI for outgoing INVITE as follows:

```
   GC_PARM_BLKP ruriparmblkp = NULL;
   long t = 0;
   …

   gc_util_insert_parm_val(&ruriparmblkp,IPSET_CONFIG,IPPARM_RURI_OVERWRITE,sizeof(unsigned
short), IP_ENABLE);
  gc_SetConfigData(GCTGT_CCLIB_NETIF, bdev, ruriparmblkp,0, GCUPDATE_IMMEDIATE, &t, EV_ASYNC);


   ……
   GC_MAKECALL_BLK      *makecallblkp;
   GC_PARM_BLKP parmblkp = NULL;
   char *szRURI  = "sip:bogus@ uc-hostname.xyz:5061";
   ……

   /* set GCLIB_ADDRESS_BLK with destination string & type*/
   strcpy(gclib_makecallp->destination.address,port[index].destination_num);
   /* set the called number plan to Transparent type  */
   gclib_makecallp->destination.address_type = GCADDRTYPE_TRANSPARENT;
```

Then set the Request-URI string using the IPPARM_REQUEST_URI either in the GC_PARM_BLKP in **gc_MakeCall( )** or the **gc_SetUserConfig( )** API (for single call) as follows:

```
   gc_util_insert_parm_ref_ex(&parmblkp,IPSET_SIP_MSGINFO,IPPARM_REQUEST_URI,(unsigned
long)(strlen(szRURI)+1),szRURI);
   makecallblkp->gclib->ext_datap = parmblkp;
   ….
   if (gc_MakeCall(pline->ldev, &pline->call[callindex].crn,…… makecallblkp,…
```

Once this is done, the destination address of the GC_PARM_BLK of the **gc_MakeCall( )** will determine the "To" field in the outgoing INVITE destination address, the R-URI of the outgoing SIP INVITE will be the value set via the IPPARM_REQUEST_URI; however the call is made as resolved for the destination address specified in the destination address GC_PARM_BLKP.

To disable the feature, use the **gc_SetConfigData( )** again on the board device but with a value of IP_DISABLE as follows:

```
   gc_util_insert_parm_val(&ruriparmblkp,IPSET_CONFIG,IPPARM_RURI_OVERWRITE,sizeof(unsigned
short), IP_DISABLE);
```

1.37 MSML Server Software Support Removed

Service Update 328 removes MSML Media Server Software support. Support was added in Service Update 286. MSML support is now available under Dialogic® PowerMedia™ Extended Media Server (XMS).

1.38 Support for TCP Connection Reuse in SIP

Service Update 328 provides support for Transmission Control Protocol (TCP) Connection Reuse in SIP dialog Requests from certain User Agents (UA).

Some UAs expect TCP connection reusability by providing SIP messages headers with ephemeral ports explicitly listed, so that Requests can be sent to them on existing TCP connections instead of opening a new TCP connection. However, when an INVITE Request is received by the HMP SIP stack, the stack assumes that the TCP ports listed are configured ports (rather than ephemeral), and tries to establish a new TCP connection on this port for sending new Requests to the UA within the same dialog. While this is expected behavior, the UA has no listen capabilities on this port, and further Requests like REFER or BYE would fail.

With the introduction of this feature, when enabled at the board level, HMP will perform TCP port reusability, irrespective of the nature of the TCP ports in the incoming INVITE Request that initiated the dialog.

This feature is intended for UAs that implement TCP connection reusability, where their INVITE SIP Requests have ephemeral ports explicitly listed in headers, rather than configured ones. For example, the SIP message below has TCP transport headers with ephemeral ports explicitly listed, expecting that new Requests to the UA reuse the existing TCP connections instead of opening a new connection.

```
INVITE sip:200@192.168.0.42:5060 SIP/2.0
Via: SIP/2.0/TCP 192.168.0.62:1493;rport;branch=z9hG4bK593b5b2d0ed23d117a9a3-b63d9c5c-62a1965d
Via: SIP/2.0/UDP 192.168.0.73:61800;branch=z9hG4bK-d8754z-e9178c45a10c2655-1---d8754z-
;rport=61800
Max-Forwards: 69
Contact: <sip:1994@192.168.0.62:1493;transport=tcp>
To: "200"<sip:200@192.168.0.62>
From: "1994"<sip:1994@192.168.0.62>;tag=5d039c72
Call-ID: ZDE2ODNjNWE4YjkxMDc2ZjkyYWJjNGI0YTY0NmY5YzU.
CSeq: 2 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
```

Note: This feature only applies to the reusability of already established TCP transport connections in incoming INVITEs; it does not affect the behavior of TCP connections established on new INVITE Requests out of HMP. Neither does it affect Responses out of HMP, which always reuse the same TCP connection that was established in its corresponding incoming Request, irrespective of this feature.

1.38.1 API Implementation

To take advantage of this feature, the existing set ID IPSET_CONFIG with the new parameter ID, IPPARM_CONNECTION_REUSE, are passed via the **gc_SetConfigData( )** function to enable and disable this feature on a virtual board basis.

By default, this feature is disabled. The following table shows the new parameter ID in the IPSET_CONFIG parameter set.

Parameter ID	Data Type & Size	Description	SIP/H.323
IPPARM_CONNECTION_REUSE	Type: int Size: sizeof(int)	This parameter is used for enabling or disabling TSP Connection Reuse of SIP signaling messages. Possible values are: • IP_ENABLE • IP_DISABLE – Default	SIP only

Example

```
parmblkp = NULL;

gc_util_insert_parm_val(&parmblkp, IPSET_CONFIG,
                IPPARM_CONNECTION_REUSE, sizeof(int), IP_ENABLE);

if (gc_SetConfigData(GCTGT_CCLIB_NETIF, s_gcBrdDev, parmblkp, 0 /*timeout*/,
                     GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC) != GC_SUCCESS)
{

        sprintf(str, "gc_SetConfigData(linedev=%ld) Failed
        configuring PPARM_CONNECTION_REUSE mode", s_gcBrdDev);

        printandlog(index, GC_APIERR, NULL, str, 0);

        // handle error…
}

sprintf(str, "gc_SetConfigData(linedev=%ld) Success - IPPARM_CONNECTION_REUSE
enabled", s_gcBrdDev);

printandlog(index, GC_APICALL, NULL, str, 0);

gc_util_delete_parm_blk(parmblkp);
```

1.38.2 Use Case Scenario

The following scenario shows the relevant traces when the feature is enabled. Here, the TCP transport ephemeral port in the incoming INVITE is 3637, connected to the well-known (configured) port 5060. The Contact and the Record-Route headers *wrongly* specify to use port 3637 for subsequent Requests.

```
Transmission Control Protocol, Src Port: 3637 (3637), Dst Port: 5060 (5060), Seq: 1, Ack: 1,
Len: 1001
INVITE sip:joe@10.130.1.53 SIP/2.0
Via: SIP/2.0/TCP 10.130.1.40:3637;rport;branch=z9hG4bK2442f5a0104e501ef9f99-b38b2698-d2239ac0
Via: SIP/2.0/UDP 10.130.1.29:5060;branch=z9hG4bK-d8754z-ec7a9f66162ebc68-1---d8754z-;rport=5060
Max-Forwards: 69
Contact: <sip:bob@10.130.1.40:3637;transport=tcp>
To: "joe" <sip:joe@10.130.1.40>
From: "bob" <sip:bob@10.130.1.40:3637>;tag=f6452c02
Call-ID: NzNmYTlhY2VjYWU3ZDZhNmM1MTRlNzJhODRjZTIwNjA.
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE,
SUBSCRIBE, INFO
User-Agent: Dialogic release 1102u stamp 52345
Record-Route: <sip:10.130.1.40:3637;transport=tcp;lr>
Content-Type: application/sdp
Content-Length: 308
...
```

The Response to the INVITE Request is sent correctly by HMP over the same TCP connection used to receive the Request.

For example:

```
Transmission Control Protocol, Src Port: 5060 (5060), Dst Port: 3637 (3637), Seq: 1, Ack: 1002,
Len: 804
SIP/2.0 200 OK
From: "bob"<sip:bob@10.130.1.40:3637>;tag=f6452c02
To: "joe"<sip:joe@10.130.1.40>;tag=2a0d718-0-13c4-50022-1419c-7d0ab13f-1419c
Call-ID: NzNmYTlhY2VjYWU3ZDZhNmM1MTRlNzJhODRjZTIwNjA.
CSeq: 1 INVITE
Via: SIP/2.0/TCP 10.130.1.40:3637;alias;rport;branch=z9hG4bK2442f5a0104e501ef9f99-
b38b2698-d2239ac0
Via: SIP/2.0/UDP 10.130.1.29:5060;rport=5060;branch=z9hG4bK-d8754z-ec7a9f66162ebc68-1---d8754z-
Supported: replaces
Record-Route: <sip:10.130.1.40:3637;transport=tcp;lr>
Contact: <sip:joe@10.130.1.53>
Allow: INVITE, CANCEL, ACK, BYE, OPTIONS, INFO, REFER, NOTIFY
Content-Type: application/sdp
Content-Length: 164
...
```

The BYE Request sent by HMP is now using the same TCP connection that was used for the incoming INVITE Request. For example:

```
Transmission Control Protocol, Src Port: 5060 (5060), Dst Port: 3637 (3637), Seq: 805, Ack:
1619, Len: 515
BYE sip:bob@10.130.1.40:3637;transport=tcp SIP/2.0
From: "joe"<sip:joe@10.130.1.40>;tag=2a0d718-0-13c4-50022-1419c-7d0ab13f-1419c
To: "bob"<sip:bob@10.130.1.40:3637>;tag=f6452c02
Call-ID: NzNmYTlhY2VjYWU3ZDZhNmM1MTRlNzJhODRjZTIwNjA.
CSeq: 1 BYE
Via: SIP/2.0/TCP 10.130.1.53:5060;branch=z9hG4bK-141a1-4e85ece-3753d3cd
Max-Forwards: 70
Supported: replaces
Route: <sip:10.130.1.40:3637;transport=tcp;lr>
Allow: INVITE, CANCEL, ACK, BYE, OPTIONS, INFO, REFER, NOTIFY
Allow-Events: refer
Content-Length: 0
```

1.39 Support for SIP 1xx Provisional Responses in a Ringing State

With Service Update 320, HMP can accept SIP 1xx provisional responses from the UAS and use them to reset the call timeout to prevent disconnecting the call while 1xx responses are being received. With this change, the application no longer receives a call termination after the initial connection timeout.

A SIP provisional response is any message between 101 and 199 to show that a call is progressing. Currently, an outbound call from HMP is terminated after a default period of three (3) minutes. According to RFC 3261, a call should only be terminated after a specified timeout period if no provisional responses are sent by the UAS.

With this modification, every time a provisional response is received before connection, such as receipt of a 200 OK message, the connection timer will restart. The application no longer receives call termination after the initial connection timeout, and is able to receive all provisional response messages after the **gc_MakeCall( )** function is called.

1.40 SIP Semi-Attended Call Transfer Support

Service Update 320 adds support for a SIP semi-attended call transfer. With this feature, an attended transfer can be initiated with the consultation call leg in the Alerting state (GCST_ALERTING). For more information about call transfers and related terminology, refer to the “Call Transfer Scenarios When Using SIP” section in *Dialogic Global Call IP for Host Media Processing Technology Guide*.

After the dialog is established, the Private Automatic Branch Exchange (PABX) will proceed with the transfer to the Transfer target even if the Transferor, in this case HMP, abandons the call leg.

1.40.1 Feature Description

Call transfers are performed using the Global Call API **gc_InvokeXfer( )** function. Prior to this feature, **gc_InvokeXfer( )** was able to be passed to up to two Call Reference Numbers (CRNs). When only one CRN was passed (the other one being NULL), the call transfer was unattended or in blind mode. When two CRNs were passed, the call transfer was in attended or supervised mode. In attended mode, the **gc_InvokeXfer( )** function requires that both CRNs are in GCST_CONNECTED state. If not, the function returns an error. This is because attended mode implies that the Transferor is connected to both parties (Transferee and Transfer target) before sending the REFER request.

With this feature, the second CRN can be in the ALERTING state when calling the **gc_InvokeXfer( )**. In this scenario, a REFER request with its “replaces” header is sent to the Transferee as soon as a provisional response is received, which means that the Transfer Target (either a human or his voice mail) is available.

Some PABXs support this functionality. The main use case is for a Transferor to initiate a consultation call as it would do with a standard Attended transfer; however in some circumstances, the Transferor may wish to disengage from the transfer before the session to the Transfer target is established. The Transferor may do this it knows that the Transferee will end up eventually talking to the Transfer target’s voicemail.

The aforementioned consultation call initiated by HMP can be abandoned by the application via the **gc_DropCall( )** function once the call is accepted by the PABX (for example, a 202 response to HMP’s REFER method with Replaces header). This action sends a CANCEL request toward the PABX.

If the PABX, acting as the Transferor's User Agent (UA) supports this feature, then it will proceed with the transfer as if it were an attended transfer, even after the Transferor's (HMP) CANCEL request.

Note: HMP is able to initiate a semi-attended transfer; however, it is the responsibility and functionality of the PABX, or Transferor UA, to complete the transfer accordingly.

1.40.2 Semi-attended Call Transfer Use Case

The use case below shows the steps needed to perform a successful semi-attended call transfer after the application has an active SIP call connected to the Transferee.

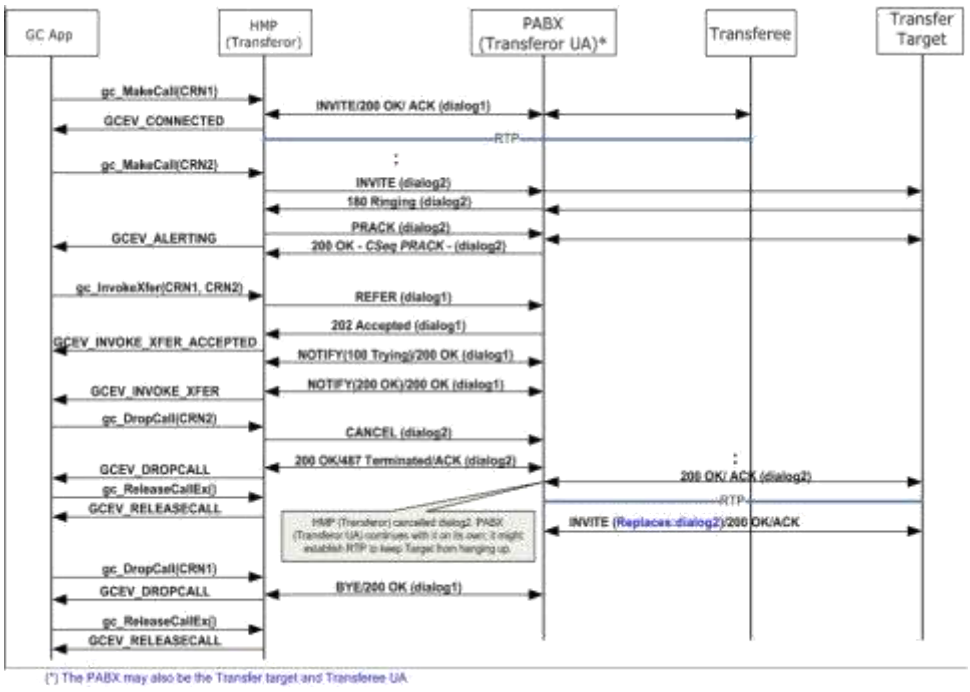
1. Call the **gc_MakeCall( )** function to send the INVITE to a Transfer target.

2. As soon as the GCEV_ALERTING event is received to indicate that the Transfer target is available, the gc_InvokeXfer( ) function is called by the application to initiate the call transfer.

3. The SIP Stack sends the REFER message to the Transferee.

4. The application disconnects (CANCEL) the pending consultation call to the Transfer target and continues as if it would had this been an Attended transfer.

5. The PABX performs the semi-attended transfer connecting the Transferee and Transfer UAs.



Notes:1. The above diagram shows a possible SIP semi-attended transfer scenario; however, there may be many possible scenarios depending on the PABX.

2. PRACKs are not required for proper feature operation, and are only shown here as an example.

1.41 Increased G.729 Density

With Service Update 317, Dialogic® HMP Software supports 1500 sessions of G.729. Refer to the Dialogic Product Center for additional information.

1.42 Overlap Send and Receive SIP Signaling Support

Service Update 317 adds SIP overlap send and receive signaling as per RFC 3578. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide* and the *Dialogic® Global Call API Programming Guide*.

1.43 VMware ESXi Virtualization Support Enhancements

Service Update 314 enhances Dialogic® HMP software support in a VMware ESXi virtual machine. This support applies to VMware ESXi versions 4.1 through 5.5, and the following of Windows guest operating systems:

- Windows Server 2008 R2 with Service Pack 1
- Windows Server 2008 with Service Pack 2 (32-bit and 64-bit versions)
- Windows 7 with Service Pack 1 (32-bit and 64-bit versions)
- Windows Server 2003 (Standard or Enterprise Edition) with Service Pack 2 (32-bit version)
- Windows Server 2003 R2 (Standard or Enterprise Edition) with Service Pack 2 (32-bit version)
- Windows Server 2012 R2 (Support added in September 2014)

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.

This feature enhances initial virtualization support introduced in Service Update 275 and found in Section 1.68, “Virtualization Support”, on page 158. The following information replaces the contents of that section.

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. For more information about virtualization, refer to the VMware web site at *www.vmware.com*.

Dialogic® HMP software can be installed in a VMWare ESXi virtual machine running a supported Windows guest operating system listed above. Each HMP instance running in a virtual machine requires a separate runtime license.

Service Update 314 was tested under the following conditions.

- The test configuration for this release included two Windows Server 2008 R2 64-bit virtual machines running on an ESXi 5.5 host.
- Each virtual machine hosted 240 ports of G.711 in an IVR test configuration.
- One of the virtual machines included 240 CSP resources.

Note: A customer may experience different or enhanced results dependent on many factors, including processor speed, memory foot print and I/O system speeds. These numbers are provided for comparison information only.

The density achieved when operating in a 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 vSphere ESXi products.

More information on VMWare vSphere and ESXi products can be found at:

http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html

1.43.1 Performance Tuning Guidelines

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.

Refer to the following VMWare vSphere ESXi documents for a thorough explanation of the terms and concepts utilized herein:

For ESXi 4.1

- vSphere Resource Management, ESX 4.1, ESXi 4.1, vCenter Server 4.1: *http://www.vmware.com/pdf/vsphere4/r41/vsp_41_resource_mgmt.pdf*
- Performance Troubleshooting for VMware vSphere 4.1: *http://communities.vmware.com/docs/DOC-14905/vsphere41-performance-troubleshooting.pdf*

For ESXi 5.0

- vSphere Resource Management, ESXi 5.0, vCenter Server 5.0: *http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-resource-management-guide.pdf*
- Performance Best Practices for VMware vSphere 5.0, VMware ESXi 5.0, vCenter Server 5.0: *http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.0.pdf*

The optimal settings for each situation will vary. The following subsections are guidelines that apply in most cases. Always refer to the VMWare documentation for the most up-to-date information.

CPU Affinity Settings

To run latency sensitive software on VMware ESXi, CPU affinity may be used. Isolating your virtual machines from one another using CPU affinity may improve performance of 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.

If using CPU affinity, VMWare recommends you add one additional physical CPU in the affinity setting to allow at least one of the virtual machine's hypervisor support threads to be scheduled at the same time as its virtual CPUs. For example, set a two-way SMP virtual machine with affinity to at least three physical CPUs.

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 per virtual machine to the VMware ESXi 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, 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 for your specific version of VMWare VSphere ESXi.

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 affined 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.
- Manual CPU affinity settings do not always provide optimal performance in all cases. Refer to the VMWare documentation referenced above for complete information.
- 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.

Resource Budgeting

The same Dialogic® PowerMedia™ HMP for Windows Release 3.0 requirements for system resources are required when operating in a VMware ESXi environment.

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. Refer to VMware network performance tuning documentation for your specific version of vSphere ESXi.

1.43.2 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.

1.44 SIP "To tag" for Inbound Calls

Service Update 314 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.

For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.45 IPMI Updates

Note with Service Update 313, the following issue with Intelligent Platform Management Interface (IPMI) updates:

On certain servers, performance impact has been seen observed as delays in execution of Deferred Procedure Calls (DPC) used by Dialogic software. This impacts the performance of a real time telephony system and may result in audio quality issues and, in some extreme cases, hung ports. If using IPMI, it is recommended to upgrade to the latest available versions of the server manufacturer's firmware which corrects the latency issues.

1.46 Anti-virus Software Policy

Service Update 313 announces Dialogic's general policy regarding third-party anti-virus software. Dialogic understands and acknowledges the desire for customers / end users to install anti-virus software in their environment and is providing this policy statement for guidance in this area. Loading and running any third-party anti-virus software on Dialogic® HMP-based servers, regardless of the operating system (Linux or Windows), may compromise the performance of a real time telephony system. Consequently, it is recommended that if such anti-virus software is required by the customer for security purposes, it should be configured to run during periods of minimal to no-call traffic (i.e., system inactivity) to minimize the risk of interference or disruption.

If a problem requiring diagnosis occurs on a server running anti-virus software, and the antivirus software is suspected of causing or contributing to the problem, Dialogic Technical Support will likely ask the customer to remove or disable the anti-virus software before proceeding with further diagnosis. Dialogic does not validate any particular anti-virus package nor does it endorse the use of a particular third-party anti-virus software vendor or product.

Note: For 64-bit Windows 7, Windows Vista, and Windows Server 2008 operating systems using anti-virus programs, it is recommended that users exclude Dialogic subdirectories and also any application sub-directories from the system.

1.47 User Account Control Recommendation

With Service Update 313, it is recommended that users of 64-bit Windows 7, Windows Vista, and Windows Server 2008 operating systems disable Microsoft's User Account Control because it has been shown to cause intermittent audio gaps when streaming audio.

1.48 Increase in Concurrent Sessions on a Single Server

Service Update 310 increases HMP density support for certain combined IP (RTP) sessions), Voice, Conferencing, and IP Call Control (SIP) sessions.

1.48.1 Feature Description

With this feature, HMP supports up to 1500 RTP channels, up to 375 enhanced RTP channels,1125 concurrent standard Voice, approximately 168 concurrent conferencing resources, and 1500 concurrent IP Call Control (SIP) sessions.

Note: This feature requires either the Windows Server 2008 or Windows 7 operating system, in 32 or 64 bit versions. Earlier Windows operating systems do not support this feature.

1.48.1.1 CPU and Memory Considerations

Higher channel densities require greater CPU performance for the system. Thus, to reach the required densities, a high-end, multi-processor server is required. In general, server multi-processor performance is specified as a combination of CPU speed, processor type, number of processors, and number of cores per processor.

This feature requires the following minimum server considerations for the higher densities specified, based on Intel Xeon 56xx, and assuming about 30 percent CPU is left for the application:

Processor Type	RTP/ Voice	Enhanced RTP	Conferencing	EC	Minimum CPU Speed
Intel Dual Quad Xeon 56xx	1000	500	150	No	2000 MHz
Intel Dual Hex Xeon 56xx	1000	500	150	No	1700 MHz
Intel Dual Hex Xeon 56xx	1500	500	225	No	2700 MHz

The Dialogic® Product Center, located at *https://prodcenter.dialogic.com*, offers a calculator tool to aid in density and target server system verification based on the desired license characteristics.

1.49 New TDX_DRVNOMEM Event

Service Update 310 adds an unsolicited event, TDX_DRVNOMEM, to the Voice API library. This event is added to address IPY00093815 (refer to the Release Issues chapter). Now, the Voice library will return the TDX_DRVNOMEM event if the underlying device driver is unable to allocate physical memory. Applications must link to the Voice library to be able to receive the TDX_DRVNOMEM event. This event is for information only and the condition is often “terminal.” Restarting the system may be required to restore it to normal operation.

Note: Applications will not see this event during normal operations.

1.50 PDK Support for Automatic Answer and Reject of Inbound Calls

With Service Update 307, the Protocol Development Kit (PDK) is extended to support automatic answer and reject of inbound calls.

1.50.1 Feature Implementation

With this enhancement to the PDK upon enablement through Global Call, when the **gc_DropCall( )** function is called before **gc_AnswerCall( )**, the protocol will answer and then immediately hang-up the call. This behavior enacts a full disconnect, complete with the sending of a proper signal to the switch to abandon the call.

To accomplish this, the CDP_Forced_Release_Enabled parameter is added to the respective Country Dependent Parameter (.cdp) files. This parameter controls the behavior of the protocol when **gc_DropCall( )** is called before a call is connected and allows to enable and disable this new functionality.

Note: In order to disconnect the call in this fashion, the PDK will connect the call and immediately disconnect. However, you may wish to consult your service provided to see if unexpected charges or usage may apply for this use case prior to implementation of this feature.

1.50.2 Protocol Variants

The functionality of the CDP_Forced_Release_Enabled parameter is added to the following protocols:

Protocol	Variant File
MELCAS Lineside Bidirectional	pdk_sw_e1_mcls_io.cdp
Nortel Meridian Lineside E1 Bidirectional	pdk_sw_e1_ntmd_io
United States T1 FXS/LS Bidirectional	pdk_us_ls_fxs_io
E1 CAS Bidirectional	pdk_us_mf_io
India R2 Bidirectional	pdk_in_r2_io
Argentina R2 Bidirectional	pdk_ar_r2_io
Australia R2 Bidirectional	pdk_au_r2_io.cdp
Brazil R2 Bidirectional	pdk_br_r2_io.cdp

CDP_Forced_Release_Enabled

Enable the protocol to support "forced release" of incoming calls from the offered or accepted state. The support for forcing release of incoming calls is supported under this implementation for flexibility with Global Call applications which are permitted to call **gc_DropCall( )** from the Offered or Accepted state. In these states, the call will be answered transparently without notification to the application and then immediately disconnected (i.e., a "forced release" of the line). Note that in doing this, additional implications external to the PDK and Global Call might exist and should be considered, for instance call billing.

Values:

- 0 = Does not support forced release. No implicit answer will be performed transparently in this scenario, and only a PDK hang-up signal will be generated. (Default)
- 1 = Supports forced release. Calls are answered and then dropped immediately.

Notes:1. Not all protocols are supported on all boards in the Dialogic® DNI series.

2. The following protocols only apply to the Dialogic® DNI/601TEPHMPW board: India R2 Bidirectional
Argentina R2 Bidirectional
Australia R2 Bidirectional
Brazil R2 Bidirectional

Refer to the *Dialogic® Global Call CDP Configuration Guide* for more information.

1.51 Service URN Support

Service Update 303 supports SIP Uniform Resource Names (URNs) to be used when contacting a Service Namespace Identifier.

Note: Service in this context refers to URNs for Emergency and other well-known services.

For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.52 Dialogic® D80PCIE-LS 32-bit Compatibility Support

Service Update 303 introduces support for the Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports 32-bit compatibility mode on the 64-bit versions of Windows 7 and Windows Server 2008 R2 for the Dialogic® D80PCIE-LS board. Refer to the 64-bit Operating System Support in 32-bit Compatibility Mode section for additional information.

1.53 Support for Initial Silence Termination

With Service Update 303, 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 Preserving Data in User Configuration Files in the *Dialogic® Host Media Processing 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.

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).

1.54 Upgrading Intel Network Adapter Drivers on Windows Server 2008 and Windows 7

Service Update 299 recommends upgrading device drivers for Intel network adaptors to the latest version available. Doing so will avoid unexpected dropped or duplicated RTP packets, which would be observed as, for instance, DTMF detection failures, missing termination on silence conditions, or other types of media failures.

Driver packages for all the Intel adaptors are available at:

http://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=18720

For example, at the time of this writing, the following Windows 7 device driver for the Intel 82575EB Gigabit Network Adaptor provided acceptable RTP performance:

Current version Date: 9/29/2010 Version: 11.7.32.0

Previous versions may cause the aforementioned side effects.

1.55 Increase in SUBSCRIBE and OPTIONS SIP Requests

With Service Update 299, the number of concurrent SUBSCRIBE and OPTIONS SIP requests HMP can handle concurrently is now 256. Previously, 50 was the maximum. Refer to IPY00056119 in the Release Issues chapter.

1.56 SIP UPDATE Post-Connection Support

With Service Update 299, HMP supports handling SIP UPDATE post-connection requests with limited response capabilities in third-party call control (3PCC) mode. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.57 64-bit Operating System Support in 32-bit Compatibility Mode

Service Update 298 introduces support for the 64-bit versions of Windows 7, Windows Server 2008 (with Service Pack 2), and Windows Server 2008 R2 operating systems in 32-bit compatibility.

Notes:1. Only Dialogic® DNI Boards are supported in this mode in Service Update 298.

2. Refer to IPY00093028 in the Release Issues chapter for information about a video problem.

Although recompilation of Dialogic® HMP 32-bit applications to run on the 64-bit version of Windows operating systems is not required, the following two application development scenarios are available:

Scenario 1. Transfer (or install) your 32-bit application to a 64-bit Windows machine.

Transfer (or install) your 32-bit application binaries to a 64-bit Windows 7 or higher machine with this Dialogic® PowerMedia™ HMP for Windows Release 3.0 installed, and simply run your application. The application should run the same as it would on the equivalent 32-bit Windows operating systems provided that it does not have any other 32/64-bit dependencies outside of HMP.

Scenario 2. Build the HMP application on 64-bit Windows operating systems.

Install a supported version of Visual Studio on the 64-bit Windows operating system, open your existing (unchanged) project file, and compile (build) it. The resulting application should be equivalent to the one created in scenario 1 above. Note that the target platform set in the Visual Studio configuration manager must still remain as Win32 (32-bit). Setting the platform to x64 is not currently supported. The currently support compiler versions are:

- Microsoft Visual Studio 2005 (all editions) includes C++ compiler version 8.0. Customer applications with Windows 7 or Windows Server 2008 should use version 8. Applications must be compiled as 32-bit binaries.
- Microsoft Visual Studio 2005 with Visual Studio 2005 Service Pack 1 and Visual Studio 2005 Service Pack1 Update for Windows Vista. This compiler is required for application development under Windows 7 or Windows Server 2008.

Note: On 64-bit Windows systems, the Dialogic® PowerMedia™ HMP for Windows Release 3.0 is installed in the “Program Files (x86)” folder instead of the usual “Program Files” folder. If your application has any path dependencies on HMP components, it will need to be adjusted.

1.58 Session Timer Support SDP in ReINVITE

Service Update 297 supports adding SDP data inside a session timer reINVITE message. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.59 Improvement to Call Progress Analysis on Dialogic® HMP Thin Blades

Service Update 297 implements an improvement to Call Progress Analysis on Dialogic® Digital Network Interface (DNI) boards (thin blades).

1.59.1 Implementation

With this improvement, using Global Call and the **gc_Makecall( )** function to initiate a PSTN call that results in either a busy or fast busy/reorder tone will return a GCEV_DISCONNECT event. The **gc_ResultInfo( )** function, which provides more information about the event, will return either a GCRV_BUSY (busy tone detected) or a GCRV_CONGESTION (fast busy/reorder tone detected) cause value.

Previously, Global Call did not distinguish between a busy tone or a fast busy/reorder tone when performing call progress analysis under Global Call and would report Destination Busy for either tone.

Note: This improvement applies only to CAS and R2MF protocols.

For more information, refer to the *Dialogic® Voice API Programmer's Guide*, the *Dialogic® Global Call API Library Reference*, and the *Dialogic® Global Call API Programming Guide*.

1.60 Support for Dialogic® D/4PCIU Boards

Service Update 291 adds support for the Dialogic® D/4PCIUFEQ and Dialogic® D/4PCIU4SEQ Springware boards.These boards are analog loop start, RoHS commercial product 6/6 boards, used for developing advanced communications applications extending the HMP media capabilities to the analog loop-start PSTN network.

The Dialogic® D/4PCIUFEQ board provides four-port basic voice processing and DSP-based Group 3 fax support (DSP fax or SoftFax). The Dialogic® D/4PCIU4SEQ board provides four-port basic voice processing with continuous speech processing (CSP). Both are half-length PCI Express form factor boards. The boards have a subset of the features and functionality of the current analog Dialogic® JCT boards. They also provide HMP media streaming between their four analog interfaces and HMP; however, they have no CT Bus connectivity.

Refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Release Guide* for more information about using these Dialogic® Springware boards with Dialogic® HMP software. For information about the features of the Dialogic® JCT boards, go to:

http://www.dialogic.com/products/media/jct/default.htm

Notes:1. The Dialogic® D/4PCIU board will not function with any other board in the system, and must be the only board in the system. This is a permanent, stand-alone restriction.

2. There are special considerations for these boards because they do not have a CT Bus. For important restrictions and considerations, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Release Guide*.

3. When installing either the Dialogic® D/4PCIUFEQ or Dialogic® D/4PCIU4SEQ board, be sure to refer to the Installation Guide (*Dialogic® Quick Install Card*) that is provided with each board for important guidelines for selecting the slot where a board can be installed, and information about interconnection to the analog PSTN network or a PBX.

1.61 Increase in Fax Channels Support

With Service Update 291, Dialogic® HMP Software supports 500 channels of fax. This feature applies to both T.38 fax and V.17 PCM fax. For information about using T.38 fax, refer to the *Dialogic® Global Call IP Technology Guide*. Refer to the *Dialogic® Fax Software Reference* for information about using V.17 PCM fax.

Note: For 500 channels, a minimum of an eight-core 3 GHz system is required.

1.62 Increase in CSP Channels

With Service Update 289, Dialogic® HMP Software supports 500 channels of continuous speech processing (CSP). Refer to the *Dialogic® Continuous Speech Processing API Library Reference* and the *Dialogic® Continuous Speech Processing API Programming Guide* for information about using CSP.

1.63 MSML Server Software Support

Service Update 286 adds MSML Media Server Software support with Windows 7 and Windows Server 2008 operating systems. Refer to the *Dialogic® MSML Media Server Software User's Guide* for information about using MSML.

1.64 Monitor Mode Support for HMP Conferencing

With Service Update 286, 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.

1.64.1 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 (e.g., 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.

1.64.2 Changes to the Conferencing API Library

To implement this feature, three new API functions cnf_Listen( ), cnf_UnListen( ) and cnf_GetXmitSlot( ) are introduced. The data structure, SC_TSINFO, is used with the functions to provide time slot information.

Note: It is recommended that the two types of connection methods, dev_Connect( )/dev_Disconnect( ) and cnf_listen( )/cnf_unlisten( ), not be used simultaneously. If they are, then the application must take extreme caution to insure that the connections are properly managed.

cnf_Listen( )

Name:	int cnf_Listen(a_PtyHandle, a_pTimeslotInfo, a_pUserInfo)	
Inputs:	SRL_DEVICE_HANDLE a_PtyHandle	• valid party handle
	SC_TSINFO *a_pTimeslotInfo	• pointer to TDM bus time slot information structure
	void * a_pUserInfo	• pointer to user-define data
Returns:	CNF_SUCCESS on success CNF_ERROR on error	
Includes:	srllib.h cnflib.h	
Category:	TDM routing	
Mode:	Asynchronous	

■ Description

The cnf_Listen( ) function connects a party receive channel to a TDM bus time slot, 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 party transmit channel.

The cnf_Listen( ) function returns immediately before the operation is completed with CNF_SUCCESS, providing argument validation passed, otherwise a CNF_ERROR is returned. After the operation completes, a notification event is received and, if successful, the party receive channel is connected to the TDM bus time slot. Although multiple party channels may listen (be connected) to the same TDM bus time slot, the receive channel of a given party device can connect to only one TDM bus time slot.

Parameter	Description
a_PtyHandle	specifies a party device handle obtained from a previous open
a_pTimeslotInfo	specifies a pointer to the SC_TSINFO structure
a_pUserInfo	specifies a pointer to user-defined data

Termination Events

The termination events for this function are:

CNFEV_LISTEN
Indicates successful completion of this function at which point the party device's receive channel is connected to the TDM bus time slot originally specified in a_pTimeslotInfo.

CNFEV_LISTEN_FAIL
Indicates indicates the function failed.

Cautions

This function fails when an invalid party handle is specified or when an invalid TDM bus time slot number is specified.

Errors

The CNF_ERROR_INFO data structure provides error information for the device handle when an API function fails. Upon failure the user should call **cnf_GetErrorInfo( )**, then error info for the device will be returned in the CNF_ERROR_INFO structure. Error codes are returned as: ECNF_xxxxx

Possible errors for this function include:

ECNF_INVALID_DEVICE
invalid device handle

ECNF_SUBSYSTEM internal
subsystem error

ECNF_INVALID_PARM
invalid parameter

The CNF_EVENT_INFO data structure provides event information for the device handle when a notification event is enabled or disabled. This structure is used by and enabled by the **cnf_EnableEvents( )** function.Use **sr_waitevt( )**, **sr_enbhdlr( )** or other SRL functions to collect an event code, depending on the programming model in use. For more information, refer to the *Dialogic® Standard Runtime Library API Library Reference*. Error codes returned as: CNFEV_xxxxxx

Possible errors for this function include:

CNFEV_INVALID_DEVICE
invalid device handle

CNFEV_SUBSYSTEM
internal subsystem error

CNFEV_INVALID_PARM
invalid parameter

Example

```
#include <stdlib.h>
#include <stdio.h>

// Dialogic   headers
#include   "srllib.h"
#include  "dxxxlib.h"
#include "cnflib.h"

#define MAX_DEVNAME100
#define SRWAITTIMEOUT 10000

long ProcessEvt();

int main(int argc, char* argv[])
{
    char cnfbdname[MAX_DEVNAME] = {"cnfB1"};
    char dxdevname[MAX_DEVNAME] = {"dxxxB1C1"};
    long devh = -1;
    SRL_DEVICE_HANDLE cnfbdh = -1;
    SRL_DEVICE_HANDLE cnfh = -1;
    SRL_DEVICE_HANDLE cnfptyh = -1;
    long ts;
    SC_TSINFO scts;
    int mode = SR_POLLMODE;

    /* Set SRL to run in polled (non-signal) mode */
    if( sr_setparm( SRL_DEVICE, SR_MODEID, &mode ) == -1 )
    {
        printf( "Error: cannot set srl mode\n" );
        exit( 1 );
    }

    cnfbdh = cnf_OpenEx(cnfbdname, NULL, NULL, EV_SYNC);
    if (cnfbdh == -1)
    {
        printf("Error during call to cnf_OpenEx\n");
        /* perform error processing */
        exit(1);
    }

    /* open conferences */
    cnfh = cnf_OpenConference(cnfbdh, NULL, NULL, NULL);
    if (cnfh == -1)
    {
        printf("Error during call to cnf_OpenConference\n");
        /* perform error processing */
        exit(1);
    }

    if(sr_waitevt(SRWAITTIMEOUT) != -1)
    {
        if (!ProcessEvt())
        {
            /* perform error processing */
            exit(1);
        }
    }
    else
    {
        printf("Error during call to sr_waitevt\n");
        /* perform error processing */
        exit(1);
    }

    cnfptyh = cnf_OpenParty(cnfbdh, NULL, NULL, NULL);
```

```
if (cnfptyh == -1)
{
    printf("Error during call to cnf_OpenParty\n");
    /* perform error processing */
    exit(1);
}

if(sr_waitevt(SRWAITTIMEOUT) != -1)
{
    if (!ProcessEvt())
    {
        /* perform error processing */
        exit(1);
    }
}
else
{
    printf("Error during call to sr_waitevt\n");
    /* perform error processing */
    exit(1);
}

/* open a voice device */
devh = dx_open(dxdevname, 0);
if (devh == -1)
{
    printf("Error during call to dx_open\n");
    /* perform error processing */
    exit(1);
}

scts.sc_numts = 1;
scts.sc_tsarrayp = &ts;

if (dx_getxmitslot(devh, &scts) == -1)
{
    printf("Error during call to dx_getxmitslot\n");
    /* perform error processing */
    exit(1);
}

printf("Voice device %s (devh=%ld) is transmitting on
       %ld\n", ATDV_NAMEP(devh), devh, ts);

if (cnf_Listen(cnfptyh, &scts, NULL) == -1)
{
    printf("Error during call to cnf_Listen\n");
    /* perform error processing */
    exit(1);
}

printf("Successful call to cnf_Listen\n");

if(sr_waitevt(SRWAITTIMEOUT) != -1)
{
    if (!ProcessEvt())
    {
        /* perform error processing */
        exit(1);
    }
}
else
{
    printf("Error during call to sr_waitevt\n");
    /* perform error processing */
    exit(1);
}
```

```
    if (cnf_UnListen(cnfptyh, NULL) == -1)
    {
        printf("Error during call to cnf_UnListen\n");
        /* perform error processing */
        exit(1);
    }

    printf("Successful call to cnf_UnListen\n");

    if(sr_waitevt(SRWAITTIMEOUT) != -1)
    {
        if (!ProcessEvt())
        {
            /* perform error processing */
            exit(1);
        }
    }
    else
    {
        printf("Error during call to sr_waitevt\n");
        /* perform error processing */
        exit(1);
    }

    return 0;
}

long ProcessEvt()
{
    long ret = 1;
    int devh;
    int evttype;
    long evtlen;
    void* datap;

    printf("ProcessEvt()\n");

    devh = sr_getevtdev();
    evttype = sr_getevttype();
    evtlen = sr_getevtlen();
    datap = sr_getevtdatap();

    switch(evttype)
    {
    case CNFEV_OPEN_CONF:
        printf("Received CNFEV_OPEN_CONF\n");
        break;

    case CNFEV_OPEN_CONF_FAIL:
        printf("Received CNFEV_OPEN_CONF_FAIL\n");
        ret = 0;
        break;

    case CNFEV_OPEN_PARTY:
        printf("Received CNFEV_OPEN_PARTY\n");
        break;

    case CNFEV_OPEN_PARTY_FAIL:
        printf("Received CNFEV_OPEN_PARTY_FAIL\n");
        ret = 0;
        break;

    case CNFEV_LISTEN:
        printf("Received CNFEV_LISTEN\n");
        break;
```

```
    case CNFEV_LISTEN_FAIL:
        printf("Received CNFEV_LISTEN_FAIL\n");
        ret = 0;
        break;

    case CNFEV_UNLISTEN:
        printf("Received CNFEV_UNLISTEN\n");
        break;

    case CNFEV_UNLISTEN_FAIL:
        printf("Received CNFEV_UNLISTEN_FAIL\n");
        ret = 0;
        break;

    default:
        printf("Unhandled event: devh(%d); evttype(0x%x)", devh,
        evttype); break;
    }

    return ret;
}
```

See Also

- cnf_GetXmitSlot( )
- cnf_UnListen( )

cnf_UnListen( )

Name:	int cnf_UnListen(a_PtyHandle, a_pUserInfo)	
Inputs:	SRL_DEVICE_HANDLE a_PtyHandle	• valid party handle
	void * a_pUserInfo	• pointer to user-define data
Returns:	CNF_SUCCESS on success CNF_ERROR on error	
Includes:	srllib.h cnflib.h	
Category:	TDM routing	
Mode:	Asynchronous	

Description

The cnf_UnListen( ) function disconnects the conference party receive channel from the TDM bus. The function returns immediately before the operation is completes with CNF_SUCCESS, providing argument validation passed, otherwise a CNF_ERROR is returned. After the operation completes, a notification event is received and, if successful, the party receive channel is disconnected from the TDM bus time slot.

Calling the cnf_Listen( ) function to connect to a different TDM bus time slot automatically breaks an existing connection. Thus, when changing connections, there is no need to call the cnf_UnListen( ) function first.

Parameter	Description
a_PtyHandle	specifies a party device handle obtained from a previous open
a_pUserInfo	specifies a pointer to user-defined data

Termination Events

The termination events for this function are:

CNFEV_UNLISTEN
Indicates successful completion of this function at which point the party device's receive channel is connected to the TDM bus time slot originally specified in a_pTimeslotInfo.

CNFEV_UNLISTEN_FAIL
Indicates indicates the function failed.

Cautions

This function fails when an invalid party handle is specified.

Errors

The CNF_ERROR_INFO data structure provides error information for the device handle when an API function fails. Upon failure the user should call **cnf_GetErrorInfo( )**, then error info for the device will be returned in the CNF_ERROR_INFO structure. Error codes are returned as: ECNF_xxxxx

Possible errors for this function include:

ECNF_INVALID_DEVICE
invalid device handle

ECNF_SUBSYSTEM internal
subsystem error

The CNF_EVENT_INFO data structure provides event information for the device handle when a notification event is enabled or disabled. This structure is used by and enabled by the **cnf_EnableEvents( )** function.Use **sr_waitevt( )**, **sr_enbhdlr( )** or other SRL functions to collect an event code, depending on the programming model in use. For more information, refer to the *Dialogic® Standard Runtime Library API Library Reference*. Error codes returned as: CNFEV_xxxxxx

Possible errors for this function include:

CNFEV_INVALID_DEVICE
invalid device handle

CNFEV_SUBSYSTEM
internal subsystem error

Example

For an example of this function, see the example for **cnf_Listen( )**.

See Also

- cnf_GetXmitSlot( )
- cnf_Listen( )

cnf_GetXmitSlot( )

Name:	int (a_PtyHandle, a_pTimeslotInfo)	
Inputs:	SRL_DEVICE_HANDLE a_PtyHandle	• valid party handle
	SC_TSINFO *a_pTimeslotInfo	• pointer to TDM bus time slot information structure
Returns:	CNF_SUCCESS on success CNF_ERROR on error	
Includes:	srllib.h cnflib.h	
Category:	TDM routing	
Mode:	Synchronous	

Description

The cnf_GetXmitSlot( ) function returns the time division multiplexing (TDM) bus time slot number of the conference party transmit channel. The TDM bus time slot information is contained in an SC_TSINFO structure that includes the number of the TDM bus time slot connected to the conference party transmit channel.

Parameter	Description
a_PtyHandle	specifies a party device handle obtained from a previous open
a_pTimeslotInfo	specifies a pointer to the SC_TSINFO structure

Cautions

This function fails when an invalid SC_TSINFO structure pointer or value(s) is specified. The data structure must be initialized to one for the sc_numts field, and have memory allocated for one long element for the sc_tsarrayp pointer field.

Errors

If the function fails with a CNF_ERROR, use cnf_GetErrorInfo to obtain the reason for the error. Possible errors for this function include:

ECNF_INVALID_DEVICE
invalid device handle

ECNF_SUBSYSTEM internal
subsystem error

ECNF_INVALID_PARM
invalid parameter

Example

```
#include <stdlib.h>
#include <stdio.h>

// Dialogic  headers
#include "srllib.h"
```

```
#include "cnflib.h"

#define MAX_DEVNAME100
#define SRWAITTIMEOUT 10000

long ProcessEvt();

int main(int argc, char* argv[])
{
    char cnfbdname[MAX_DEVNAME] = {"cnfB1"};
    SRL_DEVICE_HANDLE cnfbdh = -1;
    SRL_DEVICE_HANDLE cnfh = -1;
    SRL_DEVICE_HANDLE cnfptyh = -1;
    long ts;
    SC_TSINFO scts;
    int mode = SR_POLLMODE;

    /* Set SRL to run in polled (non-signal) mode */
    if( sr_setparm( SRL_DEVICE, SR_MODEID, &mode ) == -1 )
    {
        printf( "Error: cannot set srl mode\n" );
        exit( 1 );
    }

    cnfbdh = cnf_OpenEx(cnfbdname, NULL, NULL, EV_SYNC);
    if (cnfbdh == -1)
    {
        printf("Error during call to cnf_OpenEx\n");
        /* perform error processing */
        exit(1);
    }

  /* open conferences */
    cnfh = cnf_OpenConference(cnfbdh, NULL, NULL, NULL);
    if (cnfh == -1)
    {
        printf("Error during call to cnf_OpenConference\n");
        /* perform error processing */
        exit(1);
    }

    if(sr_waitevt(SRWAITTIMEOUT) != -1)
    {
        if (!ProcessEvt())
        {
            /* perform error processing */
            exit(1);
        }
    }
    else
    {
        printf("Error during call to sr_waitevt\n");
        /* perform error processing */
        exit(1);
    }

    cnfptyh = cnf_OpenParty(cnfbdh, NULL, NULL, NULL);
    if (cnfptyh == -1)
    {
        printf("Error during call to cnf_OpenParty\n");
        /* perform error processing */
        exit(1);
    }

    if(sr_waitevt(SRWAITTIMEOUT) != -1)
    {
        if (!ProcessEvt())
```

```
        {
            /* perform error processing */
            exit(1);
        }
    }
    else
    {
        printf("Error during call to sr_waitevt\n");
        /* perform error processing */
        exit(1);
    }

    scts.sc_numts = 1;
    scts.sc_tsarrayp = &ts;

    if (cnf_GetXmitSlot(cnfptyh, &scts))
    {
        printf("Error during call to cnf_GetXmitSlot\n");
        /* perform error processing */
        exit(1);
    }

    printf("Party %s (cnfptyh=%ld) is transmitting on %ld\n", ATDV_NAMEP(cnfptyh), cnfptyh,
    ts); return 0;
}

long ProcessEvt()
{
    long ret = 1;
    int devh;
    int evttype;
    long evtlen;
    void* datap;

    printf("ProcessEvt()\n");

    devh = sr_getevtdev();
    evttype = sr_getevttype();
    evtlen = sr_getevtlen();
    datap = sr_getevtdatap();

    switch(evttype)
    {
    case CNFEV_OPEN_CONF:
        printf("Received CNFEV_OPEN_CONF\n");
        break;

    case CNFEV_OPEN_CONF_FAIL:
        printf("Received CNFEV_OPEN_CONF_FAIL\n");
        ret = 0;
        break;

    case CNFEV_OPEN_PARTY:
        printf("Received CNFEV_OPEN_PARTY\n");
        break;

    case CNFEV_OPEN_PARTY_FAIL:
        printf("Received CNFEV_OPEN_PARTY_FAIL\n");
        ret = 0;
        break;

    default:
        printf("Unhandled event: devh(%d); evttype(0x%x)", devh, evttype);
```

```
            break;
        }

        return ret;
}
```

- **See Also**

  - cnf_Listen( )
  - cnf_UnListen( )

New Data Structure, SC_TSINFO

```
typedef struct sc_tsinfo {
        unsigned long sc_numts;
        long *sc_tsarrayp;
} SC_TSINFO;
```

- **Description**

  This structure defines the TDM bus time slot information. It may contain CT Bus or HMP soft CT Bus time slot data, and is used by the cnf_GetXmitSlot( ) and cnf_Listen( ) functions.

- **Field Descriptions**

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

  sc_numts
  specifies the number of time slots to follow; must be set to 1 for this release.

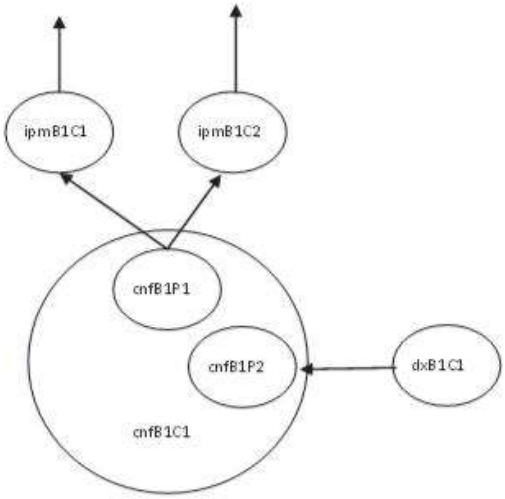
  sc_tsarrayp
  specifies the time slot ID number.

1.64.3 Use Cases

This section provides two possible use cases for reference.

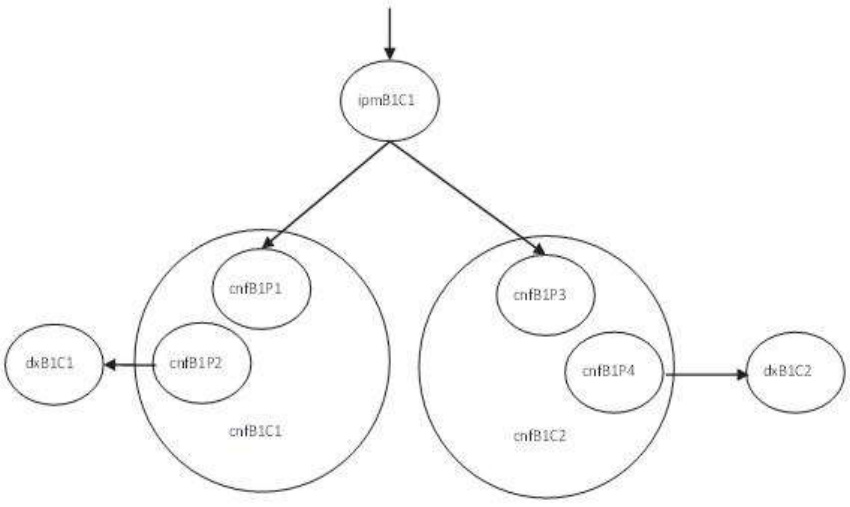
Multiple IPM Devices Listen to a Single Conference Party

In this use case, multiple externally facing devices, such as IPM devices, make half-duplex listening connections to a conference party. Here, the summed media from the conference can be broadcast to one or more external connections by using only one conference party. In the illustration below, the arrows designate the direction of the data.



Multiple Conference Parties Listen to a Single IPM Device

In this use case, multiple conference parties in different conferences make half-duplex connections to a single device such as an IPM device. Here, the single inbound media stream from the external connection can be summed into two different conferences.



1.65 Using IP Call Control (1PCC and 3PCC) Licenses without Media

Service Update 283 adds the ability to use IP Call Control licenses without media in both 1PCC and 3PCC modes. To use an 1PCC-only license, obtain and activate the license according to procedures documented in the *Dialogic® Host Media Processing Software Release 3.0WIN Administration Guide*. Once the license has been activated, a 1PCC-only Global Call application can be run.

Note: Dialogic Services (DCM) should not be started as these services provide media capabilities and are not required for an 1PCC-only application. If run, DCM will fail to start if the activated license does not define any media.

1.66 Recording and Playback of G.729A Files

With Service Update 283, the application is able to record and playback G.729A files in a Microsoft WAV file format.

1.66.1 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.

1.67 Support for the Dialogic® D/80PCIE-LS Media Board

Service Update 279 provides support for the Dialogic® D/80PCIE-LS Media Board. The board is a full-length PCI Express eight-port analog loop start for developing advanced communications applications extending the HMP media capabilities to and from the analog loop start PSTN network. The Dialogic® D/80PCIE-LS board has the same features and functionality as the current analog Dialogic® JCT boards. It also provides media streaming between HMP and its on-board analog interfaces and voice devices.

The board provides support for on-board basic voice processing, DSP-based Group 3 fax support (DSP fax or SoftFax) and continuous speech processing (CSP) in one PCI Express slot.

Refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Release Guide* for important considerations when using the Dialogic® D/80PCIE-LS board, in addition to information about supported coders, configuration considerations and the analog line adaptation utility (LineAdapt).

Note: When installing the Dialogic® D/80PCIE-LS board, be sure to refer to the Installation Guide (*Dialogic® Quick Install Card*) that is provided with each board for important information about PCI Express power budgeting and guidelines for selecting the slot where a board can be installed, and information on interconnection to the analog PSTN network or a PBX.

1.67.1 Documentation Considerations

This is the initial analog product supported with the Dialogic® PowerMedia™ HMP for Windows Release 3.0. As a result, several analog-applicable documents now appear on the online bookshelf.These documents are listed under the Analog Device (Dialogic® D/80PCIE-LS board) subheading and include the following:

- *Dialogic® Springware Architecture Products on Windows Configuration Guide (revised)*
- *Dialogic® Global Call Analog Technology Guide*
- *Dialogic® Voice API Library Reference*
- *Dialogic® Voice API Programming Reference*
- *Learn Mode and Tone Set File API Software Reference*

1.68 Virtualization Support

Service Update 275 introduces initial support for virtualization using VMware ESXi 4.0 VMware ESXi 4.0 Update 1 Installable. This release offers IP-only support with the following supported versions of Windows Server 2003:

- Windows Server 2003 (Standard or Enterprise Edition) with Service Pack 1 or 2
- Windows Server 2003 R2 (Standard or Enterprise Edition)
- Windows Server 2003 R2 (Standard or Enterprise Edition) with Service Pack 2

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.

1.68.1 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.

1.68.2 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. 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 vShpere 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.

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 3.0WIN 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.

1.68.3 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.

1.69 Overlap-Receive Support for Limited SIP-I Interworking Scenarios

Service Update 279 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. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.70 Unspecified G.723.1 Bit Rate in Outgoing SIP Requests with SDP

With Service Update 279, 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). For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.71 Processing Multiple 18x Provisional Responses

Service Update 279 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. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.72 TLS and SRTP Channel Support Increase

With Service Update 279, 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.

1.73 Registering Authentication Data without Realm String

Service Update 275 introduces a method for registering authentication data without using realm string. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.74 Support for a FAX Gateway

The Service Update provides an enhancement to provide fax T.38/V.17 Gateway capabilities. The Fax Gateway allows Fax data to be received via one transport mechanism and then converts and sends the data via a different transport type. This will allow a host running Dialogic® HMP Software to pass Fax data from an IP network (via T.38 protocol) to the PSTN (via V.17) and vice versa. This enhancement provides the programming mechanism to access this new feature, including the ability to start the gateway process and to be notified when the gateway process completes.

The T.38/V.17 gateway provides a means to convert one form of fax transmission to the other, such that each receiving end is unaware of the originating format. For the Dialogic® HMP Software Windows implementation, the conversion processing takes place in the HMP fax firmware layer. No image processing is done on the data, nor is there any T.30 protocol processing.

The API and event support needed to enable gateway processing include the following:

- A new defined protocol state value that is passed to the function **fx_initstat( )**. The new value will initiate a gateway session. The new value is: DF_T38GW.
- A new FAX event type that is returned to an enabled application. The new event type will provide notification when the gateway session completes. The new event type is: TFX_T38GW.
- New gateway error values that are returned from ATFX_ESTAT when problems arise. The new error values (and their meaning) are as follows:
  - EFX_SIGNALTIMEOUT - *Signal time-out - no data or events received during GW session*
  - EFX_DCNTIMEOUT - *DCN timeout - GW session almost complete but no DCN received*
  - EFX_BADIPADDRESS - *Bad IP address - T38 subsystem did not get remote IP address - check R4 application*
  - EFX_CTBUSERROR -*CT Bus error w/TDM portion of GW session - check R4 application*

For more information, refer to the documentation updates in Section 3.4.9, “Dialogic® Fax Software Reference”, on page 376.

1.75 Handling Non-2xx Responses to T.38 Switch

Service Update 275 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. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.76 MIME Insertion in Outgoing ACK

Service Update 272 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. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.77 Support for WaitCall Cancellation

With Service Update 272, 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.

1.77.1 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.

1.77.2 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
gcerr.h

Category: Basic

Mode: Asynchronous

Dialogic® Platform and Technology: IP†

†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.gcMsg,
                     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*.

1.78 Increase in TLS and SRTP Channels

With Service Update 272, Dialogic® HMP software software now 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.

1.79 Support for Windows 7 and Windows Server 2008 Operating Systems

Service Update 270 introduces support for the 32-bit versions of Windows 7 and Windows Server 2008 (with Service Pack 2) operating systems. For information about installing and using either operating system with the Dialogic® PowerMedia™ HMP for Windows Release 3.0, refer to the new versions of the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide* and the *Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide*.

Note: For Windows 7 and Windows Server 2008, applications must be compiled as 32-bit binaries.

1.80 Channel Density Upgrade (G.711 Voice Only)

With Service Update 270, the Dialogic® PowerMedia™ HMP for Windows Release 3.0 now supports 1000 channels of G.711 Voice on Windows 7 and Windows Server 2008 operating systems.

1.81 3PCC Support for Dynamic Selection of Outbound SIP Proxy

Service Update 270 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.

1.82 Windows XP SP3 Operating System Support

In addition to the supported operating systems listed in the Release Guide, the following operating system version is now supported with Service Update 267:

- Windows XP Professional with SP3

1.83 Defer the Sending of SIP Messages

With Service Update 267, the application can delay sending the appropriate response to an incoming BYE request (such as 200 OK), as well as delay the sending of a BYE request until the **gc_ReleaseCallEx( )** function is issued on the channel. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide*.

1.84 Intel Xeon 55xx Processor Support

With Service Update 262, Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports the Dual-Core and Quad-Core Intel Xeon 55xx processor.

1.85 Support for Dynamic Selection of Outbound SIP Proxy

With Service Update 262, 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. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide,* the *Dialogic® Global Call API Programming Guide,* and the *Dialogic® Global Call API Library Reference*.

1.86 Retrieving SIP Inbound RFC 2833

With Service Update 256, 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).

1.86.1 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 Call APIs, refer to the *Dialogic® Global Call IP Technology Guide*, *Dialogic® Global Call API Programming Guide*, and the *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(&parmblkp, IPSET_DTMF, IPPARM_SUPPORT_DTMF_BITMASK,
                        sizeof(char), IP_DTMF_TYPE_RFC_2833);

if (gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, parmblkp, GC_ALLCALLS) !=
          GC_SUCCESS) { // process error
}
    gc_util_delete_parm_blk(parmblkp);
```

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(&infoparmblkp,
                                        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.

1.87 Runtime Support for Edge Selection in Digit Reporting

With Service Update 255, the application is able to choose when digits are reported as part of call status transition (CST) events, either at the leading edge of the digit or the trailing edge.

1.87.1 Feature Description

Prior to this feature, there was no way to configure Dialogic® HMP voice resources for the reporting of CST digit detection on the trailing edge. The only setting available for CST digit event notification was leading edge.

Choosing the leading or trailing edge on which digits are to be reported is accomplished via the **dx_setparm( )** function and the existing parameter, DXCH_DFLAGS. The selection is done on a channel level. The application will be able to toggle between the two edge selections. The edge selection will be active until the next selection is made.

The DXCH_DLFLAGS parameter defaults to leading edge if it is not explicitly set by the application. When detection of CST digits is enabled, the setting of DXCH_DLFLAGS allows changing the reporting edge of digits.

Specifying Edge Selection

The existing voice channel parameter, DXCH_ DLFLAGS, affects the reporting timing of CST digit detection events. The values are:

- 0 – Default. Digit event is immediately reported upon detection (leading edge).
- 1 – Digit event reporting is delayed to its trailing edge, when its detection ceases.

DXCH_ DLFLAGS is defined as follows:

```
/* digit detection edge select */
#define DXCH_DFLAGS ((PM_SHORT|PM_FW|PM_DXXX|PM_CH) | 0x0801L)
```

Example:

```
…
/* Set chname to a voice channel name, e.g., dxxxB1C1, dxxxB1C2,...
*/ if ((chdev = dx_open(chname,NULL)) == -1) {
       /* process error */
}
…

/* 1 – Trailing Edge, 0 – Leading Edge */
int parmval = 1;
if (dx_setparm(chdev, DXCH_DFLAGS, (void *)&parmval) == -1)
{
      /* process error */
}

/* Use dx_setevtmsk() to enable digit call status transition events on this channel.*/
if (dx_setevtmsk(chdev, DM_DIGITS) == -1)
{
      /* process error */
}
```

Edge selection choice takes effect in the voice channel immediately.

Guidelines

The following guidelines are provided for edge selection:

- To select an edge, call the **dx_setparm( )** function followed by **dx_setevtmsk( )** with the appropriate parameters.
- To switch the edge selection, you must first turn off **dx_setevtmsk( )** using the DM_DIGOFF mask.

Notes:1. This feature is limited to the reporting digits as part of CST events. Edge selection will not affect the regular digit buffering reporting done through the **dx_getdig( )** function, nor will it affect digit termination timing when using any DV_TPT digit termination condition on any other relevant voice functions.

2. The default digit type is DTMF. MF type of digits is not supported by this feature.

3. The leading or trailing edge digit setting does not affect digit detection but rather affects the reporting of the digit. In other words, DTMF signal detection continues unaltered regardless of the digit edge reporting selection.

4. The CST event is still reported as DE_DIGITS (and not as DE_DIGOFF) in the DX_CST.cst_events irrespective of the edge selection.

1.87.2 Documentation

The online bookshelf provided with this release contains information about all system release features including features for application development, configuration, administration, and diagnostics.

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

1.88 Support for GSM Coder

With Service Update 255, the Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports the GSM 6.10 full-rate coder (Microsoft format). For information about this coder, refer to the *Dialogic® Voice API Programming Guide*. For information about using this coder with the Voice API, refer to the *Dialogic® Voice API Library Reference*.

1.89 Additional Operating System Support

In addition to the supported operating systems listed in the Release Guide, the following operating system version is now supported with Service Update 255:

- Windows 2003 Web Edition SP2

1.90 Support for RFC 3326 SIP Header for BYE and CANCEL Messages

With Service Update 251, the application can send and receive a Reason header for SIP BYE and CANCEL messages via the generic Global Call SIP header access functionality.

1.90.1 Feature Description

With this feature, the application has the ability to add/append the Reason headers on outgoing BYE and CANCEL requests, which indicate why the call was terminated. In the event of a BYE or CANCEL request, the application can now determine how to treat failures depending on the reason, and to transmit the reason code to the other leg of the call.

The existing Global Call methods for the retrieving of generic SIP headers will remain the same for BYE and CANCEL messages. The application uses the **gc_SetConfigData( )** function on the incoming side to register the Reason generic header and uses the **gc_SetUserInfo( )** function to send the generic header. The GCEV_DISCONNECT parameter block contains the set ID IPSET_SIP_MSGINFO and parameter ID IPPARM_SIP_HDR and data containing the generic header.

For detailed information about setting and retrieving SIP message header fields, refer to the *Dialogic® Global Call IP Technology Guide*. Updates to this document for the purpose of this feature can be found in the Documentation Updates chapter, Dialogic® Global Call IP Technology Guide.

1.90.2 Examples

To allow for the manipulation of generic SIP headers in Global Call, the following code is added before calling the **gc_Start( )** function:

```
IPCCLIB_START_DATA ipcclibstart;
IP_VIRTBOARD ip_virtboard[1];

long request_id = 0;
int index = 1;
GC_PARM_DATAP t_gcParmDatap = NULL;

CCLIB_START_STRUCT cc_Lib_Start[2] ;
GC_START_STRUCT gc_Lib_Start;

memset(&ipcclibstart, 0, sizeof(IPCCLIB_START_DATA));
memset(ip_virtboard, 0,sizeof(IP_VIRTBOARD)*1);

INIT_IPCCLIB_START_DATA(&ipcclibstart, 1, ip_virtboard);
ipcclibstart.max_parm_data_size = 1000;
INIT_IP_VIRTBOARD(&ip_virtboard[0]);
ip_virtboard[0].sip_msginfo_mask = IP_SIP_MSGINFO_ENABLE; /* override SIP
                                                          *  message
                                                          */


cc_Lib_Start[0].cclib_name = "GC_IPM_LIB";
cc_Lib_Start[0].cclib_data = NULL;
cc_Lib_Start[1].cclib_name = "GC_H3R_LIB";
```

```
    cc_Lib_Start[1].cclib_data = &ipcclibstart;
    gc_Lib_Start.cclib_list = cc_Lib_Start;
    gc_Lib_Start.num_cclibs = 2;

    if (gc_Start(&gc_Lib_Start) != GC_SUCCESS) {
}
```

The following code is added before Drop Call on the outbound side. It demonstrates how a Reason header is added to an outgoing BYE or CANCEL SIP request resulting from the **gc_DropCall( )** function.

```
    char *pReasonHdr = "reason: SIP cause=290 text=Call completed elsewhere";

    gc_util_insert_parm_ref(&parmblkp, IPSET_SIP_MSGINFO, IPPARM_SIP_HDR,
    (unsigned char)strlen(pReasonHdr ) + 1, pReasonHdr );
    gc_SetUserInfo(GCTGT_GCLIB_CRN, pline->call[0].crn, parmblkp,
                  GC_SINGLECALL);
    gc_util_delete_parm_blk(parmblkp);
```

The following code is added upon receiving a GCEV_OPENEX event on the incoming side board device. This registers a Reason header that can be retrieved upon receiving a GCEV_DISCONNECT event.

```
    char *pReasonHdr = "reason";
    long request_id = 0;
    GC_PARM_BLKP  parmblkp = NULL;
    GC_PARM_DATA_EXT parmdata;
    GC_PARM_DATAP t_gcParmDatap = NULL;
    char siphdr[IP_SIP_HDR_MAXLEN];
    gc_util_insert_parm_ref(&parmblkp, IPSET_CONFIG,
    IPPARM_REGISTER_SIP_HEADER,
    strlen(pReasonHdr ) + 1,  pReasonHdr);

    if (gc_SetConfigData(GCTGT_CCLIB_NETIF, port[index].ldev, parmblkp,
        0, GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC) != GC_SUCCESS)
```

The following code is added upon receiving a GCEV_DISCONNECT event on the incoming side:

```
    char *pReasonHdr = "reason";
      long request_id = 0;
      GC_PARM_BLKP  parmblkp = NULL;
      GC_PARM_DATA_EXT parmdata;
      GC_PARM_DATAP t_gcParmDatap = NULL;
      char siphdr[IP_SIP_HDR_MAXLEN];

    case GCEV_DISCONNECT:
          parmblkp = (GC_PARM_BLK *) metaevent.extevtdatap;
          while (t_gcParmDatap = gc_util_next_parm(parmblkp, t_gcParmDatap)){

          if(t_gcParmDatap->set_ID == IPSET_SIP_MSGINFO && t_gcParmDatap->parm_ID
             == IPPARM_SIP_HDR){
          memcpy(&siphdr, (char*)t_gcParmDatap->value_buf, t_gcParmDatap-
          >value_size); sprintf(str, "Generic Sip Header = %s", siphdr);
          printandlog(pline->index, MISC, NULL, str, 0);
       }
    }
```

1.91 Support for Proxy Bypass in SIP Register Requests

With Service Update 248, the application can override the automatic route to a configured proxy previously set while sending outgoing REGISTER request messages. This feature is applicable in 1PCC and 3PCC modes, and applies only to SIP REGISTER. It does not apply to H.323 protocol or any other SIP Message. For more information about this feature, refer to the *Dialogic® Global Call IP Technology Guide* and *Dialogic® Global Call API Library Reference.*

1.92 Increase in CSP Instances

Service Update 248 increases the current number of speech licenses. Refer to the Dialogic Product Center for additional information.

1.93 Support for RFC 3311 UPDATE Message

With Service Update 241, the application can support SIP UPDATE requests and responses in third-party call control (3PCC) mode.

1.93.1 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. Presently the UPDATE message is only supported in conjunction with SIP Session Timer refreshing and does not address updating of session parameters. This feature will utilize the Global Call Extension API mechanism for application notification and message construction.

1.93.2 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.

1.93.3 New Parameter Values

The existing parameter ID, IPPARM_MSGTYPE, supports this feature. The following new IPPARM_MSGTYPE parameter values are used for sending and receiving UPDATE requests and responses:

Parameter ID	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
	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+

1.93.4 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;
.
.
```

1.93.5 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, parmblkp, &retblkp, 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, parmblkp, &retblkp, EV_ASYNC)

Sample Code

```
// Set the SDP for 3PCC only
printf("\n  Setting  SDP  \n");
GC_PARM_BLKP 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_BLKP  set3PCCSDPInfo1(GC_PARM_BLKP 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" \
     "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_BLKP   parmblkp=NULL;
   GC_PARM_BLKP   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_SENDMSG, parmblkp, &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");
   }
}
```

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_RECEIVEMSG.

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_BLKP 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->parmblk;

   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 200 OK 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(&parmblkp,
                            IPSET_MSG_SIP,
                             IPPARM_MSGTYPE,
    sizeof(int),
    IP_MSGTYPE_SIP_UPDATE_OK);

    gc_util_insert_parm_val(&parmblkp,
                        IPSET_MSG_SIP,
    IPPARM_MSG_SIP_RESPONSE_CODE,
    sizeof(int),
    200);

    if (gc_Extension(GCTGT_GCLIB_CRN, port[0].crn, IPEXTID_SENDMSG, parmblkp,
        &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");
    }
}
```

1.93.6 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Global Call IP, refer to the following documents:

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

1.94 Exposing Raw RFC 2833 Data Using Global Call API

With Service Update 233, the application can expose received raw RFC 2833 data for each DTMF tone received using the Global Call API library. This feature is applicable in 1PCC mode only.

1.94.1 Feature Implementation

Currently, received raw RFC 2833 data is exposed at the IP Media Library level. Here, the data exists for the duration of each RFC 2833 event in milliseconds, as well as the amplitude (returned with IPMEV_TELEPHONY_EVENT). This feature extends the existing Global Call API **gc_SetUserInfo( )** function, which allows the application to set technology-specific user information. This function is used to enable notification when a DTMF tone is received. Once enabled, the application receives the new Global Call event, GCEV_TELEPHONY_EVENT, for every tone received.

This functionality is enabled/disabled on a per-channel basis using the **gc_SetUserInfo( )** function. By default, this functionality is disabled.

New Parameter IDs and Values

The existing set ID IPSET_DTMF is used to set DTMF-related parameters for notification, suppression, or sending DTMF digits. The following new parameter IDs support this feature:

Parameter ID	Value	Description
IPPARM_TELEPHONY_EVENT_DTMF	IP_ENABLE IP_DISABLE (default)	Value for enabling/disabling notification event GCEV_TELEPHONY_EVENT when DTMF tones are received.
IPPARM_TELEPHONY_EVENT_INFO	IPM_TELEPHONY_INFO	Parameter ID used to receive raw RFC 2833 data, bundled in the IPM_TELEHONY_INFO structure.

IPPARM_TELEPHONY_EVENT_DTMF

Used for DTMF digit notification when received within an RTP stream in RFC 2833 format. Notification is via the GCEV_TELEPHONY_EVENT.

IPPARM_TELEPHONY_EVENT_INFO

Used for fetching raw RFC 2833 DTMF digit information from the GCEV_TELEPHONY_EVENT event. A GCEV_TELEPHONY_EVENT notification is received for every incoming RFC 2833 DTMF digit. The information is received in the form of an IPM_TELEPHONY_INFO data structure.

New Event

THE GCEV_TELEPHONY_EVENT event delivers raw RFC 2833 data in an IPM_TELEPHONY_INFO structure. The **gc_SetUserInfo( )** function is used to enable event notification.

The IPM_TELEPHONY_INFO data structure contains information about the volume, duration and digit ID of the received RFC 2833 DTMF digit. For example:

```
typedef enum
{
   TEL_INFOTYPE_EVENT,
   TEL_INFOTYPE_TONE
} eIPM_TELEPHONY_INFO_TYPE;

typedef struct ipm_telephony_event_info_tag
{
     unsigned int   unVersion;                  /* Structure version for library use only */
     eIPM_TELEPHONY_EVENT_ID eTelephonyEventID; /* The named event usually DTMF named event */
     short          sVolume;                    /* The power level for the DTMF event tone*/
     unsigned short  usDuration;                /* Duration for the DTMF digit in ms*/
} IPM_TELEPHONY_EVENT_INFO, *PIPM_TELEPHONY_EVENT_INFO;

typedef struct ipm_telephony_info_tag
{
    unsigned int  unVersion;                 /* Structure version for library use only */
    eIPM_TELEPHONY_INFO_TYPE eTelInfoType;   /* RFC2833 Info type - named event or tone */
    union
```

```
    {
      IPM_TELEPHONY_EVENT_INFO TelEvtInfo;     /* DTMF named event info eg. DTMF digit */
      IPM_TELEPHONY_TONE_INFO  TelToneInfo;      /* Not used as part of this FR */
   }TelephonyInfo;
} IPM_TELEPHONY_INFO, *PIPM_TELEPHONY_INFO;
```

Information about the received DTMF tone is contained in the IPM_TELEPHONY_EVENT_INFO data structure. The following fields are applicable to this feature:

unVersion
: Specifies the 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
: Specifies a named event, typically a DTMF named event. The data type of the telephony_event field is an eIPM_TELEPHONY_EVENT_ID enumeration that lists all possible tone signal identifiers as described in RFC 2833. The eIPM_TELEPHONY_EVENT_ID is an enumeration with the following values:

  - 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_*
  - SIGNAL_ID_EVENT_DTMF_#

sVolume
: Specifies the power level associated with the DTMF event tone, expressed in dBm0 after dropping the sign (i.e., with a range from 0 to -63 dBm0 resulting in values from 0 to 63).

usDuration
: Specifies the duration of the DTMF digit in milliseconds.

1.94.2 Enabling Reception of RFC 2833 Raw Data

The following example demonstrates enabling event notification:

```
#include "gcip.h"
#include "gclib.h"
#include "gcip_defs.h"

int enable_dtmf_event(LINEDEV linedev)
{
    GC_PARM_BLKP parmblkp = NULL;
```

```
    /* Set the parameter block */
    gc_util_insert_parm_val(&parmblkp, IPSET_DTMF,
    IPPARM_TELEPHONY_EVENT_DTMF, sizeof(int), IP_ENABLE);

    /* Enable DTMF events functionality in the IPCCLIB */
    if (gc_SetUserInfo(GCTGT_GCLIB_CHAN, linedev, parmblkp, GC_ALLCALLS) !=
    GC_SUCCESS)
    {
           /* Process error */
    }

    /* Free the parameter block */
    gc_util_delete_parm_blk(parmblkp);

    /* More processing */
    return;
}
```

1.94.3 GCEV_TELEPHONY_EVENT Processing Example

```
#include "gcip.h"
#include "gclib.h"
#include "gcip_defs.h"
#include "ipmlib.h"

void process_event(void)
{
    METAEVENT metaevent;
    GC_PARM_BLKP my_blkp = NULL;
    GC_PARM_DATAP my_datap;
    IPM_TELEPHONY_INFO my_telInfo;

    if(gc_GetMetaEvent(&metaevent) != GC_SUCCESS)
    {
           /* Process error */
    }

    switch(metaevent.evttype)
    {
           .
           .
           .
           case GCEV_TELEPHONY_EVENT:
                  /* Make a copy of the parm blk */
                  if(metaevent.extevtdatap)
                 {
                         if ( gc_util_copy_parm_blk( &my_blkp,
                         (GC_PARM_BLKP)(metaevent.extevtdatap))!= GC_SUCCESS)
                         {
                                /* Process error */
                         }

                          my_datap = gc_util_find_parm(my_blkp, IPSET_DTMF,
                          IPPARM_TELEPHONY_EVENT_INFO);

                          if (my_datap != NULL)
                          {
                                 memcpy(&my_telInfo, my_datap->value_buf,
                                 my_datap->value_size);
                          }

                          switch(my_telInfo.eTelInfoType)
```

```
                                {
                                        case TEL_INFOTYPE_EVENT:
                                                /* Fetch DTMF tone information */
                                                break;
                                         default:

                                                /* Print Error */
                                }

                        }
                        else
                        {
                                  /* Process error */
                        }
                        break;
        .
        .
        .
    }
}
```

1.94.4 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Global Call IP, refer to the following documents:

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

For more information about IP Media API, refer to the following documents:

- *Dialogic® IP Media Library API Programming Guide and Library Reference*

1.95 Support for the Samsung IAP PBX E1 ISDN Protocol

Service Update 229 supports the proprietary E1 ISDN protocol variance known as the external ERBT Samsung IAP PBX protocol.

1.95.1 Feature Description

The Samsung IAP PBX offers a feature known as ERBT, which is similar to the Caller Ring-back Tone (CRBT) offered by many mobile carriers and service providers. CRBT service allows a subscriber to configure their service so that callers can hear a customized ring-back sound when others call the subscriber from anywhere in the telephone network, regardless of mobile or land line.

ERBT is offered as a proprietary ISDN Q.931 protocol derivative for the enterprise environment. ERBT provides a similar CRBT-like functionality on the IAP PBX. The ERBT feature comes in two modes:

- Internal: PBX handles both the signaling and the media services
- External: PBX off loads the media services to an external media server through digital E1 lines

Note: This Service Update only supports the “External” mode of ERBT.

1.95.2 Product Support

The following Dialogic® Digital Network Interface Boards support this feature:

- DNI/310TEPEHMP
- DNI/610TEPEHMP
- DNI/1200TEPEHMP
- DNI/1210TEPEHMP

1.95.3 Feature Compatibility

This feature is backwards compatible in that the Dialogic on-board protocol stack behaves as expected for a particular call unless a special SETUP message is received from a PBX channel.

When an incoming call contains a special SETUP message indicating an incoming external ERBT call, the firmware enters a separate state; however, the application still receives a standard GCEV_OFFERED event on the global call channel. At that time, a **gc_AcceptCall( )** is expected on the given global call channel, prompting an ALERTING response to the PBX channel. Any application response other than **gc_AcceptCall( )** is ignored and will produce no response from the board (acknowledgment or failure). The PBX channel responds by sending address data.

Notes:1. This feature is implemented on a call-related basis and is only persistent for the duration of that particular call.

2. The DCM Trunk Configuration should be set to E1 ISDN, either Q.Sig or Net5, for a network interface to support this protocol. Other protocols do not support this feature.

3. This protocol applies to inbound calls only. Outbound external ERBT is not supported.

1.96 Support for NAT Traversal in SIP Media Session

With Service Update 225, the application is able to send RTP/RTCP packets to the correct destination in a Network Address Translation (NAT) environment. Refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference* for information about using this feature.

1.97 Support for Egress 183 Session Progress Message

With Service Update 225, the application has the ability to send both informational response messages 180 (Ringing) and 183 (Session Progress). Currently only one, but not both can be sent. Additionally, the application is capable of specifying the SDP included with the 183 Session Progress message.

1.97.1 Feature Description

This feature allows user applications to transmit 183 Session Progress responses using the new Global Call API function, **gc_SipSessionProgress( )**. The application can also transmit both 180 Ringing and 183 Session Progress using a combination of **gc_AcceptCall( )** and **gc_SipSessionProgress( ).**

This feature is applicable when operating in 1PCC and 3PCC mode for SIP protocol only. A description of the feature and an example of its use is provided in the following section.

1.97.2 New API Library Function

The new Global Call function, **gc_SipSessionProgress( )**, is defined as follows:

Name:	int gc_SipSessionProgress(crn,mode)	
Inputs:	CRN crn	• call reference number
	unsigned long mode	• completion mode (EV_ASYNC or EV_SYNC)
Returns:	0 if successful < 0 if unsuccessful	
Includes:	gclib.h	
Category:	Optional call handling	
Mode:	Asynchronous or synchronous	
Dialogic® Platform and Technology:	IP† SIP †Refer to the *Global Call Technology Guides* for additional information.	

■ Description

The **gc_SipSessionProgress( )** function indicates to the originator that the call will be answered. This function provides a “183 Session Progress” to the destination party request acknowledging that the call has been received but is not yet answered. Upon successful completion of the **gc_SipSessionProgress( )** function, the call state changes from the Offered state to the Accepted state. If the call state is already in the Accepted state, then it will remain in the Accepted state.

This function may be used when the application needs to inform the remote side about SDP information.

Parameter	Description
crn	Specifies the call reference number for the call between the remote party A and the local party that received the call.
mode	Set to EV_ASYNC for asynchronous execution or EV_SYNC for synchronous.

Notes:1. In 1PCC mode, the SDP cannot be controlled by the user application. Therefore, **gc_SipSessionProgress( )** cannot be used to change any SDP information.

2. SDP can be set and sent when operating in 3PCC mode only.

■ Termination Events

The termination events for this function are:

GCEV_SIP_SESSIONPROGRESS
Indicates a successful completion of the function.

GCEV_TASKFAIL
Indicates indicates the function failed.

■ Cautions

The **gc_SipSessionProgress( )** function will fail when called for an unsupported technology. The error value EGC_UNSUPPORTED will be the Global Call value returned when the **gc_ErrorInfo( )** function is used to retrieve the error code.

■ Errors

If this function returns <0 to indicate failures, use the **gc_ErrorInfo( )** function for error information. If the GCEV_TASKFAIL event is received, use the **gc_ResultInfo( )** function to retrieve information about the event.

■ Example

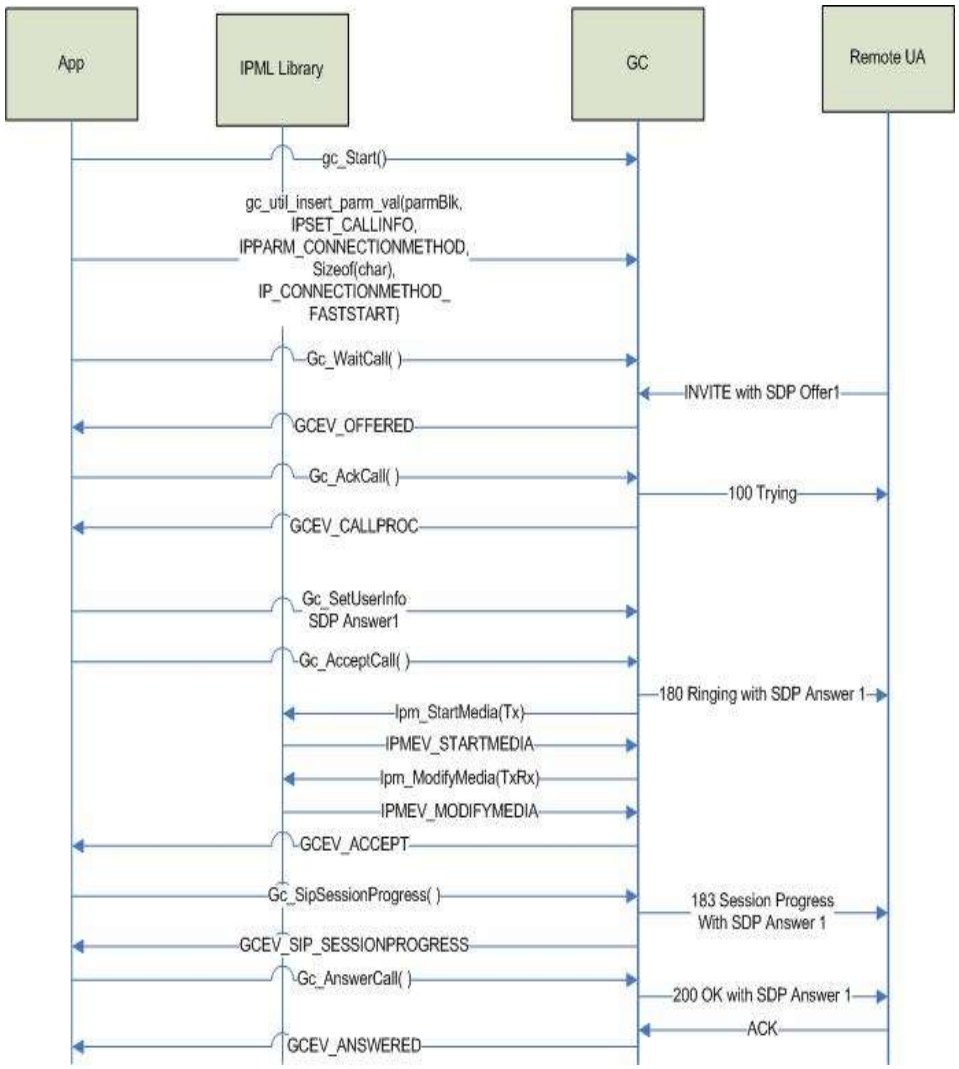
```
#include <gclib.h>
#include <gcerr.h>
.
.
.
/*
 * Assume the following has been done:
 * 1. Opened line devices for each time slot
 * 2. Wait for a call using gc_WaitCall( )
 * 3. An event has arrived and has been converted to a
 * metaevent using gc_getMetaEvent() or
 * gc_getMetaEventEx()
 * 4. The event is determined to be a GCEV_OFFERED event
 */
 Int progress_call(void)
 {
     CRNcrn;/* Call reference number */
     GC_INFOgc_error_info;/* GlobalCall error information data */

 /*
  * Send a 183 Session Progress response
```

```
 */
     Crn = metaevent.crn;
     if (gc_SipSessionProgress(crn, , EV_ASYNC) != GC_SUCCESS) {
     /* process error returned */
          Gc_ErrorInfo( &gc_error_info );
          Printf("Error gc_SipSessionProgress( ) on device
                    handle: 0x%1x, GC ErrorValue:
                    0x%hx - %s, CCLibID: %i - %s, CC
                    ErrorValue: 0x%1x - %s\n",
                    Metaevent.evtdev,
                    gc_error_info.gcValue,
                    gc_error_info.gcMsg,
                    gc_error_info.ccLibId,
                    gc_error_info.ccLibName,
                    gc_error_info.ccValue,
                    gc_error_info.ccMsg);
          return (gc_error_info.gcvalue);
    }
 /*
  * gc_SipSessionProgress( ) terminates with
  * GCEV_SIP_SESSIONPROGRESS event. When the
  * GCEV_SIP_SESSIONPROGRESS event is received, the
  * state changes to Accepted and the
  * gc_AnswerCall( ) can be issued to complete the
  * connection.
  */
    return (0);
}
```

1.97.3 Use Case Scenario

The scenario outlined in the figure shows an Ingress INVITE with and SDP offer (fast start), followed by a subsequent Egress 180 Ringing with SDP and 183 Session Progress response. This scenario is in 3PCC mode.



1.98 Software Rebranding Update

Beginning with Service Update 210, the Dialogic® HMP Software default installation directory structure changes from Intel\HMP to Dialogic\HMP. A complete uninstall of the previous software release is required before any customer installs Service Update 210 or higher from a machine that had Service Update 202 or lower. Once Service Update 210 or higher is installed, then future service updates may use the update install rather than the full install.

Prior to performing the uninstall, or any other install, a back up of the entire file structure of the former Intel\HMP\data is recommended because the whole directory is removed by an uninstall. At the very least, be sure that the following file types are backed up in a separate user-defined directory outside of the former Intel\HMP directory:

- Configuration files (*.config and .uconfig*) located in directories Intel\HMP\data and Intel\HMP\cfg
- License files (*.lic*) located in directory Intel\HMP\data
- Log files (*.log*) Intel\HMP\log

A log file captures what happens during the uninstall. For more information, refer to the "Installation Log Files" section in *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide*.

Other software rebranding changes include the renaming of the HMP License Manager from Intel HMP License Manager to Dialogic HMP License Manager, and the removal of PBX technology support.

1.99 Global Call API Access to New H.323/Q.931 Message IEs

Service Update 210 adds this previously released feature where you can access and configure the called and calling party number information elements (IEs) and various subfields of the H.322/Q.931 SETUP message.

1.99.1 Feature Description

You now have the ability to access additional fields in the calling party number (CGPN) and called party number (CDPN) IEs within a H.225/Q.931 SETUP message when using H.323 IP call signaling.

The SETUP message is a standard call signaling message used by a calling H.323 entity to establish a connection with the called entity. This message is currently supported with limited access to information fields in the CGPN and CDPN IEs. Presently only the Called/Calling Party number can be modified by the application via **gc_SetUserInfo( )** or when invoking the **gc_MakeCall( )** function. You now can use an existing parameter set ID and its new parameter IDs to send and receive these CPN fields via Global Call over an IP network.

1.99.2 New Parameter IDs

An existing parameter set ID (IPSET_CALLINFO) and new parameter IDs support the CPN information as shown in the table below:

New IPSET_CALLINFO Parameter IDs

Parameter ID	Set	Sent	Retrieve
IPPARM_CGPN_TYPE_OF_NUMBER IPPARM_CDPN_TYPE_OF_NUMBER IPPARM_CGPN_NUMBERING_PLAN_ID IPPARM_CDPN_NUMBERING_PLAN_ID IPPARM_CGPN_SCREENING_INDICATOR IPPARM_CGPN_PRESENTATION_INDICATOR	GC_PARM_BLK **gc_SetUserInfo( )**	**gc_MakeCall( )**	**gc_Extension( )** (IPEXTID_GETINFO) and asynchronous GCEV_EXTENTIONCMPLT completion event

Parameter ID Details

Parameter ID	Data Type & Size	Description
IPPARM_CGPN_TYPE_OF_NUMBER IPPARM_CDPN_TYPE_OF_NUMBER	Type: unsigned char Size: 1 byte	Contains the type of number in the CGPN or CDPN
IPPARM_CGPN_NUMBERING_PLAN_ID IPPARM_CDPN_NUMBERING_PLAN_ID	Type: unsigned char Size: 1 byte	Contains the numbering plan identification in the CGPN or CDPN
IPPARM_CGPN_SCREENING_INDICATOR	Type: unsigned char Size: 1 byte	Contains the screening indicator in the CGPN
IPPARM_CGPN_PRESENTATION_INDICATOR	Type: unsigned char Size: 1 byte	Contains the presentation indicator in the CGPN

The following definitions are in the *gcip_defs.h* file:

```
#define IPPARM_ CGPN_TYPE_OF_NUMBER                0x13
#define IPPARM_ CDPN_TYPE_OF_NUMBER                0x14
#define IPPARM_ CGPN_NUMBERING_PLAN_ID             0x15
#define IPPARM_ CDPN_NUMBERING_PLAN_ID             0x16
#define IPPARM_ CGPN_SCREENING_INDICATOR           0x17
#define IPPARM_ CGPN_PRESENTATION_INDICATOR        0x18
```

The following data variables are in the *gcip.h* file:

```
typedef unsigned char  CPN_TON;      /* Type of number */
typedef unsigned charunsigned char  CPN_NPI;      /* Numbering plan identification */
typedef unsigned charunsigned char  CPN_SI;       /* Screening Indicator        */
typedef unsigned charunsigned char  CPN_PI;       /* Presentation Indicator */
```

1.99.3 Enabling the Setting and Retrieving of Q.931 Message IEs

To enable the setting and retrieving of all supported Q.931 message IEs, set the h323_msginfo_mask field in IP_VIRTBOARD structure to a value of IP_H323_MSGINFO_ENABLE for each IPT board device **before** calling **gc_Start( )**.

Note: By default the underlying H.323 stack is not enabled to receive incoming Q.931 message IEs.

A code snippet showing how to do this is given below:

```
IP_VIRTBOARD virtBoard[MAX_BOARDS];
memset(virtBoard,0,sizeof(IP_VIRTBOARD) * MAX_BOARDS);
bid = 1;
INIT_IP_VIRTBOARD(&virtBoard[bid]);
// fill up other board parameters
….
 virtBoard[bid].localIP.ip_ver = IPVER4;
 virtBoard[bid].localIP.u_ipaddr.ipv4 = (unsigned int) IP_CFG_DEFAULT;
…
virtBoard[1].h323_msginfo_mask = IP_H323_MSGINFO_ENABLE;
//Then use the virtBoard structure in the gc_Start() function appropriately
```

You can also enable reception of other H.323 fields simultaneously via the code:

```
virtBoard[1].h323_msginfo_mask = IP_H323_MSGINFO_ENABLE | IP_H323_ANNEXMMSG_ENABLE;
```

1.99.3.1 Stopping the Reception of CPN Information

Currently, there is no way for the user application to turn off the reception of Q.931 IEs received by the underlying H323 stack once they are enabled, without stopping the application or restarting the stack.

1.99.4 Setting Up CPN Fields in the GC_PARM_BLK Data Structure

Before calling the **gc_MakeCall( )** function, you must set up the CPN fields to be included in the GC_PARM_BLK data structure. The GC_PARM_BLK should include the existing parameter set ID IPSET_CALLINFO and the newly defined parameter IDs described in the New Parameter IDs section, which specifies which CPN fields are to be set in the parameter block structure of a Make Call block.

To set up the CPN fields in the GC_PARM_BLK structure, call the **gc_util_insert_parm_ref( )** function.

Code Example

The following is an example of how to specify the CPN fields for sending.

```
#include <stdio.h>
#include <string.h>
#include <gcip.h>
#include <.h>
```

```
void main()
{
   CPN_TON    cgpn_ton, cdpn_ton;
   CPN_NPIcgpn_npi, cdpn_npi;
   CPN_SIcgpn_si;
   CPN_PIcgpn_pi;
   GC_PARM_BLKPpParmBlock;

  /*. .   Main Processing….*/
  /*  Set CPN fields in the Make Call Block to be sent out via SETUP message */

  cgpn_ton = 0x1;   // Note that the field values must be valid.
  cdpn_ton = 0x4;
  cgpn_npi = 0x1;
  cdpn_npi = 0x1;
  cgpn_si = 0x0;
  cgpn_pi = 0x0;

    gc_util_insert_parm_ref(&pParmBlock,
                            IPSET_ CALLINFO,
                            IPPARM_CGPN_TYPE_OF_NUMBER,
                            sizeof(unsigned char),
                            &cgpn_ton);

    gc_util_insert_parm_ref(&pParmBlock,
                            IPSET_ CALLINFO,
                            IPPARM_CDPN_TYPE_OF_NUMBER,
                            sizeof(unsigned char),
                            &cdpn_ton);

    gc_util_insert_parm_ref(&pParmBlock,
                            IPSET_ CALLINFO,
                            IPPARM_CGPN_NUMBERING_PLAN_ID,
                            sizeof(unsigned char),
                            &cgpn_npi);

    gc_util_insert_parm_ref(&pParmBlock,
                            IPSET_ CALLINFO,
                            IPPARM_CDPN_NUMBERING_PLAN_ID,
                            sizeof(unsigned char),
                            &cdpn_npi);

    gc_util_insert_parm_ref(&pParmBlock,
                            IPSET_ CALLINFO,
                            IPPARM_CGPN_SCREENING_INDICATOR,
                            sizeof(unsigned char),
                            &cgpn_si);

    gc_util_insert_parm_ref(&pParmBlock,
                            IPSET_ CALLINFO,
                            IPPARM_CGPN_PRESENTATION_INDICATOR,
                            &cgpn_pi);

  /*. .. Continue Main processing. … call gc_MakeCall() */
}
```

1.99.5 Generating CPN Fields in a SETUP Message

If you choose to insert the CPN data into the GC_MAKECALL_BLK using the **gc_SetUserInfo( )** function, refer to the *Dialogic® Global Call IP Technology Guide* for details on how this can be done.

After setting the CPN signaling message fields as described in the Setting Up CPN Fields in the GC_PARM_BLK Data Structure section, the user application calls the **gc_MakeCall( )** function passing it a pointer to the GC_MAKECALL_BLK.

1.99.6 Retrieving CPN Information

When the underlying H.323 stack is enabled to retrieve incoming call info fields, including CPN information, any CPN fields detected in an associated H.225 message will be retrieved and stored in Global Call.

Use the **gc_Extension( )** function to retrieve the CPN data within any valid incoming H.225 message containing CPN fields, while a call is in any state, after the OFFERED state. This is similar to receiving other call related information via the GCEV_EXTENSIONCMPLT event received as a termination event to the **gc_Extension( )** function.

Set the target_type to GCTGT_GCLIB_CRN. Set the target_id to the actual CRN. If incoming CPN data is available, the information is included with the corresponding GCEV_EXTENSIONCMPLT asynchronous termination event.

The extevtdatap field in the METAEVENT structure for the GCEV_EXTENSIONCMPLT event is a pointer to an EXTENSIONEVTBLK structure that contains a GC_PARM_BLK with the requested CPN information.

When trying to retrieve the CPN information, it is necessary to specify each of the CPN parameters in the extension request, for which information from the stack is needed.

Code Examples

Specifying CPN Field for Receiving

A code example of how to specify the CPN fields for receiving is shown below. This example is just for the Calling Number Type of Number field. The method to specify the other CPN data would be similar.

```
int getCPNInfo(CRN crn)
{
   GC_PARM_BLKP gcParmBlk = NULL;
   GC_PARM_BLKP retParmBlk;
   int frc;

   frc = gc_util_insert_parm_val(&gcParmBlk,
                                 IPSET_CALLINFO,
                                 IPPARM_CGPN_TYPE_OF_NUMBER,
                                 sizeof(unsigned char),1);

   if (GC_SUCCESS != frc)
   {
       return GC_ERROR;
   }
```

```
   frc = gc_Extension (GCTGT_GCLIB_CRN,
                       crn,
                       IPEXTID_GETINFO,
                       gcParmBlk,
                       &retParmBlk,
                       EV_ASYNC);
   if (GC_SUCCESS != frc)
   {
       return GC_ERROR;
   }

   gc_util_delete_parm_blk(gcParmBlk);

    return GC_SUCCESS;

}
```

Retrieving CPN Information

A code example of how to extract CPN information from an unsolicited GCEV_EXTENSIONCMPLT event received as a result of a request for call-related information is shown below:

```
int OnExtension(GC_PARM_BLKP parm_blk,CRN crn)
{
   GC_PARM_DATA *parmp = NULL;
   parmp = gc_util_next_parm(parm_blk,parmp);

   if (!parmp)
   {
      return GC_ERROR;
   }

   while (NULL != parmp)
   {
     switch (parmp->set_ID)
     {
       case IPSET_CALLINFO:
           switch (parmp->parm_ID)
           {
             case IPPARM_CGPN_TYPE_OF_NUMBER:
                   printf("\tReceived CPN data Calling Party Type of Number: %d\n",
(*(unsigned char*)(parmp->value_buf)));
    break;

    case IPPARM_CDPN_TYPE_OF_NUMBER:
                   printf("\tReceived CPN data Called Party Type of Number: %d\n",
(*(unsigned char*)(parmp->value_buf)));
    break;

     default:
                   printf("\tReceived unknown extension parmID %d\n",
                   parmp->parm_ID);

     break;
          }

    break;
    parmp = gc_util_next_parm(parm_blk,parmp);
  }
}
```

1.99.7 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Global Call IP, refer to the following documents:

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

For more information about IP Media API, refer to the following documents:

- *Dialogic® IP Media Library API Programming Guide and Library Reference*

1.100 Support for Ingress 183 Session Progress Informational Response Containing SDP

With Service Update 202, the application can enable/disable the sending of an unsolicited event to the application upon receipt of a SIP183 Session Progress informational response message. This feature can be enabled at the board level and at a per-call level.

1.100.1 Feature Description

This feature allows user applications to register for an unsolicited call status event upon receiving an ingress 183 Session Progress response. Both first party call control (1PCC) and third party call control (3PCC) modes are supported.

For 3PCC, access to the associated Session Description Protocol (SDP) answer information contained in the informational response message is also provided.

When operating in 3PCC mode, the application will initiate early media by establishing a bidirectional path (transfer/receive) based on the media attributes offered by the remote endpoint as specified in the SDP answer.

Note: This feature applies to Fast Start only.

1.100.2 New Parameter Value

The following table lists the new Global Call parameter value added for the set ID GCSET_CALLEVENT_MSK and parameter ID GCACT_SETMSK/GCACT_ADDMSK/GCACT_SUBMSK combination for this feature.

Definition (#define)	Type	Description
GCMSK_PROGRESSING	Global Call Set ID	Value for enabling/disabling sending of GCEV_PROGRESSING event to the application upon receipt of 183 session response message.

1.100.3 Code Examples

The following example illustrates how to enable the feature at the board level to support sending an unsolicited event (GCEV_PROGRESSING) to the application.

```
#include "gclib.h"
..
..

// To set on a board level
GC_PARM_BLK *pParmBlock = NULL;
long request_id = 0;

gc_util_insert_parm_val(&pParmBlock,
                        GCSET_CALLEVENT_MSK,
                        GCACT_SETMSK,
                        sizeof(long),
                        GCMSK_PROGRESS);

// Set config data
gc_SetConfigData(GCTGT_CCLIB_NETIF,
                 boarddev,
                 pParmBlock,
                 0,
                 GCUPDATE_IMMEDIATE,
                 &request_id,
                 EV_ASYNC);

gc_util_delete_parm_blk(pParmBlock);
```

The following example illustrates enabling this feature at the call level.

```
#include "gclib.h"
..
..

GC_PARM_BLK *ParmBlock = NULL;

gc_util_insert_parm_val(&pParmBlock,
                        GCSET_CALLEVENT_MSK,
                        GCACT_SETMSK,
                        sizeof(long),
                        GCMSK_PROGRESS);

// Set config data
gc_SetUserInfo(GCTGT_GCLIB_CHAN,
               ldev,
               pParmBlock,
               GC_ALLCALLS);

gc_util_delete_parm_blk(pParmBlock);
```

1.100.4 Retrieving SDP Data from GCEV_PROGRESSING in 3PCC Mode

The GCEV_PROGRESSING event has an associated GC_PARM_BLK that contains extra data about the event. Depending on the list of header fields that the application has registered to receive, the GC_PARM_BLK associated with the GCEV_PROGRESSING event may contain multiple parameter elements that use the

IPSET_SIP_MSG_INFO/IPPARM_SIP_HDR ID pair. The SDP data in the GCEV_PROGRESSING event can be retrieved using the IPSET_SDP/IPPARM_SDP_ANSWER ID pair.

The following example illustrates how the user application can retrieve the information from the unsolicited event in the 3PCC mode.

```
switch(metaeventp->evttype)
{
   case GCEV_PROGRESSING:
     printf("Received GCEV_PROGRESSING. Retrieve the SDP data
     \n"); get3PCCSDPInfo(metaeventp->extevtdatap);

   break;
   default:
     printf("Unexpected event …… %d", metaeventp->evttype);
   break;
}

int get3PCCSDPInfo(GC_PARM_BLK* pParmBlk)
{
   int retCode = 0;  // for success case.

   printf(" Looking for 3PCC SDP...\n");

   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:

         if (3PCCmode == 1) {
 printf(" 3PCC SDP recieved (IPPARM=0x%x):\n%s\n\n",
                                    ParmDataExt.parm_ID,
                                    (char *)(ParmDataExt.pData));
   } else if (3PCCmode > 1) {
              printf(" ERROR!!! 3PCC SDP recieved, but not
                      expecting any! (IPPARM=0x%x):\n%s\n\n",
                                    ParmDataExt.parm_ID,
                                    (char *)(ParmDataExt.pData));
 retCode = -1;
   } else {
              printf(" ERROR!!! 3PCC SDP recieved, but not in 3PCC
                      mode! (IPPARM=0x%x):\n%s\n\n",
                                    ParmDataExt.parm_ID,
                                    (char *)(ParmDataExt.pData));
 retCode = -1;
   }
   break;

   default:

   if (3PCCmode == 1) {
              printf(" ERROR!!! 3PCC SDP recieved, invalid IPPARM!
                      (IPPARM=0x%x):\n%s\n\n",
```

```
                                        ParmDataExt.parm_ID,
                                        (char *)(ParmDataExt.pData));
   } else if (3PCCmode > 1) {
               printf(" ERROR!!! 3PCC SDP recieved with invalid
                       IPPARM, but not expecting any!
                       (IPPARM=0x%x):\n%s\n\n",
                                        ParmDataExt.parm_ID,
                                        (char *)(ParmDataExt.pData));
         retCode = -1;
   } else {
               printf(" ERROR!!! 3PCC SDP recieved with invalid
                       IPPARM, but not even in 3PCC mode!
                       (IPPARM=0x%x):\n%s\n\n",
                                        ParmDataExt.parm_ID,
                                        (char *)(ParmDataExt.pData));
         retCode = -1;
   }
   break;
}
    }

    UtilRet = gc_util_next_parm_ex(pParmBlk, &ParmDataExt);

  }
}
return retCode;
     }
```

1.101 Support for DM_PORT_CONNECT_INFO Data Structure

With Service Update 202, the application can use the **dev_PortConnect( )** API library function to specify transmit and receive port connection information.

1.101.1 DM_PORT_CONNECT_INFO Data Structure

This feature is supported by using the DM_PORT_CONNECT_INFO data structure, and setting the parameter flag, DMFL_TRANSCODE_ON, in the unFlags field.

The description of the DM_PORT_CONNECT_INFO data structure is:

```
typedef struct
{
     unsigned int         unVersion;
     unsigned int         unFlags;
     DM_PORT_INFO         port_info_tx;
     DM_PORT_INFO         port_info_rx;
} DM_PORT_CONNECT_INFO, *PDM_PORT_CONNECT_INFO;

typedef const DM_PORT_CONNECT_INFO* CPDM_PORT_CONNECT_INFO;
```

Description

This structure specifies transmit and receive port information for a connection. This structure is a child structure of the DM_PORT_CONNECT_INFO_LIST structure. For more information about the Dialogic® Device Management library, refer to the *Dialogic® Device Management API Library Reference.*

The INIT_DM_PORT_CONNECT_INFO inline function is provided to initialize the structure.

Field Descriptions

The fields of DM_PORT_CONNECT_INFO data structure are described as follows:

unVersion
: The version number of the data structure. Use the inline function to initialize this field to the current version.

unFlags
: Flags specifying details of the connection to establish:
  - DMFL_TRANSCODE_ON - default mode
  - DMFL_TRANSCODE_NATIVE - native (no transcoding)

  Note: Video transcoding is not supported. Set the flag to DMFL_TRANSCODE_NATIVE for ports of type DM_PORT_MEDIA_TYPE_VIDEO.

port_info_tx
: Transmit port information, specified in the DM_PORT_INFO structure.

port_info_rx
: Receive port information, specified in the DM_PORT_INFO structure.

1.102 Support for NCM Library Tracing within RTF Logging

With Service Update 199, the application has support for the Dialogic® NCM (Native Configuration Manager) library tracing within the Runtime Trace Facility (RTF) log.

1.102.1 Function Description

RTF provides a mechanism for tracing the execution path of the Dialogic® Runtime libraries. The resulting log file/debug stream output helps troubleshoot runtime issues for applications using Dialogic® software.

With this feature, logging information for the NCM API is available for customer viewing in the RTF logs. The customer no longer has to look in more than one file to obtain this logging information.

1.102.2 Porting NCM API Traces to RTF Logs

The RTF tool obtains trace control settings and output formatting from an RTF configuration file (*RtfConfigWin.xml*). Each runtime library has several levels of tracing that can be dynamically enabled/disabled by editing the RTF configuration file while your application runs, and entering `rtftrace -start` and `rtftrace -stop`. To enable logging, the `Subsystem Name` and `Parameter Values` tags need to be in the *RtfConfigWin.xml* file:

The following configuration example demonstrates how to enable attributes to place trace functions in the RTF Log file by assigning “1” as the parameter value.

```
RtfConfigWin.xml
<Module family="DM3,SPWR,HMP" name="OAMSYSLOG" state="1"
    technology="OAM"> <MLabel name="ErrorEx" state="1"/>
    <MLabel name="Warning" state="1"/>
    <MLabel name="Eventing" state="1"/>
    <MLabel name="Info" state="1"/>
    <MLabel name="ApiEntry" state="1"/>
    <MLabel name="ApiExit" state="1"/>
 </Module>
```

1.103 Native T.38 Hairpinning Support

This Dialogic® HMP Software Release, Service Update 195, supports native T.38 hairpinning under 3rd Party Call Control (3PCC) SIP. For information about this feature, refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

1.104 AMD Support for PSTN

With Service Update 192, Dialogic® PowerMedia™ HMP for Windows Release 3.0 includes AMD support for PSTN (PCI Express boards only).

1.105 Channel Density Upgrade (G.711, Voice & Conferencing Only)

Service Update 192 for the Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports 750 channels without conferencing and 580 channels with conferencing.

Note: The NIC performance parameters need to be set at densities higher than 400 channels to obtain optimal performance in voice quality and to avoid possible packet loss. These parameters can be set on the Properties page in the Device Manager. Before making parameter changes, you may need to install the latest drivers for the NIC. To get to your network interface card's Properties page, proceed as follows:

- Right-click on My Computer and go to Manage. The Computer Management screen is displayed.
- In the left pane under System Tools, click on Device Manager. Device Manager is displayed in the right pane.
- In Device Manager, expand Network adapters.
- Right-click on your network adapter and click Properties. For example, on the Properties page for an Intel PRO/1000 MT Desktop Adapter, the Advanced tab includes a selection for Performance Options.
- Under this option, increase the Receive Descriptors and Transmit Descriptors to 2048.

1.106 Support for Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards

With Service Update 182, Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards. These Dialogic® HMP Interface Boards are high-performance digital network interface boards in a full-length PCI Express form factor. The DNI/310TEPEHMP has one E1/T1 network interface, the DNI/610TEPEHMP has two E1/T1 network interfaces, and the DNI/1210TEPEHMP has four E1/T1 network interfaces.

The use of Dialogic® HMP Interface Boards with Dialogic® HMP Software enables applications developed using Dialogic® API libraries in an HMP environment to connect to PSTN networks through E1 or T1 network interfaces. The Dialogic® HMP Interface Boards can be used for converged IP-PSTN solutions that require both IP and PSTN connectivity.

This section provides information about:

- Features of the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards
- Installing the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards
- Installing Dialogic® HMP Software
- Obtaining a License File
- Configuring the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards
- Enabling Echo Cancellation with Non-Linear Processing

1.106.1 Features of the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards

Features of the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards include:

Interface to Dialogic® HMP Software
: Allows host-based voice, speech, conference, fax, and IP transcoding to be accessible from the PSTN interface; can be configured in a wide range of densities, scalable in individual port increments

One, two, or four digital network interfaces
: Provides different densities to support a cost-effective range of solutions

Software-selectable trunks configure DNI Boards for either E1 or T1
: Reduces the total cost of ownership by increasing flexibility, reducing inventory, and simplifying the purchasing process and test effort

Support for a wide range of PSTN protocols including ISDN and CAS signaling Allows a choice of PSTN protocols

Dialogic® Global Call Software
: Provides a consistent programming interface for call control utilized by boards with Dialogic® DM3 architecture and by Dialogic® HMP Software

Host streaming interface
: Enables a low-latency, 256-duplex channel interface to host-based media and IP networks

Soft H.100 CT Bus
: Provides a software-defined bus that extends the traditional board-based H.100 CT Bus to host-based media and IP networks

For further information about features, applications, and technical specifications, refer to the product data sheet, which is available at *http://www.dialogic.com*.

1.106.2 Installing the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards

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 equipment.

Note: Refer to the Installation Guide for important information about power budgeting and guidelines for selecting the slot where a board can be installed. If the board is not configured and installed properly, it will not be detected in the system.

1.106.3 Installing Dialogic® HMP Software

The Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards require the use of a Dialogic® HMP Software release that specifically supports these boards. If you are installing one of these boards in a system that already has HMP Software installed, you should verify that your installed software version supports the board. If it does not support the board, you will need to obtain and install a Service Update that does support the board before configuring the system for the newly installed board(s).

Note: When installing the Dialogic® HMP Software, the **Circuit Connectivity Runtime Package** is required when using Dialogic® HMP Interface Boards. This is one of the install options on the Select Features screen during the installation process. If the HMP Software has already been installed without this package, you can add it by using Add/Remove Programs.

For detailed information about installing the HMP Software, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide*.

1.106.4 Obtaining a License File

Before you use the Dialogic® HMP Software and the supported Dialogic® HMP Interface Boards, you must obtain a license file containing HMP license data. 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 supported boards, but you need to obtain a license file before you can proceed with using the Dialogic® HMP Software and supported boards.

If you have been using the Dialogic® HMP Software *without* Dialogic® HMP 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 Dialogic® HMP Interface Boards. 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 Dialogic® HMP Interface Boards to a system that already has a board-based license, you can use the existing license but you may have to upgrade it.

For detailed information about obtaining and upgrading a license, refer to the *Dialogic® Host Media Processing Administration Guide*.

1.106.5 Configuring the Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards

After the Dialogic® HMP Software is installed and the appropriate licensing is obtained, you start the configuration process by invoking the Dialogic® Configuration Manager (DCM). (The Native Configuration Manager (NCM) API can also be used for system configuration.)

The *Dialogic® Host Media Processing Configuration Guide* provides detailed instructions for configuring Dialogic® HMP Software and HMP Interface Boards in the system. When configuring the system for the new PCI Express form factor boards, use the same procedures that are documented for the PCI version of the boards.

If you are using the Dialogic® Global Call protocols, refer to the *Dialogic® Global Call CDP Configuration Guide* for additional configuration procedures.

Note: In the *pdk.cfg* file, the following mlmfile options are used with the DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Boards:

- **mlmfile hmp_pdk_octal.mlm.sym** if the board is using CAS on all trunks
- **mlmfile hmp_mixed_octal.mlm.sym** if the board is mixing CAS and ISDN protocols

1.106.6 Runtime Configuration of Echo Cancellation

Refer to the Enabling Echo Cancellation with Non-Linear Processing section for information about runtime configuration of echo cancellation.

1.107 Support for 720 Channels

This Dialogic® HMP Software Release, Service Update 182, provides support for up to 720 simultaneous voice calls with DTMF using three Dialogic® DNI/2410TEPEHMP Digital Network Interface Boards.

1.108 Cisco Call Manager 6.0 Support

This Dialogic® HMP Software Release, Service Update 176, has been validated and/or tested for interoperability with the Cisco Call Manager 6.0.

Refer to the *Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide* for information about other devices tested for interoperability.

1.109 Signaling Licenses Increase to 5000

With this Dialogic® HMP Software Release, Service Update 174, IPCCLIB now allows up to 5000 channels of IP Call Control on a single virtual board. Previously, the maximum allowed was 2016 channels. For more information, refer to the *Dialogic® Global Call IP Technology Guide.*

1.110 Configuring SIP Stack Parameters with Global Call

With this Dialogic® HMP Software Release, Service Update 174, selected SIP stack parameters such as timers can now be configured with the Dialogic® Global Call API.

1.110.1 Feature Description

A new data structure, SIP_STACK_CFG, is used to configure SIP stack parameters. Details about the SIP_STACK_CFG data structure fields are given in Section 1.110.2, “SIP_STACK_CFG Data Structure”, on page 209.

To support SIP stack configuration, IP_VIRTBOARD has been updated with a new structure pointer (default is NULL) as follows:

```
typedef struct {
    ...
    ...
    /* The following is added for VIRTBOARD_VERSION_SIP_STACK_CFG support
        */ SIP_STACK_CFG *sip_stack_cfg;
    /* end VIRTBOARD_VERSION_SIP_STACK_CFG additions */
} IP_VIRTBOARD;
```

1.110.2 SIP_STACK_CFG Data Structure

The SIP_STACK_CFG structure definition has been added in the *gcip.h* file. The new data structure is described below.

Note: SIP stack parameters can only be configured once per virtual board (at **gc_Start( )**) and remain in effect throughout the Global Call application (per process).

SIP_STACK_CFG

```
typedef struct {
    unsigned long version;  /* version set by INIT_SIP_STACK_CFG */
    int retransmissionT1;
    int retransmissionT2;
    int retransmissionT4;
    int generalLingerTimer;
    int inviteLingerTimer;
    int provisionalTimer;
    int cancelGeneralNoResponseTimer;
    int cancelInviteNoResponseTimer;
    int generalRequestTimeoutTimer;
} SIP_STACK_CFG;
```

Description

The SIP_STACK_CFG data structure is used to configure selected SIP stack parameters such as timers.

The SIP_STACK_CFG data structure is referenced by the IP_VIRTBOARD data structure, which stores configuration and capability information about an IPT (virtual) board device that is populated when the device is started. An array of IP_VIRTBOARD structures (one per virtual board in the system) is referenced by the IPCCLIB_START_DATA structure, which is passed to the **gc_Start( )** function.

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

Field Descriptions

The fields of the SIP_STACK_CFG data structure are:

version
: The version number of the data structure. The correct value is set by the **INIT_SIP_STACK_CFG( )** initialization function and should not be overridden.

retransmissionT1
: Determines several timers as defined in RFC 3261. For example, when an unreliable transport protocol is used, a Client Invite transaction retransmits requests at an interval that starts at T1 milliseconds and doubles after every retransmission. A Client General transaction retransmits requests at an interval that starts at T1 and doubles until it reaches T2. The default value is 1000.

retransmissionT2
: Determines the maximum retransmission interval as defined in RFC 3261. For example, when an unreliable transport protocol is used, general requests are retransmitted at an interval that starts at T1 and doubles until it reaches T2. If a provisional response is received, retransmissions continue but at an interval of T2. The parameter value cannot be less than 4000. The default value is 8000.

retransmissionT4

Determines the amount of time the network takes to clear messages between client and server transactions as defined in RFC 3261. For example, when working with an unreliable transport protocol, T4 determines the time that a UAS waits after receiving an ACK message and before terminating the transaction. The default value is 10000.

generalLingerTimer

After a server sends a final response, the server cannot be sure that the client has received the response message. The server should be able to retransmit the response upon receiving retransmissions of the request for generalLingerTimer milliseconds. The default value is 32000.

inviteLingerTimer

After sending an ACK for an INVITE final response, a client cannot be sure that the server has received the ACK message. The client should be able to retransmit the ACK upon receiving retransmissions of the final response for inviteLingerTimer milliseconds. The default value is 32000.

provisionalTimer

The provisionalTimer is set when receiving a provisional response on an Invite transaction. The transaction will stop retransmissions of the Invite request and will wait for a final response until the provisionalTimer expires. If you set the provisionalTimer to 0, no timer is set, and the Invite transaction will wait indefinitely for the final response. The default value is 180000.

cancelGeneralNoResponseTimer

When sending a CANCEL request on a General transaction, the User Agent waits cancelGeneralNoResponseTimer milliseconds before timeout termination if there is no response for the canceled transaction. The default value is 32000.

cancelInviteNoResponseTimer

When sending a CANCEL request on an Invite request, the User Agent waits cancelInviteNoResponseTimer milliseconds before timeout termination if there is no response for the canceled transaction. The default value is 32000.

generalRequestTimeoutTimer

After sending a General request, the User Agent waits for a final response generalRequestTimeoutTimer milliseconds before timeout termination (in this time the User Agent retransmits the request every T1, 2*T1, ... , T2, ... milliseconds). The default value is 32000.

1.110.3 Sample Code

The following example sets the SIP T1 timer to 64 ms.

```
#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]);

/* sip stack cfg support */
SIP_STACK_CFG sip_stack_cfg;
INIT_SIP_STACK_CFG(&sip_stack_cfg);
   virtBoard[bid].sip_stack_cfg = &sip_stack_cfg;

   sip_stack_cfg.retransmissionT1 = 64;
```

1.110.4 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Dialogic® Global Call API in general, refer to the following documents:

- *Dialogic® Global Call API Programming Guide*
- *Dialogic® Global Call API Library Reference*

For features specific to IP technology, refer to the following documents:

- *Dialogic® Global Call IP Technology Guide*

1.111 Specifying Reliable or Non-reliable SIP

With this Dialogic® HMP Software Release, Service Update 172, the application can specify either Reliable or Non-reliable SIP provisional response handling when receiving ingress INVITE messages containing the optional header tag 100rel in the Supported header.

1.111.1 Feature Description

This feature is related to the SIP PRACK Handling Support (RFC 3262) feature. The scope of that feature has been extended to give the application the control to specify either Reliable or Non-Reliable SIP provisional response handling when receiving ingress INVITE messages that contain the Supported header with option tag 100rel. Previously, reliable response handling was the only option available to the application, regardless of the optional Supported header designation.

If the application has enabled PRACK handling and the remote is capable of PRACK handling, then Global Call IP will automatically include Require:100rel" header in the outgoing provisional response except "100 Trying" for SIP inbound calls with Require:100rel header. For SIP inbound calls with Supported:100rel header, Global Call IP will not automatically include Require:100rel unless the application explicitly overrides it on a per call basis.

For SIP inbound calls with Supported:100rel header, the application will receive a GCEV_OFFERED with a GC PARM block with a set ID/parameter ID/value of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY/IP_SIP_PRACK_MANDATORY _OFF. The application can overwrite the IPPARM_SIP_PRACK_MANDATORY value on a per call basis to IP_SIP_PRACK_MANDATORY_ON using the **gc_SetUserInfo( )** with SINGLECALL parameter. This will result in the Global Call IP including a "Require:100rel" header in the outgoing provisional response except "100 Trying".

If the application sets the IPPARM_SIP_PRACK_MANDATORY value to IP_SIP_PRACK_MANDATORY_OFF on a per call basis using the **gc_SetUserInfo( )** function, then the Global Call IP will not include "Supported:100rel" nor "Require:100rel" header in the outgoing provisional response.

Note: When PRACK handling is enabled, all response messages for Ingress INVITE without a "100rel" header will contain the "Supported:100rel".

The following Global Call API functions are used to support this feature:

- **gc_SipPrack( )** sends a PRACK message.
- **gc_SipPrackResponse( )** sends a PRACK response message.

These functions are documented in Section 1.125.2, "New API Library Functions and Data Structures", on page 228.

1.111.2 Code Example

The following example demonstrates how to make PRACK handling mandatory or optional for inbound SIP:

```
Static void process_event(struct channel *pline, METAEVENT metaevent)
{
int eventtype;
int callindex;
eventtype = metaeventp->evttype;
```

```
callindex = metaeventp->crn;

switch (pline->call_state)
{
   case GCST_NULL:

   {
     switch (eventtype)
{
   case GCEV_OFFERED:
   {
      setPrackForInboundCall(1); // Mode=1 for mandatory PRACK
..
..
gc_AcceptCall(   ); // or gc_CallAck()

   }
         break;
      }
..
..
..
    break;
}
..
}


int setPrackForInboundCall(int mode /* 1 = mandatory, 0 = optional */)
{
    GC_PARM_BLK *parmblkp = NULL;

    if(mode == 1)
    {
/* Mandatory PRACK mode */
gc_util_insert_parm_val(&parmblkp, IPSET_CALLINFO,
IPPARM_SIP_PRACK_MANDATORY,
sizeof(char), IP_SIP_PRACK_MANDATORY_ON);
    }
    else
    {
/* Optional PRACK mode */
gc_util_insert_parm_val(&parmblkp, IPSET_CALLINFO, IPPARM_SIP_PRACK_MANDATORY,
                        sizeof(char), IP_SIP_PRACK_MANDATORY_OFF);
    }

    if(gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, parmblkp, GC_SINGLECALL) !=
                      GC_SUCCESS)
    {
        printf("gc_SetUserInfo(linedev=%ld) Failed configuring PRACK mode for inbound
               call", port[index].ldev);
return(-1)
    }

    gc_util_delete_parm_blk(parmblkp);

    return(0);
}
```

1.112 AMD Machine Support for IP-only Configurations

With Service Update 170, the Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports AMD machines for IP-only configurations. The following systems and licenses were tested:

Dual socket dual core Opteron 285 2.6Ghz

- 240R-240V-120M-240I

Dual socket dual core Opteron 2220SE 2.8Ghz

- 240R-120E-240V-32C-16F-240S
- 500r500v200e500c240s0f0i0m_host_pur.lic
- 240r240v120e0c0s0f240i120m_host_pur.lic

Dual socket dual core Opteron 8220 2.8GHz

- 500r500v200e500c240s0f0i0m_host_pur.lic
- 240r240v120e0c0s0f240i120m_host_pur.lic

1.113 Receive-only RFC 2833 Mode Available

This Service Update 170 allows users to specify a receive-only RFC 2833 mode. When this mode is specified, the additional audio transmission delay associated with the current full duplex RFC 2833 mode, which affects the transmitted audio RTP packets, is eliminated. The additional audio transmission delay exhibited by the current full duplex RFC 2833 mode, is the result of a mechanism employed to detect and remove in-band DTMF digits from the audio stream prior to the transmission of audio RTP packets. For information about this feature, refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

1.114 Selective Packet Filtration Method

With this Dialogic® HMP Software Release, Service Update 165, the application has the ability to filter incoming RTP data based on the remote IP address and RTP port information setup for the RTP session in use. For information about this feature, refer to the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

1.115 Native Streaming Support for Audio and Video

This Service Update 155 provides native streaming support for audio and video as a multimedia stream.

1.115.1 Feature Description

The application can use IPM and MM devices to stream audio and video natively. This allows two IPM devices to be connected together, enabling the audio and video stream to pass through Dialogic® HMP without being transcoded. This feature also allows an IPM device to be connected to an MM device, and have the MM device record and playback the audio and video stream without transcoding.

To use this feature, the application should start the IPM device using the correct coder type for this feature. Currently, the coder type used for this feature is CODER_TYPE_G726_40K_NATIVE (which will pass AMR packets). In a future release the following coder types will be available:

```
CODER_TYPE_AMRNB_4_75k_NATIVE
CODER_TYPE_AMRNB_5_15k_NATIVE
CODER_TYPE_AMRNB_5_9k_NATIVE
CODER_TYPE_AMRNB_6_7k_NATIVE
CODER_TYPE_AMRNB_7_4k_NATIVE
CODER_TYPE_AMRNB_7_95k_NATIVE
CODER_TYPE_AMRNB_10_2k_NATIVE
CODER_TYPE_AMRNB_12_2k_NATIVE
```

Next, the application connects the devices together using the **dev_portConnect( )** function. For information about using the **dev_portConnect( )** function, refer to the *Dialogic® Device Management API Library Reference* found at *http://www.dialogic.com/manuals/docs/device_mgmt_api_v6.pdf*.

1.115.2 New Audio Codec AMR Values

To record or playback the natively recorded streams, calls to the **mm_record( )** and **mm_play( )** functions should no longer use the MM_DATA_FORMAT_PCM coding type in the MM_AUDIO_CODEC structure, but instead use one of the following acceptable AMR values:

```
MM_DATA_FORMAT_AMR_NB_4_75K
MM_DATA_FORMAT_AMR_NB_5_15K
MM_DATA_FORMAT_AMR_NB_5_90K
MM_DATA_FORMAT_AMR_NB_6_70K
MM_DATA_FORMAT_AMR_NB_7_40K
MM_DATA_FORMAT_AMR_NB_7_95K
MM_DATA_FORMAT_AMR_NB_10_20K
```

1.116 License Verification

Additional license checking features are added to this Service Update 155, and require you to upgrade your license. Be sure that you are using the number of resources allocated by the license you purchased.

1.117 Retrieving FlexLM-based Licensed Feature Data

With this Service Update 153, an application operating under the third-party Call Control (3PCC) mode, can retrieve information about licensed features existing on the Dialogic® platform before calling the **gc_Start( )** function.

1.117.1 Feature Description

This feature provides the ability to retrieve the exact number of IP Call Control (IPT) devices licensed and available so that the application can pass a number less than or equal to this value to the **gc_Start( )** function. This ability to do this avoids an IPCCLIB startup failure.

The IP Call Control library needs to know the number of H.323 and SIP IPT devices to open on behalf of a IP Call Control user application. Previously, this information passed to the user application via the IPCCLIB_START_DATA structure when calling the **gc_Start( )** function. In first-party Call Control (1PCC) mode, there is an implicit one-to-one correlation between the number of IPT devices (controlled by IPCCLIB) and the number of RTP or IPM devices (controlled by the firmware and licensing scheme). If the application attempted to enable more IPT devices than the number of licensed IPM devices, an error would return indicating a non-associated IPM device.

In 3PCC mode, there is no one-to-one correspondence between IPT and IPM devices. Instead, the application manages the RTP or media streams for associated IP Call Control devices. The RTP devices can be non-Dialogic, so the application is allowed to open more IPT devices than licensed IPM devices. If IP CCLIB checks the number of IPT devices passed into the **gc_Start( )** function against the number of IP Call Control licenses available on the Dialogic® platform, the library fails to start if the application tries to open more IPT devices than the number of licenses available. This feature resolves this limitation.

1.117.2 New Header File, Defines, and Data Structures

This section introduces the changes required to provide the ability to retrieve licensed feature information from the associated Dialogic® Platform, while insulating the data from the FlexLM licensing header files.

A new header file, *LicensedFeatures.h*, is added to the Dialogic® System Release and contains the following defines:

```
#define LM_FEATURE_NAME_BASIC_VOICE   "Voice"
#define LM_FEATURE_NAME_ENHANCED_VOICE "Enhanced_RTP"
#define LM_FEATURE_NAME_CONFERENCE    "Conferencing"
#define LM_FEATURE_NAME_CSP           "Speech_Integration"
#define LM_FEATURE_NAME_FAX           "Fax"
#define LM_FEATURE_NAME_RTP           "RTP_G_711"
#define LM_FEATURE_NAME_MM            "Multimedia"
```

```
#define LM_FEATURE_NAME_IPCC          "IP_Call_Control"
#define LM_FEATURE_NAME_AMR           "AMR_NB"
#define LM_FEATURE_NAME_NA            "Native_Audio"
#define LM_FEATURE_NAME_NRTP          "Native_RTP"
#define LM_FEATURE_NAME_MUX3G         "3G-324M"
```

The following defines and data structures are added to the *devmgmt.h* header file:

```
/* FlexLM license info retrieval defines and data structures */

#define MAX_FEATURES     256 /* Max features licensed */
#define LICENSED_FEATURES_VERSION 0x000 /* License Structure version */

typedef struct
{
    LicenseFeatureEnum FeatureEnum;      /* Licensed Feature Enum        */
    unsigned short     FeatureCount;     /* Licensed Feature Total Count */
    unsigned short     rfu;              /* For future use               */
}LIC_FEATURE_DATA;

/*
   Data structure used to return block of flexlm licensed features info.
*/
typedef struct
{
    unsigned long      version;        /* Licensing structure version */
    unsigned int       NumFeatures;    /* Number of features info returned. */
    LIC_FEATURE_DATA   lic_data_buf[MAX_FEATURES];/* License data buffer   */
}LIC_FEATURE_BLK, *PLIC_FEATURE_BLK;

static __inline void INIT_LICENSED_FEATURES_VERSION(PLIC_FEATURE_BLK plic_data)
{
   plic_data->NumFeatures = 0;
   plic_data->version = LICENSED_FEATURES_VERSION;
}
```

1.117.3 Code Example

In this example, the application first initializes the data structure in which the licensing information is returned, and then passes a pointer to the LIC_FEATURE_BLK structure to the **dev_getLicFeatureData( )** function. The returned LIC_FEATURE_BLK data structure contains the number of actual licensed features existing on the Dialogic® platform, as well as each licensed feature's name and count (number of instances).

```
#include "devmgmt.h"

int main()
{
 int rc;
 unsigned long lic_msk;
 LIC_FEATURE_BLK lic_data;
 INIT_LICENSED_FEATURES_VERSION(&lic_data);
 rc = dev_GetLicFeatureData(&lic_data);
 if (rc != 0)
 {
  printf("Could not get licensed feature information...Feature not supported or FlexLM
          license file is not active\n");
 }
 else
 {
  for (unsigned int i = 0; i < lic_data.NumFeatures; i++)
  {
   printf("\n*****************************");
```

```
   printf("\n  Licensed Feature Name: %s",
          FeatureNameTable[lic_data.lic_data_buf[i].FeatureEnum]);
   printf("\n Licensed Feature Count: %d ",lic_data.lic_data_buf[i].FeatureCount);

   if ( 0 == strcmp(FeatureNameTable[lic_data.lic_data_buf[i].FeatureEnum],
      LM_FEATURE_NAME_IPCC))
    printf("\n Number of 3PCC IP Call Control devices licensed =
          %d ",lic_data.lic_data_buf[i].FeatureCount);
  }
 }
}
```

1.118 Method for Enabling the ATDX_BUFDIGS( ) Function

This Dialogic® HMP Software Release, Service Update 150, provides a method for enabling **ATDX_BUFDIGS( ).** The Voice API Extended Attribute function, **ATDX_BUFDIGS( )** returns the number of digits in the firmware since the last **dx_getdig( )** request for a given channel. Although the **ATDX_BUFDIGS( )** function is not supported on Dialogic® PowerMedia™ HMP for Windows Release 3.0, it will work as designed if you set the "SupportForSignalCount" registry key.

To enable the **ATDX_BUFDIGS( )** function on Dialogic® PowerMedia™ HMP for Windows Release 3.0, add a "1" to the Data field of the registry key. For example:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Dialogic\Cheetah
Add a Key
Name: SupportForSignalCounting
Type:  REG_DWORD
Data:  1
```

1.119 Enabling Echo Cancellation with Non-Linear Processing

With this Service Update 144, the application has the ability to enable or disable echo cancellation with Non-Linear Processing (NLP) on a per channel basis using the Dialogic® Global Call API.

Note: This feature is currently supported on the Dialogic® DNI601TEPHMP, DNI2410TEPEHMP, DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards. Refer to the Support for Dialogic® DNI/310TEPEHMP, DNI/610TEPEHMP, and DNI/1210TEPEHMP Digital Network Interface Boards and Support for Dialogic® DNI2410TEPEHMP Digital Network Interface Board sections for additional board information.

1.119.1 Feature Description

This feature allows the application to dynamically set echo cancellation parameters, such as NLP and enabled/disabled, at run time. The application uses the existing Global Call

APIs **gc_SetConfigData( )** to set (enable/disable) a PSTN channel's echo canceller and **gc_GetConfigData( )** to query its current value for a given line device (channel basis).

New Parameter ID and Values

The following definitions are used for enabling and disabling echo cancellation:

```
Set ID: CCSET_DM3FW_PARM
Parm ID: CCPARM_ECHOCANCEL
Valid Values: CCDM3FW_PARMECHOCANCEL_DISABLE 0x0
                  CCDM3FW_PARMECHOCANCEL_ENABLE 0x1
                   CCDM3FW_PARMECHOCANCEL_ENABLE_NLP 0x2
```

The target type is GCTGT_CCLIB_CHAN, where the target value is the linedev handle.

Using the CCSET_DM3FW_PARM set ID and the new echo cancellation defines, the application can set up the PSTN channels with echo cancellation capability. These echo cancellation values can be combined as a bitmask to create the following modes of echo cancellation.

Parameter ID	Value	Description
CCDM3FW_PARMECHOCANCEL_DISABLE	0x0	No echo cancellation
CCDM3FW_PARMECHOCANCEL_ENABLE	0x1	Echo cancellation enabled
CCDM3FW_PARMECHOCANCEL_ENABLE | CCDM3FW_PARMECHOCANCEL_ENABLE_NLP	0x3	Echo cancellation enabled, NLP enabled

Note: The following values are not supported and will return an error if used:

```
CCDM3FW_PARMECHOCANCEL_ENABLE_NLP
CCDM3FW_PARMECHOCANCEL_DISABLE|CCDM3FW_PARMECHOCANCEL_ENABLE_NLP
```

Example

```
...
GC_PARM_BLKP echo_blkp = NULL;
int req_id;
...


/* insert parm by value */
if ( gc_util_insert_parm_val( &echo_blkp, CCSET_DM3FW_PARM, CCPARM_ECHOCANCEL, sizeof( char
), (char)(CCDM3FW_PARMECHOCANCEL_ENABLE | CCDM3FW_PARMECHOCANCEL_ENABLE_NLP)) != GC_SUCCESS
) {

     sprintf(str, "gc_util_insert_parm_val(CCSET_DM3FW_PARM, CCPARM_ECHOCANCEL, sizeof( char
     ), (char)CCDM3FW_PARMECHOCANCEL_ENABLE) Failed");
      printandlog(index, GC_APIERR, NULL, str, 0);
      exitdemo(1);
}

/* Enable Echo Cancellation */
if (gc_SetConfigData(GCTGT_CCLIB_CHAN, port[index].ldev, echo_blkp, 0,
GCUPDATE_IMMEDIATE, &req_id, EV_ASYNC) != GC_SUCCESS) {
     sprintf(str, "gc_SetConfigData(GCTGT_CCLIB_CHAN, targetID:0x%x, mode:EV_ASYNC
     Failed", port[index].ldev);
```

```
        printandlog(index, GC_APIERR, NULL, str, 0);
        exitdemo(1);
}
/* delete parm blk */
...
```

1.120 Support for Dialogic® DNI2410TEPEHMP Digital Network Interface Board

With the Service Update 144, Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports the new Dialogic® DNI2410TEPEHMP Digital Network Interface Board. The DNI2410TEPEHMP Board is a high-density, high-performance, digital network interface board with eight T1/E1 network interfaces in a full-length PCI Express form factor.

The use of digital network interface boards with Dialogic® HMP Software enables applications developed using Dialogic® API libraries in an HMP environment to connect to PSTN networks through T1 or E1 network interfaces. The digital network interface boards can be used for converged IP-PSTN solutions that require both IP and PSTN connectivity.

This section provides information about:

- Installing the Dialogic® DNI2410TEPEHMP Digital Network Interface Board
- Installing the Dialogic® HMP Software
- Licensing
- Configuring the Dialogic® DNI2410TEPEHMP Digital Network Interface Board
- Runtime Configuration of Echo Cancellation

1.120.1 Installing the Dialogic® DNI2410TEPEHMP Digital Network Interface 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 equipment using splitter cables.

Note: Refer to the Installation Guide for important information about power budgeting and guidelines for selecting the slot where a board can be installed. If the board is not configured and installed properly, it will not be detected in the system.

1.120.2 Installing the Dialogic® HMP Software

The Dialogic® DNI2410TEPEHMP Digital Network Interface Board requires the use of a Dialogic® HMP Software release that specifically supports it. If you are installing the DNI2410TEPEHMP Board in a system that already has HMP Software installed, you should verify that your installed software version supports the board. If it does not support the board, you will need to obtain and install a Service Update that does support the DNI2410TEPEHMP Board before configuring the system for the newly installed board(s).

Note: When installing the HMP Software, the **Circuit Connectivity Runtime Package** is required when using digital network interface boards. This is one of the install options on the Select Features screen during the install process. If the HMP Software has already been installed without this package, you can add it by using Add/Remove Programs.

For detailed information about installing the HMP Software, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide*.

1.120.3 Licensing

Before you use the Dialogic® HMP Software and the supported digital network interface boards, you must obtain a license file containing HMP license data. 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 supported boards, but you need to obtain a license file before you can proceed with using the HMP Software and supported 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® DNI2410TEPEHMP Digital Network Interface 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 already have a board-based license, you can use it with any new DNI2410TEPEHMP Boards, but you may have to upgrade it.

For detailed information about obtaining and upgrading a license, refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Administration Guide*.

1.120.4 Configuring the Dialogic® DNI2410TEPEHMP Digital Network Interface Board

Once the Dialogic® HMP Software is installed and the appropriate licensing is obtained, you start the configuration process by invoking the configuration manager (DCM). (The Native Configuration Manager (NCM) API can also be used for system configuration.)

Refer to the *Dialogic® Host Media Processing Configuration Guide* for detailed instructions about using DCM to configure HMP Software and the Dialogic® DNI2410TEPEHMP Board.

1.120.5 Runtime Configuration of Echo Cancellation

Refer to the Enabling Echo Cancellation with Non-Linear Processing section for information about runtime configuration of echo cancellation.

1.121 Switching Connections from Full to Half Duplex and from Half to Full Duplex

With this Service Update 144, Dialogic® PowerMedia™ HMP for Windows Release 3.0 provides the ability to add or remove a half duplex stream on a party that is already in a conference.

1.121.1 Feature Description

With this feature, the application can change a party from full duplex to half duplex and back again. To support this feature, the enhancement made to the MSML library was isolated to “joining” and “unjoining” parties to a conference.

1.121.1.1 Changing a Full Duplex Connection to Half Duplex

The following example demonstrates changing the connection to half duplex. The party is added to a conference in full duplex mode, and then modified by removing the stream from the network connection to the conference.

```
<msml version="1.0">
<join id1="conn:31ee908-0-13c4-14c41-159958c4-14c41" id2="conf:exampleconference"
/> </msml>

(conference -> networkconnection)
<msml version="1.0">
<unjoin id1="conn:31ee908-0-13c4-14c41-159958c4-14c41" id2="conf:exampleconference">
<stream media="audio" dir="from-id1">
</stream>
</unjoin>
</msml>
```

1.121.1.2 Changing a Half Duplex Connection to Full Duplex

The following example demonstrates changing the connection to full duplex. The party is added to a conference in half duplex mode, and then modified to add the stream from the network connection to the conference.

```
(conference -> networkconnection)
<msml version="1.0">
<join id1="conn:31ee908-0-13c4-14c41-159958c4-14c41" id2="conf:exampleconference">
<stream media="audio" dir="to-id1">
</stream>
</join>
</msml>

<msml version="1.0">
<join id1="conn:31ee908-0-13c4-14c41-159958c4-14c41" id2="conf:exampleconference">
<stream media="audio" dir="from-id1">
</stream>
</join>
</msml
```

1.121.2 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

1.122 Configure Persistent SIP Header Operations

With this Service Update 133, the application can configure a SIP header, such as a Contact header, for all egress (outgoing) SIP messages regardless of 1PCC or 3PCC mode.

1.122.1 Feature Description

This feature provides the ability to configure persistent SIP headers on all egress (outgoing) SIP messages for all calls or a single SIP session regardless of operational mode. The value of the persistent SIP header will remain in effect until the call is terminated or unless the application changes the value during the call.

Persistent headers are configured on a Line Device or CRN. The duration parameter is used to determine the scope of the configuration. When setting the value of a SIP header, it is assumed that the assigned value's syntax will comply with the requirements specified in the *IETF RFC 3261: SIP: Session Initiation Protocol* and the *IETF RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*. It is the responsibility of the application to ensure compliance.

This feature also allows the application to remove existing persistent headers by header name on the Line Device or CRN. The duration parameter is used to determine the scope of the removal, for example, for all calls or single SIP session. The application may want to set multiple persistent Contact headers for all outgoing messages. If the application wishes to overwrite the previous persistent header configuration, it has to remove that configuration before setting the new one. For example, application may want to change the Contact header only for the current call (CRN), but wish to preserve the persistent Contact header for all other calls on the same line device.

Note: The application is not allowed to set SIP persistent headers across all line devices on a virtual board. Setting SIP headers on virtual board only applies to standalone SIP transactions.

The Global Call library also supports the retrieval of SIP headers for ingress (incoming) messages. Refer to the *Dialogic® Global Call IP Technology Guide* for more information.

1.122.2 New Defines to Set Persistent SIP Headers

The Global Call library allows the application to set SIP generic headers using the **gc_SetUserInfo( )** function on a CRN with GC_SINGLECALL duration. Using this method, only the next outgoing SIP message will consume the configured header. This can be done on all outgoing SIP messages within the call.

The following table lists the new defines to set persistent SIP headers.

Define	Type	Comment
GC_SINGLE_SIP_SESSION	Duration define	Duration field value set in **gc_SetUserInfo( )** for a single SIP session.
IPPARM_SIP_HDR_REMOVE	CClib Parameter ID	Used to remove persistent SIP header configuration by header name.

GC_ALLCALLS duration

- Used when the application desires to use the same SIP generic header, such as Contact header, for all outgoing SIP messages on all SIP calls from a particular line device.

GC_SINGLE_SIP_SESSION duration

- Used when the application wants to use the same SIP generic header for all outgoing SIP messages within one call only. This one call can be either on CRN (for new or existing calls), or on Line Device (for next established call). Configured SIP headers remain persistent until the call is terminated. The next call on the same Line Device uses the default SIP headers.

1.122.3 Code Examples

This section provides code examples for setting or modifying persistent Contact headers.

Set Persistent Headers on Line Device for All Calls

```
#include “gclib.h”
..
..
GC_PARM_BLK *pParmBlock = NULL;
char ContactHeader[] = "Contact: <sip:user@example.com>";
char ContactHeaderName[] = "Contact";

//First remove existing header configuration
//This is optional

gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_SIP_MSGINFO,
                        IPPARM_SIP_HDR_REMOVE,
                        (char)(strlen(ContactHeaderName)+1),
                        ContactHeaderName);

gc_SetUserInfo(GCTGT_GCLIB_CHAN, ldev, pParmBlock, GC_ALLCALLS);

gc_util_delete_parm_blk(pParmBlock);

pParmBlock=NULL;

//Set persistent Contact header
gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_SIP_MSGINFO,
                        IPPARM_SIP_HDR,
                        (char)(strlen(ContactHeader)+1),
                        ContactHeader);

gc_SetUserInfo(GCTGT_GCLIB_CHAN, ldev, pParmBlock, GC_ALLCALLS);

gc_util_delete_parm_blk(pParmBlock);
```

Set Persistent Headers on Line Device for Single SIP Session

```
#include “gclib.h”
..
..
GC_PARM_BLK *pParmBlock = NULL;
char ContactHeader[] = "Contact: <sip:user@example.com>";
char ContactHeaderName[] = "Contact";

//First remove existing header configuration
//This is optional

gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_SIP_MSGINFO,
                        IPPARM_SIP_HDR_REMOVE,
                        (char)(strlen(ContactHeaderName)+1),
                        ContactHeaderName);

gc_SetUserInfo(GCTGT_GCLIB_CHAN, ldev, pParmBlock, GC_SINGLE_SIP_SESSION);

gc_util_delete_parm_blk(pParmBlock);

pParmBlock=NULL;

//Set persistent Contact header
gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_SIP_MSGINFO,
                        IPPARM_SIP_HDR,
                        (char)(strlen(ContactHeader)+1),
                        ContactHeader);

gc_SetUserInfo(GCTGT_GCLIB_CHAN, ldev, pParmBlock, GC_SINGLE_SIP_SESSION);

gc_util_delete_parm_blk(pParmBlock);
```

Set Persistent Headers on CRN for Single SIP Session

```
#include "gclib.h"
..
..
GC_PARM_BLK *pParmBlock = NULL;
char ContactHeader[] = "Contact: <sip:user@example.com>";
char ContactHeaderName[] = "Contact";

//First remove existing header configuration
//This is optional

gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_SIP_MSGINFO,
                        IPPARM_SIP_HDR_REMOVE,
                        (char)(strlen(ContactHeaderName)+1),
                        ContactHeaderName);

gc_SetUserInfo(GCTGT_GCLIB_CRN, crn, pParmBlock, GC_SINGLE_SIP_SESSION);

gc_util_delete_parm_blk(pParmBlock);

pParmBlock=NULL;

//Set persistent Contact header
gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_SIP_MSGINFO,
                        IPPARM_SIP_HDR,
                        (char)(strlen(ContactHeader)+1),
                        ContactHeader);

gc_SetUserInfo(GCTGT_GCLIB_CRN, crn, pParmBlock, GC_SINGLE_SIP_SESSION);

gc_util_delete_parm_blk(pParmBlock);
```

1.122.4 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Dialogic® Global Call APIs, refer to the following document:

- *Dialogic® Global Call API Programming Guide*

1.123 SIGTRAN Signaling Support for Global Call SS7

With this Service Update 130, Dialogic® Global Call SS7 is extended to support SIGTRAN signaling configurations. Now, Global Call application designs can take advantage of the latest SIGTRAN software.

Refer to the *Dialogic® Global Call SS7 Technology Guide* for more information about the SS7 protocol.

1.124 New Processor Support

With this Service Update 130, Dialogic® PowerMedia™ HMP for Windows Release 3.0 provides support for the Core Intel Xeon Processor 53xx.

1.125 SIP PRACK Handling Support (RFC 3262)

With this Service Update 128, the Provisional Response ACKnowledgement (PRACK) method is supported in 3PCC mode. Now, the application can manage reliable provisional response-related SIP messages.

1.125.1 Feature Description

This feature provides the reliable transmit of a provisional response in the public SIP network. As request/response protocol for initiating and managing communication sessions, SIP uses provisional and final responses. Final responses are sent reliably and convey the result of request processing. Provisional responses provide information on the progress of request processing, but are not sent reliably in RFC.

The purpose of this feature is to handle PRACK in SIP using the Global Call API for IP. This involves the following tasks:

- Enable PRACK handling.
- Send PRACK-related messages out.
- Receive PRACK-related messages and extract the PRACK-related information
- Make PRACK message handling mandatory or optional for the remote endpoint in outbound calls.

Notes:1. This feature is supported in 3PCC mode only.

2. Refer to the RFC 3262 for the Offer/Answer Model with PRACK messages.

1.125.2 New API Library Functions and Data Structures

This section contains information about changes and additions to the Global Call API Library.

New Functions

The following new functions support the PRACK handling feature:

- **gc_SipPrack( )** sends a PRACK message.
- **gc_SipPrackResponse( )** sends a PRACK response message.

Name: int gc_SipPrack(crn, parmblk, mode)

Inputs: CRN crn
- call reference number

GC_PARM_BLKP parmblk
- pointer to an optional parameter block containing SDP content for the SIP PRACK message

unsigned long mode
- completion mode (EV_ASYNC)

Returns: 0 if successful
<0 if unsuccessful

Includes: gclib.h

Category: third-party call control

Mode: Asynchronous

Description

This SIP protocol specific function is used in third-party call control (3PCC) mode to send a PRACK message to the remote party of an outbound INVITE or re-INVITE call.

Parameter	Description
crn	specify a call reference number
parmblk	points to an optional parameter block containing SDP content for the SIP PRACK message. This is set to NULL if no SDP content is included in the outbound PRACK message.
mode	set to EV_ASYNC for asynchronous execution.

Termination Events

The termination events for this function are:

GCEV_SIP_PRACK_OK
Indicates that the PRACK event was sent successfully.

GCEV_SIP_PRACK_FAILED
Indicates that the PRACK message could not be sent because the state was invalid to call this function.

Example

Refer to the Code Examples section.

Name: int gc_SipPrackResponse(crn, parmblk, mode)

Inputs: CRN crn
- call reference number

GC_PARM_BLKP parmblk
- pointer to an optional parameter block containing SDP content for the SIP PRACK message

unsigned long mode
- completion mode (EV_ASYNC)

Returns: 0 if successful
<0 if unsuccessful

Includes: gclib.h

Category: third-party call control

Mode: Asynchronous

Description

This SIP protocol specific function is used in third-party call control (3PCC) mode to send a PRACK response message to the remote party of an outbound INVITE or re-INVITE call.

Parameter	Description
crn	specify a call reference number
parmblk	points to an optional parameter block containing SDP content for the SIP PRACK message. This is set to NULL if no SDP content is included in the outbound PRACK message.
mode	set to EV_ASYNC for asynchronous execution.

Termination Events

The termination events for this function are:

GCEV_SIP_PRACK_RESPONSE_OK
Indicates that the PRACK response was sent successfully.

GCEV_SIP_PRACK_RESPONSE_FAILED
Indicates that the PRACK response could not be sent because the state was invalid to call this function.

Example

Refer to the Code Examples section.

New Events

The following new events are added to support PRACK handling:

GCEV_SIP_PRACK
Indicates arrival of a SIP PRACK message.

GCEV_SIP_PRACK_RESPONSE
Indicates arrival of a SIP PRACK response message.

New Data Elements

This new field for the IP_VIRTBOARD data structure can be adjusted for each virtual board:

E_SIP_PrackEnabled
Enables PRACK method. By default, PRACK is disabled. A value of ENUM_Enabled means that the application wants to enable the PRACK handling.

New SIP IP Parameters

The following SIP IP parameters have been added via the Global Call API for this feature.

IPPARM_SIP_PRACK_MANDATORY
This parameter is used as a parameter ID, along with a set ID of IPSET_CALLINFO, when PRACK handling is enabled. It specifies whether the application wants to make PRACK handling mandatory or optional for the outbound SIP call. It also indicates if the remote endpoint has made the PRACK handling mandatory or optional for the SIP inbound call.

IP_ SIP_PRACK_MANDATORY_ON
When PRACK handling is enabled, this parameter is used as a value for the set ID/parameter ID of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY to indicate that PRACK handling is mandatory.

IP_ SIP_PRACK_MANDATORY_OFF
When PRACK handling is enabled, this parameter is used as a value for the set ID/parameter ID of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY to indicate that PRACK handling is optional.

1.125.3 Enabling PRACK Handling

To enable a PRACK handling for SIP in 3PCC mode, the application sets the E_SIP_PrackEnabled field to ENUM_Enabled. The IP_VIRTBOARD structure is then passed to the **gc_Start( )** function. A set ID/parameter ID of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY is specified in the GCEV_OFFERED and GCEV_ALERTING events parm block.

A set ID/parameter ID/value of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY/IP_SIP_PRACK_MANDATORY_ON indicates that the remote endpoint has made the PRACK handling mandatory.

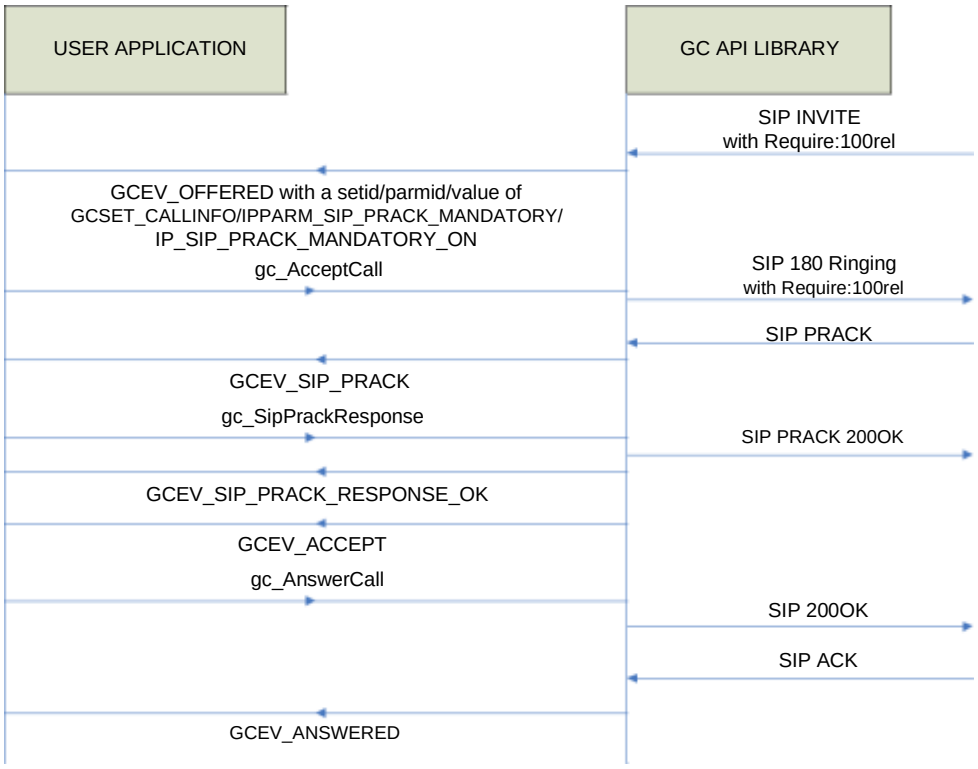
A set ID/parameter ID/value of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY/IP_SIP_PRACK_MANDATORY_OFF indicates that the remote endpoint has made the PRACK handling optional.

When the set ID/parameter ID combination is not present, then the remote endpoint is unable to handle PRACK messages. The call will proceed without any PRACK message exchanges.

1.125.3.1 PRACK Enabled and Remote Endpoint is Capable

If the application enables PRACK handling, and the remote endpoint is capable, the application automatically includes “Require:100rel” header in the outgoing provisional response, but excludes “100 Trying”. To send out the provisional response other than “100 Trying,” the application calls **gc_AcceptCall( )**. This causes the remote endpoint to respond with a PRACK message.

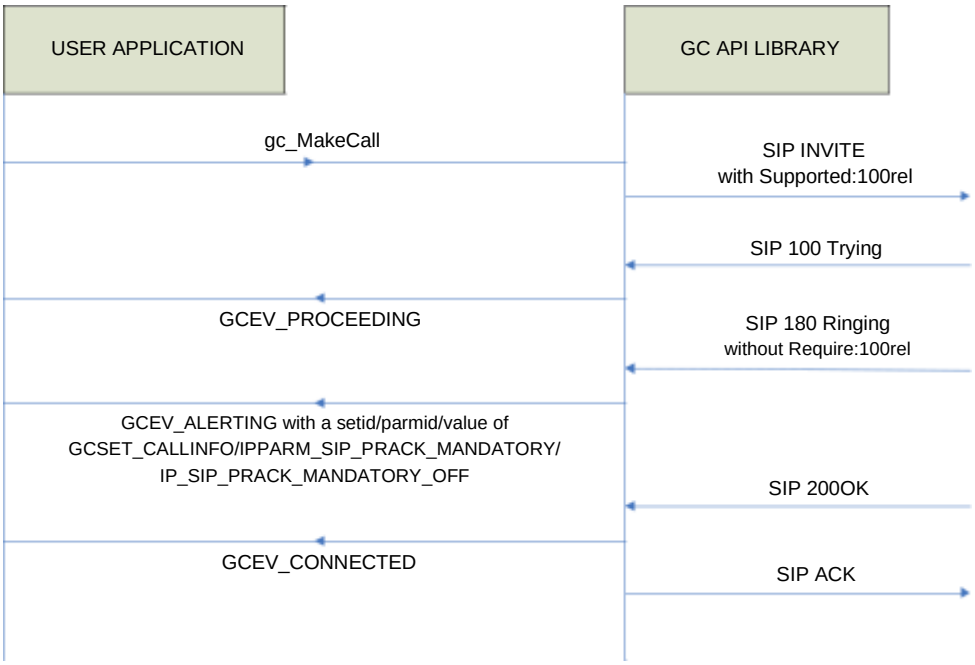
Upon receipt of a PRACK request, the application is notified with a GCEV_SIP_PRACK event. The application calls **gc_SipPrackResponse( )** to send out a PRACK response to the remote endpoint. If the PRACK response is successfully sent, a GCEV_SIP_PRACK_RESPONSE_OK is returned. A return of GCEV_SIP_PRACK_RESPONSE_FAILED indicates a failure in sending out the PRACK response.



1.125.3.2 PRACK Enabled and Remote Endpoint is Optional

If PRACK handling is enabled for outbound calls, the API will automatically include “Supported:100rel” header in the outgoing INVITE message by default making PRACK handling optional for the remote end point.

Note: The application can overwrite this default behavior on a per board basis using **gc_SetConfigData( )** or on a per call/channel basis using **gc_SetUserInfo( )** by changing the value of set ID/parameter ID of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY. A value of IP_SIP_PRACK_MANDATORY_ON will make the PRACK handling mandatory for the remote endpoint by putting “Require:100rel” header in an outgoing INVITE message and a value of IP_SIP_PRACK_MANDATORY_OFF will make the PRACK handling optional for the remote endpoint by putting “Supported:100rel” header in an outgoing INVITE message.



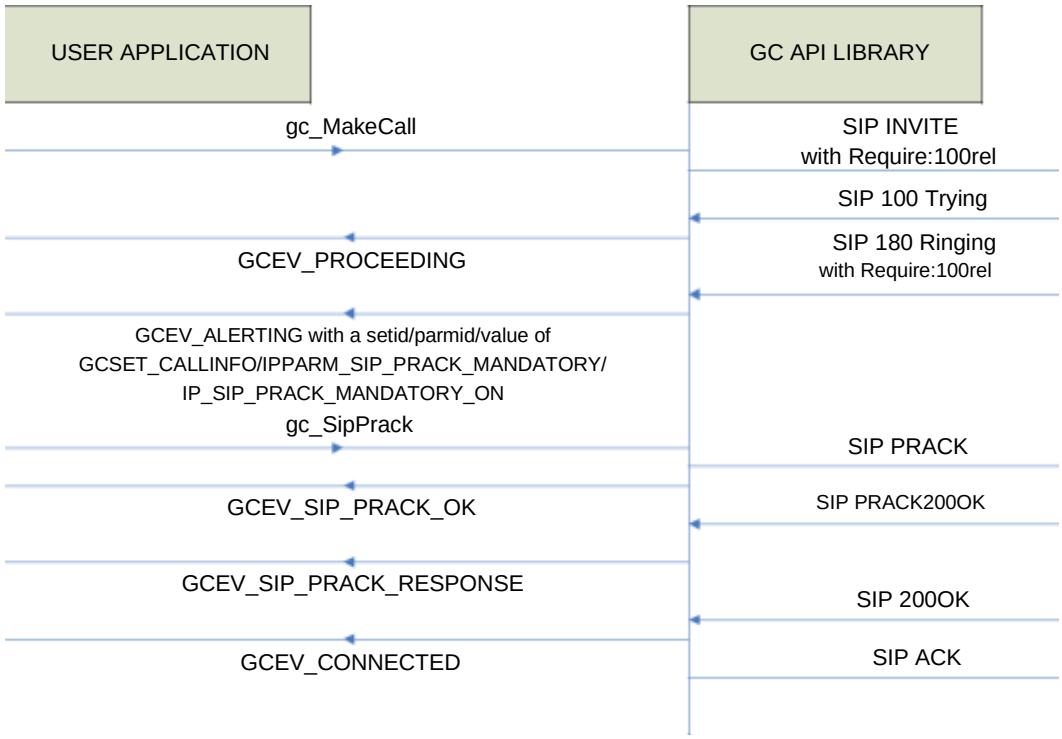
1.125.3.3 PRACK Enabled and Remote Endpoint is Mandatory

Upon receiving a provisional response other than “100 Trying”, the API will generate a GCEV_ALERTING event to the application with a set ID/parameter ID/value of IPSET_CALLINFO/IPPARM_SIP_PRACK_MANDATORY/IP_SIP_PRACK_MANDATORY_ON if “Require:100” header is present in the incoming provisional response (excluding “100 Trying”).

The application calls **gc_SipPrack( )** to send out PRACK for this provisional response. The application is notified with either a GCEV_SIP_PRACK_OK for successfully sending

out a PRACK or with a GCEV_SIP_PRACK_FAILED in case of failure in sending out a PRACK.

When the remote endpoint responds with an ACK response, the API generates a GCEV_SIP_PRACK_RESPONSE to the application.



1.125.3.4 Remote Endpoint Does Not Support PRACK

When the application sends an INVITE with “Require:100rel” to a remote site that does not support PRACK, the **gc_MakeCall( )** function fails and a GCEV_DISCONNECTED event is returned. The reason stated in the event is a bad extension.

1.125.4 Code Examples

Example A

The following example demonstrates how to enable PRACK handling.

```
CCLIB_START_STRUCT cclibStartStruct[] = {
{"GC_IPM_LIB", NULL},
"GC_H3R_LIB", &cclibStartData}
};
GC_START_STRUCT gcStartStruct;
IPCCLIB_START_DATA cclibStartData;
IP_VIRTBOARD virtBoards[1];

memset(&cclibStartData,0,sizeof(IPCCLIB_START_DATA));
```

```
memset(virtBoards,0,sizeof(IP_VIRTBOARD));

INIT_IPCCLIB_START_DATA(&ipcclibstart, 1, ip_virtboard);
INIT_IP_VIRTBOARD(&virtBoards[0]);

cclibStartData.num_boards = 1;
cclibStartData.board_list = virtBoards;
cclibStartData.version = 0x201;

//Set the mode as 3PCC. Prack handling can only be possible for 3PCC mode.
cclibStartData.media_operational_mode = MEDIA_OPERATIONAL_MODE_3PCC;

//Set the flag to enable the PRACK handling
virtBoards[0].E_SIP_PrackEnabled = ENUM_Enabled;

gcStartStruct.cclib_list = cclibStartStruct;
gcStartStruct.num_cclibs = GC_ALL_LIB;
gc_Start(&gcStartStruct);
```

Example B

Code to see if remote endpoint is interested in doing the PRACK handling.

```
void findoutIfRemoteIsInterestedInPRACK(METAEVENT metaevent,
GC_PARM_BLKP parm_blk)
{
  int remoteNotInterestedinPrackHandling = 1;

  switch (metaevent.evttype)
  {
       case GCEV_ALERTING:
       case GCEV_OFFERED:
         {
             GC_PARM_DATA_EXT parm;
             int rc;

             INIT_GC_PARM_DATA_EXT(&parm);

             rc = gc_util_next_parm_ex(parm_blk,&parm);
             if (rc != 0)
             {
                 printf("Remote endpoint is not interested in PRACK
                 handling.\n"); return;
             }

             while (rc != EGC_NO_MORE_PARMS)
             {
               switch (parm.set_ID)
               {
                   case IPSET_CALLINFO:
                     if(parm.parm_ID == IPPARM_SIP_PRACK_MANDATORY)
                     {
                          if(*parm.pData == SIP_PRACK_MANDATORY_ON)
                          {
                             printf("Remote endpoint has made the PRACK handling
                             mandatory.\n"); remoteNotInterestedinPrackHandling = 0;
                          }
                          else
                          if(*parm.pData==SIP_PRACK_MANDATORY_OFF)
                          {
                             printf("Remote endpoint has made the PRACK handling
                             optional.\n"); remoteNotInterestedinPrackHandling = 0;
                          }
                     }
                     break;
```

```
                default:
                  break;
             }
             rc = gc_util_next_parm_ex(parm_blk,&parm);
            }
         }
         break;

       default:
         return;
         break;
    }

    if(remoteNotInterestedinPrackHandling == 1)
    {
         printf("Remote endpoint is not interested in PRACK handling.\n");
    }
    return;
```

Example C

Code to make PRACK handling mandatory or optional to the remote endpoint for outbound SIP.

```
int setPrackForOutboundCall(int mode /* 1 = mandatory, 0 = optional */)
{
   GC_PARM_BLK *parmblkp = NULL;

   if(mode == 1)
   {

        /* Mandatory PRACK mode */
        gc_util_insert_parm_val(&parmblkp, IPSET_CALLINFO,IPPARM_SIP_PRACK_MANDATORY,
                                sizeof(char), IP_SIP_PRACK_MANDATORY_ON);
    }
    else
    {

        /* Optional PRACK mode */
        gc_util_insert_parm_val(&parmblkp, IPSET_CALLINFO,IPPARM_SIP_PRACK_MANDATORY,
                                sizeof(char), IP_SIP_PRACK_MANDATORY_OFF);
    }

    if(gc_SetUserInfo(GCTGT_GCLIB_CHAN, port[index].ldev, parmblkp, GC_ALLCALLS) != GC_SUCCESS)
    {
       printf("gc_SetUserInfo(linedev=%ld) Failed configuring PRACK mode for outbound
               call", port[index].ldev);
            return(-1)
    }

    gc_util_delete_parm_blk(parmblkp);

    return(0);
}
```

1.125.5 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all Release features including features for application development, configuration, administration, and diagnostics.

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

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

1.126 Aastra Telecom IP Phone Model 480i Support

This Service Update 123 provides support for the Aastra Telecom SIP IP Phone Model 480i.

1.127 SIP Session Timer

With Service Update 123, 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.

1.127.1 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.

1.127.2 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.

1.127.3 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.

1.127.4 Global Call IP Defines

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

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

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

Parameter ID	Data Type & Size	Description
IPSET_SIP_SESSION_TIMER	Global Call Set ID	Set ID used to configure SIP session timer
IPPARM_SESSION_EXPIRES	Global Call Parameter ID Type: int Size: sizeof(int)	Session-Expires timer value in seconds

Parameter ID	Data Type & Size	Description
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.
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.

Parameter ID	Data Type & Size	Description
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.
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.

Parameter ID	Data Type & Size	Description
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.

1.127.5 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;
```

1.127.6 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

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

1.128 Command Line Interface to the its_sysinfo Utility

This Service Update 117 provides a command line interface for invoking `its_sysinfo`, a standalone utility that gathers pertinent system data.

The Dialogic® Telecom Subsystem Summary Tool (its_sysinfo) provides a simple way to collect information about systems built using Dialogic® telecom products. The its_sysinfo tool collects data from the system on which you execute it and provides you with information about the system environment: the operating system, computer architecture, Dialogic® HMP software, and operational logs.

The its_sysinfo tool also enables you to collect baseline information about the system for quick review of configuration issues when determining system configuration consistency. This information is collected in a file, compressed, and archived as part of the complete system information collection in an archive file named *its_sysinfo.zip* (or a name you specify). If the installed system is configured in such a way that the baseline information is not available, the its_sysinfo tool will indicate “No Information Available.”

In addition to the standard information captured by the its_sysinfo tool, you can manually add files to the archive that is created by *its_sysinfo.exe* after you run the tool so you can preserve additional information that might help with resolving issues.

For a complete description of the its_sysinfo utility, refer to the Dialogic® *Host Media Processing Diagnostics Guide*.

1.128.1 Command Line Interface

On the command line, enter `its_sysinfo` filename where *filename* is the name you want to give the zip file (it can be the complete path). The its_sysinfo tool will collect system information and compress it into the zip file. If you do not specify any filename, then the information gets compressed in a zip file with the default name *its_sysinfo.zip*.

1.128.2 Information Collected by its_sysinfo

The following information is collected under *its_sysinfo.htm*, which is one of the files that is added to the archive:

- **General System Information**
  - **Environment Variables** – information about the operating system environment variables
  - **System Event Logs** – See **Table 1**.
  - **Memory and Processor** – information about the platform’s available and used memory and CPU type and number as of the time you executed the its_sysinfo tool
  - **Operating System** – information about the operating system’s version, service pack, and language
  - **/proc/meminfo file** – Linux only

- I**nformation Specific to** Dialogic® **Telecom Products**
  - **Installed Devices** – its_sysinfo collects information about devices detected in Dialogic® Telecom Subsystem configurations and startup.
  - **Configuration Settings for Installed Devices** – its_sysinfo captures the stored values used by the configuration tool for Dialogic Telecom Subsystem configurations and startup. For more information about the configuration tool, refer to the configuration guide(s) for the system release.
  - **Firmware Files and Versions** – its_sysinfo collects information about the files listing all firmware file names and version numbers.
  - **FCD Files** – its_sysinfo collects information about the *.fcd* files used by the configuration tool for Dialogic Telecom Subsystem configurations and startup. For more information about FCD files, refer to the for the release.
  - **PCD Files** – its_sysinfo collected information for the *.pcd* file used by the configuration tool for Dialogic Telecom Subsystem configurations and startup. For more information about PCD files, refer to the for the release.

The its_sysinfo tool also checks for the log files listed in Table 1 and adds them to the archive.

Note: It is possible to specify a name for some log files. However, its_sysinfo only collects files with default names.

Table 1. Log Files Archived by its_sysinfo

Log Files	Windows	Linux
OA& M Files	$(SystemRoot)\System32\anm_debug.log $(SystemRoot)\System32\anm_trace.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log.0 Or $(SystemRoot)\System32\ClusterPkg.log* $(SystemRoot)\System32\confslot.log %INTEL_DIALOGIC_DIR%\log\ctbb*.log Or $(SystemRoot)\System32\ctbb*.log dlgcInstall.log is in $(TEMP) %INTEL_DIALOGIC_DIR%\log\dlgcsyslogger.log %INTEL_DIALOGIC_DIR%\log\DM3AutoDump.log $(SystemRoot)\System32\dm3bsp.log $(SystemRoot)\System32\dm3fdspdll.log $(SystemRoot)\System32\frustatus.log %INTEL_DIALOGIC_DIR%\log\GenLoad.log %INTEL_DIALOGIC_DIR%\log\merc.log Or $(SystemRoot)\System32\merc.log %INTEL_DIALOGIC_DIR%\log\ncm.ini %INTEL_DIALOGIC_DIR%\log\oam.log %INTEL_DIALOGIC_DIR%\log\Sctsassi.log	$(INTEL_DIALOGIC_DIR)/log/board*.log $(INTEL_DIALOGIC_DIR)/log/clusterpkg.log $(INTEL_DIALOGIC_DIR)/log/clusterpkg.log.* $(INTEL_DIALOGIC_DIR)/log/dlgsyslogger.log $(INTEL_DIALOGIC_DIR)/log/genload.log $(INTEL_DIALOGIC_DIR)/log/iptconf.log $(INTEL_DIALOGIC_DIR)/log/oam.log
Demos	%INTEL_DIALOGIC_DIR%\log\rgademo.log %INTEL_DIALOGIC_DIR%\log\Board[#].log	$(INTEL_DIALOGIC_DIR)/log/rgademo.log $(INTEL_DIALOGIC_DIR)/log/Board[#].log $(INTEL_DIALOGIC_DIR)/log/dlgrhdemo.log

Table 1. Log Files Archived by its_sysinfo (Continued)

Log Files	Windows	Linux
RTF log Files	<Logfile path>/rtflog*.txt <Logfile path> found in $(INTEL_DIALOGIC_DIR)\cfg\RtfConfigwin.xml or $(DLCFGPATH)\rtfconfig.xml	<Logfile path>/rtflog*.txt <Logfile path> specified in $(INTEL_DIALOGIC_DIR)/cfg/RtfConfigLinux.xml
	<Logfile path>/rtflog.txt <Logfile path> specified in $(DLCFGPATH)\rtfconfig.xml	<Logfile path>/rtflog*.txt <Logfile path> specified in $(DLGCFGPATH)/RtfConfigLinux.xml
CASTrace log Files	$(INTEL_DIALOGIC_DIR)\log\CAStrace.log	$(INTEL_DIALOGIC_DIR)/log/CAStrace.log.*
Debug Angel Files	$(INTEL_DIALOGIC_DIR)\bin\DebugAngel.log	$(INTEL_DIALOGIC_DIR)/log/debugangel.*
Pstndiag Trace log Files	$(INTEL_DIALOGIC_DIR)\log\pstndiag.*	$(INTEL_DIALOGIC_DIR)/log/pstndiag.*
IP Protocol Files	$(SystemRoot)\System32\gc_h3r.log Or $(INTEL_DIALOGIC_DIR)\bin\gc_h3r.log $(SystemRoot)\System32\rtvsp1.log Or $(INTEL_DIALOGIC_DIR)\bin\rtvsp1.log $(SystemRoot)\System32\sdplog.txt Or $(INTEL_DIALOGIC_DIR)\bin\sdplog.txt $(SystemRoot)\System32\siplog.txt Or $(INTEL_DIALOGIC_DIR)\bin\siplog.txt **Note:** This is obsolete. IP protocols supports RTF.	$(HOME)/g*.log $(HOME)/rtvsp1.log **Note:** This is obsolete. IP protocols supports RTF.
Dr. Watson Dump and Log Files	$(USERPROFILE)\Local Settings\Application Data\Microsoft\Dr Watson\DrWtsn32.log	
Its_sysinfo logfile files…	its_sysinfo.log	its_sysinfo.log

Notes: 1. Windows: DIALOGIC_DIR=C:\Program Files\Dialogic and SystemRoot=C:\WINNT

2. Linux: DIALOGIC_DIR=/usr/dialogic/ and HOME=/root

1.129 SIP re-INVITE Support for MSML

With Service Update 112, initial Session Initiation Protocol (SIP) re-INVITE support is added to the Media Sessions Markup Language (MSML). Refer to the *Dialogic® MSML Media Server Software User's Guide* for more information.

1.130 HTTP Play and Record Support for MSML

With Service Update 112, the application server has the ability to store media recordings directly to an HTTP server. The application server can also play a recording and/or retrieve MSML/MOML dialog source directly from an HTTP server. Refer to the *Dialogic® MSML Media Server Software User's Guide* for more information.

1.131 Specify SIP Informational Response on a Per Call Level

With Service Update 103, the application has the ability to specify a SIP informational response on a per call level. Prior to this feature, this capability was configurable only at the virtual board level.This feature supports both 1PCC and 3PCC modes.

1.131.1 Feature Description

With this feature, the **gc_AcceptCall( )** function can be used to select which 18x response to send to the originating endpoint on per call basis. This message will generally be either 180 Ringing or 183 Session Progress, although the Global Call library permits any response code in the range of 101 to 199.

The response code can be specified using **gc_SetUserInfo( )** for a given call. Previously, **gc_SetConfigData( )** could only be used to set the response code on a virtual board level. If a value is entered outside the acceptable range, GCEV_ERROR is generated indicating an invalid parameter.

The IPSET_SIP_RESPONSE_CODE parameter set, IPPARM_ACCEPT_RESP_CODE parameter ID, now support setting the response message on a per call basis. The code is set on a per call basis by inserting the IPPARM_ACCEPT_RESP_CODE parameter in a GC_PARM_BLK and calling **gc_SetUserInfo( )**. The response message is sent when the application accepts the call, and is only applicable for a single call instance when using GC_SINGLECALL. Any subsequent calls utilizing that channel revert to the default informational response.This feature can be set for all calls on a specified channel using the existing GC_ALLCALLS define in **gc_SetUserInfo( )**.

Note: The board level default setting is used when the response message is not explicitly set for a given call.

Code Example

The following example shows how to set both board- and channel-level values:

```
void SetSIPAcceptResponseForBoard(long board, long ProgressAccept) // 180 or
183 {

   GC_PARM_BLK *pParmBlockA = NULL;
   int rc;

   GC_INFO errorinfo;

   long request_id = 0;
   long boardDev;

   char boardstr[255];

   sprintf(boardstr, ":N_iptB%d:P_IP", board);

   rc = gc_OpenEx(&boardDev, boardstr, EV_ASYNC, NULL);

   Sleep(10);

   rc = gc_util_insert_parm_val(&pParmBlockA,
               IPSET_SIP_RESPONSE_CODE,
               IPPARM_ACCEPT_RESP_CODE,
               sizeof(unsigned long),
               ProgressAccept);
   if ( rc != 0 )
   {
         gc_ErrorInfo(&errorinfo);
   }
   rc = gc_SetConfigData(GCTGT_CCLIB_NETIF, boardDev, pParmBlockA, 0,
   GCUPDATE_IMMEDIATE, &request_id, EV_ASYNC);
   if (rc != GC_SUCCESS)
   {
   /* handle error */
   }
   if ( rc != 0 )
   {
         gc_ErrorInfo(&errorinfo);
   }
   gc_Close(boardDev);
}


void SetSIPAcceptResponse(long target_id, long ProgressAccept) // 180 or
183 {
  GC_PARM_BLK *pParmBlockA = NULL;
  int rc;

  GC_INFO errorinfo;

  rc = gc_util_insert_parm_val(&pParmBlockA,
              IPSET_SIP_RESPONSE_CODE,
              IPPARM_ACCEPT_RESP_CODE,
              sizeof(unsigned long),
              ProgressAccept);
  if ( rc != 0 )
  {
          gc_ErrorInfo(&errorinfo);
  }
  rc = gc_SetUserInfo(GCTGT_GCLIB_CHAN, target_id, pParmBlockA, GC_ALLCALLS );
```

```
  if ( rc != 0 )
  {
          gc_ErrorInfo(&errorinfo);
  }
}
```

1.131.2 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

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

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

1.132 Access the Transport Layer IP Address of an Inbound SIP Call

With this Service Update 110, the application has the ability to examine the actual Transport Layer IP address of an inbound SIP call. This feature provides a method to determine the authenticity of the originator, and to immediately detect unauthorized requests.

1.132.1 Feature Description

An existing parameter set ID and a new parameter ID are added to the *gcip_defs.h* header file to allow receipt of the actual Transport Layer IP address of an inbound SIP call in a decimal-dotted notation. The parameter ID, when used with the GCEV_OFFERED and GCEV_REQ_MODIFY_CALL events, extracts the remote SIP transport layer address upon receipt of incoming INVITE and re-INVITE requests.

1.132.1.1 Parameter Set ID and New Parameter ID Data Structure Definitions

The following parameter ID is added to the IPSET_CALLINFO parameter set ID:

Parameter ID	Data Type & Size	Description	SIP/ H.323
IPPARM_SIP_TRANSPORT_ ADDR	Type: IP_SIP_TRANSPOR T_ADDR Size: size of (IP_SIP_TRANSPO RT_ADDR)	Parameter ID in IPSET_CALLINFO. This parameter extracts the remote SIP transport layer address upon receipt of incoming INVITE and re-INVITE messages.	SIP

The following data structure is new to the *gcip_defs.h* file:

```
Set ID: IPSET_CALLINFO
Parameter ID: IPPARM_SIP_TRANSPORT_ADDR
Retrieved using the GCEV_OFFERED and GCEV_REQ_MODIFY_CALL events.

#define IPPARM_SIP_TRANSPORT_ADDR         0x19

////////////////////////////////////////////////////////////////////////////
// DESCRIPTION  : SIP remote Transport Address (TA) structure definitions
////////////////////////////////////////////////////////////////////////////

/* Length of the string, representing transport address IP, used by SIP Stack */

/* sizeof("[ffff:ffff:ffff:ffff:ffff:ffff:255.255.255.255] %dd")+'\0' */

#define CCLIB_SIP_TRANSPORT_IPSTRING_LEN 51

typedef enum

{

   eSIP_TRANSPORT_UNDEFINED = -1,  /* undefined transport. */
   eSIP_TRANSPORT_UDP,             /* UDP   */
   eSIP_TRANSPORT_TCP,             /* TCP   */
   eSIP_TRANSPORT_SCTP,            /* SCTP */
   eSIP_TRANSPORT_TLS,             /* TLS   */
   eSIP_TRANSPORT_OTHER            /* not one of the above */
}EnumSipTransport;

typedef enum
{
   eSIP_TRANSPORT_ADDRESS_TYPE_UNDEFINED = -1,
   eSIP_TRANSPORT_ADDRESS_TYPE_IP,
   eSIP_TRANSPORT_ADDRESS_TYPE_IP6
}EnumSipTransportAddressType;


typedef struct {
   unsigned long               version;
   unsigned short              remTAport;
   EnumSipTransport            eremTransportType;
   EnumSipTransportAddressType eremAddrType;
   char                        remTA[CCLIB_SIP_TRANSPORT_IPSTRING_LEN];
}IP_SIP_TRANSPORT_ADDR;

The version in the above structure is currently defined to be 0, in the gcip_defs.h file.

#define IP_SIP_TA_VERSION               0x000
```

1.132.2 Enabling Receipt of SIP Transport Address Information

By default, the underlying SIP stack will not be enabled to retrieve incoming SIP transport address message information elements. The sip_msginfo_mask value to the IP_SIP_TRANSADDR_ENABLE parameter for a particular virtual board must be enabled for each IPT virtual board device prior to calling **gc_Start( )**.

The value of IP_SIP_TRANSADDR_ENABLE is defined in *gcip_defs.h* as follows:

```
#define IP_SIP_TRANSADDR_ENABLE 0x08
```

Note: Currently, the only way to turn off the receipt of the SIP transport address information received by the underlying SIP stack is by stopping the application or by restarting the stack.

Code Example

```
IP_VIRTBOARD virtBoard[MAX_BOARDS];
memset(virtBoard,0,sizeof(IP_VIRTBOARD) * MAX_BOARDS);
bid = 1;
INIT_IP_VIRTBOARD(&virtBoard[bid]);
 // fill up other board parameters
 ….
 virtBoard[bid].localIP.ip_ver = IPVER4;
 virtBoard[bid].localIP.u_ipaddr.ipv4 = (unsigned int) IP_CFG_DEFAULT;
 …
 virtBoard[1].sip_msginfo_mask |= IP_SIP_TRANSADDR_ENABLE;
 //Then use the virtBoard structure in the gc_Start() function appropriately
```

1.132.3 Retrieving SIP Remote Transport Address Layer Information

The application uses the data received with the GCEV_OFFERED (in case of an incoming SIP INVITE message) and GCEV_REQ_MODIFY_CALL (in case of an incoming SIP re-INVITE message) events to retrieve the SIP transport address embedded within the received data.

The following code example demonstrates how to retrieve the incoming SIP message originator's transport address from a GCEV_OFFERED event using the new field-specific parameter ID IPPARM_SIP_TRANSPORT_ADDR. The GC_PARM_BLK structure containing the data is referenced via the extevtdatap pointer in the METAEVENT structure. In this particular scenario, the GCEV_OFFERED event is generated upon receiving a SIP INVITE message.

Code Example

```
#include "gclib.h"
..
..
METAEVENT metaevt;
GC_PARM_DATA_EXT parm_data;
GC_PARM_BLK *pParmBlock = NULL;

/* Get Meta Event */
gc_GetMetaEvent(&metaevt);
switch(metaevt->evttype)
{
…
case GCEV_OFFERED:
  currentCRN = metaevt->crn;
  pParmBlock = (GC_PARM_BLK*)(metaevt->extevtdatap);
  INIT_GC_PARM_DATA_EXT(&parm_data);

  /* going thru each parameter block data*/
```

```
while ((ret = gc_util_next_parm_ex(pParmBlock,&parm_data)) == GC_SUCCESS)
{
 switch (parm_data.set_ID)
 {
   /* Handle SIP message information */
    case IPSET_SIP_MSGINFO:
      switch (parm_data.parm_ID)
      {

      /* Copy Request URI from parameter block */
      /* NOTE: value_size = string length + 1 (for the NULL termination)
      */ case IPPARM_REQUEST_URI:
        printf("\t requestURI = %s\n",(char*)parm_data.pData);
      break;

      case IPPARM_CALLID_HDR:
       printf("\t CALLID_HDR = %s\n",(char*)parm_data.pData);
      break;
     }
    break;

    case IPSET_CALLINFO:
     switch (parm_data.parm_ID)
     {
         case IPPARM_SIP_TRANSPORT_ADDR:
         {
          if(parm_data.data_size == sizeof(IP_SIP_TRANSPORT_ADDR))
          {
          IP_SIP_TRANSPORT_ADDR *pSipTA;

              pSipTA = (IP_SIP_TRANSPORT_ADDR *)parm_data.pData;
              printf("\tGot Sip transport address IP=%s Port=%d, Transport
                       Type=%d, Address Type=%d, version = d\n",
                   pSipTA->remTA,
                   pSipTA->remTAport,
                   pSipTA->eremTransportType,
                   pSipTA->eremAddrType,
                   pSipTA->version);
            }
          }
          break;
        }
      break;
     }
   }
.
.
}
```

1.132.4 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

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

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

1.133 Access ALERTING Progress Indicator Information Element

With this Service Update 110, the application has the ability to set and/or retrieve the Progress Indicator Information Element (PIIE) in an H.323 ALERTING message. The application can also disable an automatic PROGRESS message after an ALERTING message is received.

1.133.1 Feature Description

This feature allows the application to populate and send a new PIIE and include it in the ALERTING response to an incoming call SETUP message. Currently, the Call Control Library automatically sends a PROGRESS message after every ALERTING message. With this feature, a new parameter is defined to optionally disable an automatic PROGRESS message in the virtual board. Once disabled, PROGRESS will not be sent after receipt of an ALERTING message. By default, this feature is disabled and current behavior is preserved.

The Global Call functions **gc_SetUserInfo( )** and **gc_GetMetaEvent( )** are used to set and get PIIE in an ALERTING message. Both functions use the GC_PARM_BLK and GC_PARM_DATA data structures.

The following new parameter ID, IPPARM_H323_AUTO_PROGRESS_DISABLE, is added to disable an H.323 automatic PROGRESS message after ALERTING on a virtual board.

Parameter ID	Data Type & Size	Description	SIP/ H.323
IPPARM_H323_AUTO_PROGRESS_DISABLE	Type: None Size: 0	Parameter ID in IPSET_CONFIG. This parameter is used to disable H323 automatic PROGRESS message after ALERTING on virtual board.	H.323

1.133.2 Enable Access to H.323 IE

In order to access H.323 IE, the IP_H323_MSGINFO_ENABLE must be enabled in h323_msginfo_mask on a virtual board at **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]);

// IP_H323_MSGINFO_ENABLE must be set to activate this feature
virtBoard[bid].h323_msginfo_mask = IP_H323_MSGINFO_ENABLE
```

1.133.3 Disable an Automatic PROGRESS Message

The following code illustrates how to disable an automatic PROGRESS message on a virtual board:

```
#include "gclib.h"
..
..
GC_PARM_BLK *pParmBlock = NULL;
long t = 0;

gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_CONFIG,
                        IPPARM_H323_AUTO_PROGRESS_DISABLE,
                        sizeof(int),
                         0);

// Set config data
gc_SetConfigData(GCTGT_CCLIB_NETIF,
                 boarddev,
                 pParmBlock,
                 0,
                 GCUPDATE_IMMEDIATE,
                 &t,
                 EV_ASYNC);

gc_util_delete_parm_blk(pParmBlock);
```

1.133.4 Setting the PIIE

PIIE is set using the Global Call API **gc_SetUserInfo( )** function, but information is not transmitted until calling **gc_AcceptCall( )**. The application will overwrite the PIIE in any incoming ALERTING or PROGRESS messages for the same channel.

Code Example

```
#include "gclib.h"
..
..
GC_PARM_BLK *pParmBlock = NULL;
unsigned char progress_ind[] = {0x1E,0x02,0x80,0x81};

//set up progress indicator
gc_util_insert_parm_ref(&pParmBlock,
                        IPSET_CALLINFO,
                        IPPARM_PROGRESS_IND,
                        sizeof(progress_ind),
                        progress_ind);
```

```
// Set Call Information
gc_SetUserInfo(GCTGT_GCLIB_CHAN, ldev, pParmBlock, GC_SINGLECALL);

gc_util_delete_parm_blk(pParmBlock);
gc_AcceptCall(crn, NULL, EV_ASYNC);
```

1.133.5 Retrieving PIIE

The PIIE values are reported to the application through events processed using the **gc_GetMetaEvent( )**. The following example illustrates how to retrieve PIIE from the GCEV_ALERTING event after receiving an ALERTING message.

Code Example

```
#include "gclib.h"
..
..
METAEVENT  metaevt;
GC_PARM_BLK *pParmBlock = NULL;
GC_PARM_DATA  *parmp = NULL;

/* Get Meta Event */
gc_GetMetaEvent(&metaevt);

switch(metaevt->evttype){
        .
        .
        .
        case GCEV_ALERTING:
                currentCRN = metaevt->crn;
                pParmBlock = (GC_PARM_BLK*)(metaevt->extevtdatap);
                parmp = NULL;

                /* going thru each parameter block data*/
                while ((parmp = gc_util_next_parm(pParmBlock,parmp)) != 0)
                {
                        switch (parmp->set_ID)
                        {
                        /* Handle the extended information */
                        case IPSET_CALLINFO:
                                switch (parmp->parm_ID)
                                {
                                case IPPARM_PROGRESS_IND:
                                if(parmp->data_size != 0)
                                {

                                        printf("\tGot PIIE: ");
                                        for(unsigned int pii= 0;pii<parmp->data_size;pii++)
                                        printf ("0x%x ",*(((unsigned char*)(parmp->pData))+pii));
                                        printf ("\n");
                                 }
                                 }
                                  break;
                           }
                            break;
                    }
            .
            .
            .
}
```

1.133.6 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Dialogic® Global Call APIs, refer to the following documents:

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

1.134 Support for SS7 Functionality

With the Service Update 110, Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports boards with Signaling System 7 (SS7) functionality. To support the Dialogic® boards, the Dialogic® HMP software provides SS7 board detection and initialization, board configuration and SIU configuration, timeslot allocation/routing; and Global Call development tools. Global Call supports the development of call control applications that use SS7 technology, clear channel support on Dialogic® SS7 boards (under the Global Call API), and handling of alarms on Dialogic SS7 boards.

For more detailed information, refer to the *Dialogic® Global Call SS7 Technology Guide*.

1.135 Enhancements to CNF and DCB Notification Tone

With Service Update 99, the functionality for conferencing (CNF) and audio conferencing (DCB) has been enhanced with new APIs and parameters to provide more control for enabling and disabling conference notification tones after the conference has been established, and to modify the conference notification tone values.

1.135.1 Feature Description

Currently, if you are using the DCB API, you can enable and disable the conference notification tone during conference setup using **dcb_estconf( )**, but these tone notifications cannot be changed once the conference is established. The feature allows you to change the tone notifications after the conference is established by using two new APIs, **dcb_setcnfparm( )** and **dcb_getcnfparm( )**, to set the notification to on or off depending on the current notify state.

Also currently for DCB, the tone notification (that is, tone frequency, amplitude, and length parameters) cannot be modified or changed (it is currently unexposed and always set to default values). The feature allows you to specify the conference tone amplitude (level), frequency and length by adding new parameters in the HMP CONFIG file. These parameters also are valid for CNF.

Note: The tone specification is applicable for all conferences and can only be changed when the HMP system is restarted.

CNF has been modified to include a new parameter for **cnf_SetAttribute( )** and **cnf_GetAttribute( )** to enable or disable the conference notification tone.

1.135.2 Dynamic Conference Tone Setting and Specification for DCB

1.135.2.1 New APIs for DCB

To support this feature, two new APIs for DCB have been added:

- **dcb_setcnfparm( )** - enable/disable the conference level parameters for a specific conference.
- **dcb_getcnfparm( )** - retrieve current value of conference level parameters.

For **dcb_setcnfparm( )**, the conference level parameters are as follows:

- DCB_CNF_NOTIFY - Notify or suspend notification with a short beep when a party is joining/leaving the conference.
- DCB_CNF_DTMF_MASK - Specify dual tone multi-frequency (DTMF) mask (same value can be set via **dcb_setdigitmask( )**).
- DCB_CNF_TONE_CLAMPING - Specify tone clamping switch on and off (DTMF tone clamping). This parameter mutes DTMF tones heard during a conference. Tone clamping applies to the transmitted audio going into the conference and does not affect DTMF function. It can be enabled on a board, conference, or party basis.

1.135.2.2 Change Tone Notification Frequency and Duration

To change the tone notification frequency, amplitude and duration, you must add the parameters listed below to the [0x3b] Conferencing Parameters section of the CONFIG file. Once the CONFIG file parameters are modified, use the fcdgen utility to convert the CONFIG file to an FCD file. Refer to the *Dialogic® Host Media Processing Configuration Guide* for more details.

- NotifyAddToneLevel
- NotifyAddToneFreq
- NotifyAddToneLength

1.135.2.3 Code Examples

DCB Code to Set and Get DCB_CNF_NOTIFY

```
//
// dcb_setattribute
//

#include <iostream>
#include <srllib.h>
#include <dxxxlib.h>
#include <msilib.h>
#include <dcblib.h>

using namespace std;

#define DCB_ERROR (-1)

int dspHandle = DCB_ERROR;
int cnfHandle = DCB_ERROR;
int dxHandle =  DCB_ERROR;

// process error
void process_dcb_error(int handle){
     cout << "Error " << ATDV_LASTERR(handle)
                      << " "
                      << ATDV_ERRMSGP(handle)
                      << endl;
     return;
}

int main(int args, char *argv[]){
 int rc;
 dspHandle = dcb_open("dcbB1D1",0);   // Open board
 cout << dspHandle << " = dcb_open(\"dcbB1D1\",0)" <<
 endl; if (DCB_ERROR == dspHandle){
     process_dcb_error(0);
 } else {
     dxHandle = dx_open("dxxxB1C1",0);
     cout << dxHandle << " = dx_open(\"dxxxB1C1\",0)" << endl;
     if (DCB_ERROR == dxHandle){
         process_dcb_error(0);
     }else {
         SC_TSINFO sc_tsinfo;
         long ts;
         sc_tsinfo.sc_numts = 1;
         sc_tsinfo.sc_tsarrayp = &ts ;

         rc = dx_getxmitslot( dxHandle, &sc_tsinfo);
         if (DCB_ERROR == rc){
             process_dcb_error(dxHandle);
         }else {
             cout << rc << " = dx_getxmitslot("
                  << dxHandle
                  << ", slot = " << ts << endl;

             MS_CDT cdt;
            cdt.chan_num  = ts;       // by Timeslot
           cdt.chan_sel  = MSPN_TS;  // ...
             cdt.chan_attr = MSPA_MODEFULLDUPLX|MSPA_NOAGC;


             rc = dcb_estconf(dspHandle, &cdt, 1, MSCA_ND,
             &cnfHandle); cout << rc << " = dcb_estconf("
                  << dspHandle
                  << ", &cdt,1,MSCA_ND, cnfHandle := " << hex << cnfHandle << ")" <<
             endl; if (DCB_ERROR == rc){
```

```
                process_dcb_error(dspHandle);
            } else {
                unsigned char notify_prm = DCB_CNF_NOTIFY;
                int notify_value = DCB_CNF_NOTIFY_OFF;
                rc = dcb_setcnfparm(dspHandle, cnfHandle, notify_prm,
                &notify_value); cout << rc << " = dcb_setcnfparm("
                     << dspHandle << ", " << hex << cnfHandle << ", "
                     << (int)notify_prm << ", " << notify_value << ")" <<
                endl; if (DCB_ERROR == rc){
                    process_dcb_error(dspHandle);
                }

                rc = dcb_getcnfparm(dspHandle, cnfHandle, notify_prm,
                &notify_value); cout << rc << " = dcb_getcnfparm("
                     << dspHandle << ", "
                     << hex << cnfHandle
                     << ", " << (int)notify_prm
                     << ", returns " << notify_value << ")" << endl;

                if (DCB_ERROR == rc){
                    process_dcb_error(dspHandle);
                }
            }
        }
    }
}

// cleanup

if (DCB_ERROR != cnfHandle) {
    rc = dcb_delconf(dspHandle, cnfHandle);
    cout << rc << " = dcb_delconf("
         << dspHandle << ", " << hex << cnfHandle << ")" << endl;

    if (DCB_ERROR == rc){
        process_dcb_error(dspHandle);
    }
}

if (DCB_ERROR != dxHandle ) {
    dx_close(dxHandle);
    cout << "dx_close(" << dxHandle << ")" << endl;
}

if (DCB_ERROR != dspHandle ) {
    dcb_close(dspHandle);
    cout << "dcb_close(" << dspHandle << ")" << endl;
}

return (0);
}
```

DCB Code to Get and Set Tone Clamping at Conference Level

```
int dspHandle;// handle for this DSP:
// holds result of dcb_Open("dcbB1D1",0);

int confId;// Handle for this conference: holds result of
// dcb_estconf (dspHandle, &cdt, 1, MSCA_ND,

&confId) int rc;// return code
```

```
unsigned char clamping_prm = DCB_CNF_TONE_CLAMPING;
int clamping_value;

clamping_value = TONECLAMP_ON;
rc = dcb_setcnfparm(dspHandle, cnfHandle, notify_prm, &notify_value);

    // Set tone clamping
    unsigned char clamping_prm = DCB_CNF_TONECLAMPING;
    int clamping_value = TONECLAMP_ON;
   rc = dcb_setcnfparn(dspHandle, cnfHandle, clamping_prm, &clamping_value)

if (rc == 0) {
// Tone clamping is on
}

rc = dcb_getcnfparm(dspHandle, cnfHandle, notify_prm, &notify_value);

    // Get tone clamping
    rc = dcb_setcnfparn(dspHandle, cnfHandle, clamping_prm, &clamping_value);
    if (0 == rc){
        //  clamping_value contains current tone clamping parameter
    }
if ( rc == 0 ) {
// clamping_value contains TONECLAMP_ON or
// TONECLAMP_OFF depending of the current state
}
```

DCB Code to Set and Get DTMF Digit Mask

```
int dspHandle;// handle for this DSP:
// holds result of dcb_Open("dcbB1D1",0);

int confId;// Handle for this conference: holds result of
// dcb_estconf (dspHandle, &cdt, 1, MSCA_ND,

&confId) int rc;// return code

unsigned char dtmf_prm = DCB_CNF_DTMF_MASK;
int dtmf_value;

dtmf_value = CBMM_ZERO | CBMM_ONE;
rc = dcb_setcnfparm(dspHandle, confId, dtmf_prm, & dtmf_value);

    // Set digit mask
    unsigned char dtmfmsk_prm = DCB_CNF_TONECLAMPING;
    int dtmfmsk_value = CBMM_ZERO | CBMM_ONE;
    rc = dcb_setcnfparn(dspHandle, cnfHandle, clamping_prm, &clamping_value);

    if (rc == 0) {
// DTMF mask is set for digits 0 and 1
}

rc = dcb_getcnfparm(dspHandle, confId, dtmf_prm, &dtmf_value);

    // Get digit mask
    rc = dcb_setcnfparn(dspHandle, cnfHandle, dtmfmsk_prm,
    &dtmfmsk_value); if (0 == rc){
        //  dtmfmsk_value contains current dtmf mask parameter
    }
if ( rc == 0 ) {
// dtmf_value contains current dtmf mask
}
```

1.135.3 CNF Enhancements at the Conference Level

1.135.3.1 New Parameter

Conferencing (CNF) has been modified to include a new value for ECNF_CONF_ATTR data type for the **cnf_SetAttribute( )** and **cnf_GetAttribute( )** functions:

ECNF_CONF_ATTR_NOTIFY
: sets conference notification tone enabled or disabled. Possible values are ECNF_ATTR_STATE_ENABLED and ECNF_ATTR_STATE_DISABLED.

1.135.3.2 Code Example for Setting Attribute ECNF_CONF_ATTR_NOTIFY

The following code example applies to the **cnf_SetAttribute( )** function:

```
                //
                //  Set Conference Attribute
//

                #include <iostream>
                #include <cnflib.h>

                using namespace std;

                SRL_DEVICE_HANDLE brdHandle = CNF_ERROR;
                SRL_DEVICE_HANDLE cnfHandle = CNF_ERROR;

                // process error
                void process_cnf_error(){
                 CNF_ERROR_INFO err_info;
                  int rc = cnf_GetErrorInfo(&err_info);
                    if (CNF_ERROR != rc ){
                                cout << "Error" << err_info.unErrorCode
                                       << err_info.szErrorString
                                       << err_info.szAdditionalInfo
                                       << endl;
                    } else {
                                cout << "cnf_GetErrorInfo() failed" << endl;
                    }
                    return;
                }

                //
                // returns false to indicate exit
//
                bool process_event(){
                 bool brc = true;  // return code
                 int rc;           // function result
                    int event = sr_getevttype();              // event
                    SRL_DEVICE_HANDLE handle = sr_getevtdev(); // device handle
                    void *evtdata = sr_getevtdatap();         // event data
                    cout << "got event " <<hex << event << endl;
                    switch(event){

                         case CNFEV_OPEN_FAIL:
                              if (handle == brdHandle) {
                                  brc = false;
                                  process_cnf_error();
                                  break;
                              }else {
                                  cout << "Unexpected event handle" << handle << endl;
                              }
```

```
            break;

        case CNFEV_OPEN:
            if (handle == brdHandle) {
                cnfHandle = cnf_OpenConference(brdHandle,0,0,0);
                cout << cnfHandle << " = cnf_OpenConference(" << brdHandle << ",0,0,0)" <<
endl;
                if (CNF_ERROR == cnfHandle) {
                    brc = false;
                    process_cnf_error();
                }
                break;
            }else {
                cout << "Unexpected event handle" << handle << endl;
            }
            break;

        case CNFEV_OPEN_CONF_FAIL:
            if (handle == cnfHandle) {
                brc = false;
                process_cnf_error();
                break;
            }else {
                cout << "Unexpected event handle " << handle << endl;
            }
            break;

        case CNFEV_OPEN_CONF:
            if (handle != brdHandle) {
                cout << "Unexpected event handle " << handle << endl;
            }else {
                PCNF_OPEN_CONF_RESULT pResult =
                static_cast<PCNF_OPEN_CONF_RESULT>(evtdata); cout << "Conference name is "
                << pResult->szConfName << endl; CNF_ATTR CnfAttr[2];
                CNF_ATTR_INFO CnfAttrInfo;
                int inx = 0;
                CnfAttrInfo.unVersion = CNF_ATTR_INFO_VERSION_0; ///< Structure version
                CnfAttrInfo.pAttrList = CnfAttr;               ///< Pointer to attribute
structure list

                CnfAttr[inx].unVersion  = CNF_ATTR_VERSION_0 ;       ///< Structure version
                CnfAttr[inx].unAttribute = ECNF_CONF_ATTR_NOTIFY;    ///< Attribute type
                CnfAttr[inx].unValue    = ECNF_ATTR_STATE_ENABLED;   ///< or
 ECNF_ATTR_STATE_DISABLED
                ++inx;

                // ... set other conference attributes


                CnfAttrInfo.unAttrCount = inx;  ///< Number of attribute structures in list

                rc = cnf_SetAttributes(cnfHandle, &CnfAttrInfo, 0);
                cout << rc << " = cnf_SetAttributes( " << cnfHandle << " cnfInfo, 0)" <<
                endl; if (CNF_ERROR == rc){
                    process_cnf_error();
                    brc = false;
                }
            } // if handle ==
           cnfHandle break;

        case CNFEV_SET_ATTRIBUTE_FAIL:
            if (handle != cnfHandle) {
                cout << "Unexpected event handle " << handle << endl;
            }else {
                brc = false;
                process_cnf_error();
                break;
```

```
            }
            break;

        case CNFEV_SET_ATTRIBUTE:
            if (handle != cnfHandle) {
                cout << "Unexpected event handle " << handle << endl;
            }else {
                brc = false;
                cout << "Attribute was set" << endl;
                break;
            }
            break;
        default:
            cout << "Unexpected event " << handle << endl;
            break;
   }
 return brc;
}

int main(int args, char *argv[]){
 brdHandle = cnf_Open("CnfB1",0, 0);   // Open board
 cout << brdHandle << " = cnf_Open(\"CnfB1\",0,0)" <<
 endl; if (CNF_ERROR == brdHandle){
    process_cnf_error();
 } else {

    bool brc = true;

    // wait events until process_event returns
    true while (brc){
        if( sr_waitevt(0) >= 0 ) {
            brc = process_event();
       }
    } // while

 }

 // cleanup
 if (CNF_ERROR != cnfHandle) {
     cnf_Close(cnfHandle,0);
     cout << "cnf_Close(" << cnfHandle << ")" << endl;
 }

 if (CNF_ERROR != brdHandle ) {
    cnf_Close(brdHandle,0);
    cout << "cnf_Close(" << brdHandle << ")" << endl;
 }

 return (0);
}
```

dcb_setcnfparm( )

Name: int dcb_setcnfparam(hSrlDevice, nConfId, ParameterId, value)

Inputs: int hSrlDevice • DCB SRL device handler, returned from dcb_open

int nConfId • conference Id, returned from dcb_estconf

unsigned char ParameterId • parameter code

void *value • parameter value

Returns: 0 on success
-1 on failure

Includes: dcblib.h

Category: Configuration

Mode: synchronous

Platform: DM3

Description

The **dcb_setcnfparm( )** function changes the conference parameters.

Parameter	Description
hSrlDevice	specifies the valid device handle obtained when the DSP device was opened using **dcb_open( )**
nConfId	specifies the conference identifier number
ParameterId	specifies which parameter to change
value	points to the integer or structure containing the value to be assigned to **ParameterId**. See the list of valid values for a parameter following this table.

The valid values for **ParameterId** and **value** are shown below:

DCB_CNF_NOTIFY
sets conference notification tone on or off. Possible values are DCB_CNF_NOTIFY_ON and DCB_CNF_NOTIFY_OFF.

DCB_CNF_TONE_CLAMPING
sets conference tone clamping on or off. Possible values are TONECLAMP_ON and TONECLAMP_OFF.

DCB_CNF_DTMF_MASK
sets DTMF mask. Possible values are OR-ed combination of CBMM_ZERO, CBMM_ONE, CBMM_TWO, etc.

Cautions

None.

Errors

If this function returns -1 to indicate failure, obtain the reason for the error by calling the Standard Runtime Library standard attribute function **ATDV_LASTERR( )** or **ATDV_ERRMSGP( )** to retrieve either the error code or a pointer to the error description, respectively.

EDT_BADDEV Bad
device error

E_MSINVCNF
Invalid conference number

EDT_PARAMERR
Invalid parameter. This error occurs if you execute an audio conferencing library function on a board that does not support that particular function.

Example

```
      //
// dcb_setattribute
//
#include <iostream>

#include <srllib.h>

#include <dxxxlib.h>
#include <msilib.h>
#include <dcblib.h>

using namespace std;

#define DCB_ERROR (-1)

int dspHandle = DCB_ERROR;
int cnfHandle = DCB_ERROR;
int dxHandle =  DCB_ERROR;

// process error
void process_dcb_error(int handle){
     cout << "Error " << ATDV_LASTERR(handle)
                      << " "
                      << ATDV_ERRMSGP(handle)
                      << endl;
     return;
}


int main(int args, char *argv[]){
 int rc;
 dspHandle = dcb_open("dcbB1D1",0);   // Open board
 cout << dspHandle << " = dcb_open(\"dcbB1D1\",0)" <<
 endl; if (DCB_ERROR == dspHandle){
     process_dcb_error(0);
 } else {
     dxHandle = dx_open("dxxxB1C1",0);
     cout << dxHandle << " = dx_open(\"dxxxB1C1\",0)" << endl;
     if (DCB_ERROR == dxHandle){
         process_dcb_error(0);
     }else {
         SC_TSINFO sc_tsinfo;
         long ts;
         sc_tsinfo.sc_numts = 1;
         sc_tsinfo.sc_tsarrayp = &ts ;
```

```
        rc = dx_getxmitslot( dxHandle, &sc_tsinfo);
        if (DCB_ERROR == rc){
            process_dcb_error(dxHandle);
        }else {
            cout << rc << " = dx_getxmitslot("
                 << dxHandle
                 << ", slot = " << ts << endl;

            MS_CDT cdt;
           cdt.chan_num  = ts;       // by Timeslot
          cdt.chan_sel  = MSPN_TS;  // ...
            cdt.chan_attr = MSPA_MODEFULLDUPLX|MSPA_NOAGC;


            rc = dcb_estconf(dspHandle, &cdt, 1, MSCA_ND,
            &cnfHandle); cout << rc << " = dcb_estconf("
                 << dspHandle
                 << ", &cdt,1,MSCA_ND, cnfHandle := " << hex << cnfHandle << ")" <<
            endl; if (DCB_ERROR == rc){
                process_dcb_error(dspHandle);
            } else {
                unsigned char notify_prm = DCB_CNF_NOTIFY;
                int notify_value = DCB_CNF_NOTIFY_OFF;
                rc = dcb_setcnfparm(dspHandle, cnfHandle, notify_prm,
                &notify_value); cout << rc << " = dcb_setcnfparm("
                     << dspHandle << ", " << hex << cnfHandle << ", "
                     << (int)notify_prm << ", " << notify_value << ")" <<
                endl; if (DCB_ERROR == rc){
                    process_dcb_error(dspHandle);
                }

                rc = dcb_getcnfparm(dspHandle, cnfHandle, notify_prm,
                &notify_value); cout << rc << " = dcb_getcnfparm("
                     << dspHandle << ", "
                     << hex << cnfHandle
                     << ", " << (int)notify_prm
                     << ", returns " << notify_value << ")" << endl;

                if (DCB_ERROR == rc){
                    process_dcb_error(dspHandle);
                }
            }
        }
    }
}

// cleanup

if (DCB_ERROR != cnfHandle) {
    rc = dcb_delconf(dspHandle, cnfHandle);
    cout << rc << " = dcb_delconf("
         << dspHandle << ", " << hex << cnfHandle << ")" << endl;

    if (DCB_ERROR == rc){
        process_dcb_error(dspHandle);
    }
}


if (DCB_ERROR != dxHandle ) {
    dx_close(dxHandle);
    cout << "dx_close(" << dxHandle << ")" << endl;
}

if (DCB_ERROR != dspHandle ) {
    dcb_close(dspHandle);
```

```
        cout << "dcb_close(" << dspHandle << ")" << endl;
    }

    return (0);
}
```

See Also

- **dcb_getcnfparm( )**

dcb_getcnfparm( )

Name: dcb_getcnfparam(hSrlDevice, nConfId, Parameter, value)

Inputs:

int hSrlDevice	• DCB SRL device handler, returned from dcb_open
int nConfId	• conference Id, returned from dcb_estconf
unsigned char Parameter	• parameter code, enum
void *value	• parameter value

Returns: 0 on success
-1 on failure

Includes: dcblib.h

Category: Configuration

Mode: synchronous

Platform: DM3

Description

The **dcb_getcnfparm( )** function retrieves current value of the conference parameters.

Parameter	Description
hSrlDevice	specifies the valid device handle obtained when the DSP device was opened using **dcb_open( )**
int nConfId	specifies the conference identifier number
Parameter	specifies which parameter is requested; currently only DCB_CNF_NOTIFY is supported.
value	points to the integer or structure containing the value to be assigned to **Parameter**. See the list of valid values for a parameter following this table.

The valid values for **Parameter** and **value** are shown below:

DCB_CNF_NOTIFY
sets conference tone on or off. Possible values are DCB_CNF_NOTIFY_ON and DCB_CNF_NOTIFY_OFF.

Cautions

None.

Errors

If this function returns -1 to indicate failure, obtain the reason for the error by calling the Standard Runtime Library standard attribute function **ATDV_LASTERR( )** or **ATDV_ERRMSGP( )** to retrieve either the error code or a pointer to the error description, respectively.

EDT_BADDEV Bad
device error

E_MSINVCNF
Invalid conference number

EDT_PARAMERR
Invalid parameter. This error occurs if you execute an audio conferencing library function on a board that does not support that particular function.

Example

```
//
// dcb_setattribute
//

#include <iostream>

#include <srllib.h>

#include <dxxxlib.h>
#include <msilib.h>
#include <dcblib.h>

using namespace std;

#define DCB_ERROR (-1)

int dspHandle = DCB_ERROR;
int cnfHandle = DCB_ERROR;
int dxHandle =  DCB_ERROR;

// process error
void process_dcb_error(int handle){
     cout << "Error " << ATDV_LASTERR(handle)
                      << " "
                      << ATDV_ERRMSGP(handle)
                      << endl;
     return;
}


int main(int args, char *argv[]){
 int rc;
 dspHandle = dcb_open("dcbB1D1",0);   // Open board
 cout << dspHandle << " = dcb_open(\"dcbB1D1\",0)" <<
 endl; if (DCB_ERROR == dspHandle){
     process_dcb_error(0);
 } else {
     dxHandle = dx_open("dxxxB1C1",0);
     cout << dxHandle << " = dx_open(\"dxxxB1C1\",0)" << endl;
     if (DCB_ERROR == dxHandle){
```

```
        process_dcb_error(0);
    }else {
        SC_TSINFO sc_tsinfo;
        long ts;
        sc_tsinfo.sc_numts = 1;
        sc_tsinfo.sc_tsarrayp = &ts ;

        rc = dx_getxmitslot( dxHandle, &sc_tsinfo);
        if (DCB_ERROR == rc){
            process_dcb_error(dxHandle);
        }else {
            cout << rc << " = dx_getxmitslot("
                 << dxHandle
                 << ", slot = " << ts << endl;

            MS_CDT cdt;
           cdt.chan_num  = ts;       // by Timeslot
          cdt.chan_sel  = MSPN_TS;  // ...
            cdt.chan_attr = MSPA_MODEFULLDUPLX|MSPA_NOAGC;


            rc = dcb_estconf(dspHandle, &cdt, 1, MSCA_ND,
            &cnfHandle); cout << rc << " = dcb_estconf("
                 << dspHandle
                 << ", &cdt,1,MSCA_ND, cnfHandle := " << hex << cnfHandle << ")" <<
            endl; if (DCB_ERROR == rc){
                process_dcb_error(dspHandle);
            } else {
                unsigned char notify_prm = DCB_CNF_NOTIFY;
                int notify_value = DCB_CNF_NOTIFY_OFF;
                rc = dcb_setcnfparm(dspHandle, cnfHandle, notify_prm,
                &notify_value); cout << rc << " = dcb_setcnfparm("
                     << dspHandle << ", " << hex << cnfHandle << ", "
                     << (int)notify_prm << ", " << notify_value << ")" <<
                endl; if (DCB_ERROR == rc){
                    process_dcb_error(dspHandle);
                }

                rc = dcb_getcnfparm(dspHandle, cnfHandle, notify_prm,
                &notify_value); cout << rc << " = dcb_getcnfparm("
                     << dspHandle << ", "
                     << hex << cnfHandle
                     << ", " << (int)notify_prm
                     << ", returns " << notify_value << ")" << endl;

                if (DCB_ERROR == rc){
                    process_dcb_error(dspHandle);
                }
            }
        }
    }
}

// cleanup

if (DCB_ERROR != cnfHandle) {
    rc = dcb_delconf(dspHandle, cnfHandle);
    cout << rc << " = dcb_delconf("
         << dspHandle << ", " << hex << cnfHandle << ")" << endl;

    if (DCB_ERROR == rc){
        process_dcb_error(dspHandle);
    }
}


if (DCB_ERROR != dxHandle ) {
```

```
    dx_close(dxHandle);
    cout << "dx_close(" << dxHandle << ")" << endl;
}

if (DCB_ERROR != dspHandle ) {
    dcb_close(dspHandle);
    cout << "dcb_close(" << dspHandle << ")" << endl;
}

return (0);
}
```

See Also

- **dcb_setcnfparm( )**

Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Dialogic® DCB Conferencing APIs, refer to the following document:

- *Dialogic® Audio Conferencing API Library Reference*

For more information about Dialogic® CNF Conferencing APIs, refer to the following document:

- *Dialogic® Conferencing API Library Reference*

For more information about modifying the CONFIG file, refer to the following document:

- *Dialogic® Host Media Processing Configuration Guide*

1.136 dx_reciottdata( ) Enhancements and Support for SCR

With Service Update 97, the **dx_reciottdata( )** function has the following enhancements: initial silence compression and voice activity detector (VAD) with event notification. Also with the Service Update, you have support for silence compressed record (SCR), which enables recording with silent pauses eliminated to reduce the size of recorded files without loss of intelligibility.

1.136.1 Feature Description

1.136.1.1 dx_reciottdata( ) Enhancements

The **dx_reciottdata( )** function, used to record voice data, has two new modes:

RM_VADNOTIFY
generates an event on detection of VAD during the recording operation. The new event is TDX_VAD.

Note: TDX_VAD is not an indication of function termination; it is an unsolicited event.

RM_ISCR
adds initial silence compression to the VAD detection capability.

Note: The RM_ISCR mode can only be used in conjunction with RM_VADNOTIFY.

To enable these modes, OR them to **dx_reciottdata( )** function mode parameter. For example:

```
t_Return=dx_reciottdata(DevHandle, Iott, Tpt, &t_Xpb, EV_ASYNC|RM_VADNOTIFY);

t_Return=dx_reciottdata(DevHandle, Iott, Tpt, &t_Xpb, EV_ASYNC|RM_VADNOTIFY|RM_ISCR);
```

Notes:1. The default setting for VAD notify is set to disabled. If it is not specified in the Record Mode argument in the function call, it is disabled.

2. The default setting for ISCR is set to disabled. If it is not specified in the Record Mode argument in the function call, it is disabled.

When these two modes are used together, no data is recorded as output until voice activity is detected on the line. The TDX_VAD event indicates the initiation of voice. The output file will be empty before the VAD is detected, although some initial silence may be included as specified in the CONFIG file. Initial silence is the amount of silence on the line before VAD is detected. When using RM_ISCR, the default value for the amount of allowable silence is 3 seconds. Any initial silence longer than that will be truncated. This default value can be changed by modifying a parameter in the CONFIG file for the board. Refer to the Configuring the Software section.

Supported Coders

These enhancements to the **dx_reciottdata( )** function are supported for the following encoding methods and sampling rates:

- OKI ADPCM, 6 kHz with 4-bit samples (24 kbps) and 8 kHz with 4-bit samples (32 kbps), VOX and WAVE file formats
- Linear PCM, 8 kHz sampling 64 Kbps (8 bits), 8 kHz sampling 128 Kbps (16 bits)
- G.711 PCM, 6 kHz with 8-bit samples (48 kbps) and 8 kHz with 8-bit samples (64 kbps) using A-law or mu-law coding, VOX and WAVE file formats
- G.726 bit-exact voice coder at 8 kHz with 2-, 3-, 4-, or 5-bit samples (16, 24, 32, 40 kbps), VOX and WAVE file formats

Configuring the Software

To change the default value for the amount of allowable silence when using RM_ISCR, you must add a new parameter in the CONFIG file that was selected for your board. The parameter is **0x416**, and must be added in the [encoder] section of the CONFIG file. The initial silence value for the parameter is specified directly in seconds, for example:

```
[encoder]
SetParm=0x416,6
```

This sets the maximum amount of allowable silence to 6 seconds. Any initial silence longer than that will be truncated.

1.136.1.2 Support for Silence Compressed Record (SCR)

The silence compressed record (SCR) feature is discussed in more detail in the following topics:

- Overview of Silence Compressed Record
- Enabling Silence Compressed Record
- Encoding Methods Supported in Silence Compressed Record

Overview of Silence Compressed Record

The silence compressed record feature (SCR) enables recording with silent pauses eliminated. This results in smaller recorded files with no loss of intelligibility.

The SCR algorithm operates on one msec blocks of speech and uses a two-fold approach to determine whether a sample is speech or silence. Two probability of speech values are calculated using a zero crossing algorithm and an energy detection algorithm. These values are put together to calculate a combined probability of speech.

The energy detection algorithm allows you to modify the background noise threshold range. Signals above the high threshold are declared speech, and signals below the low threshold are declared silence.

Speech or silence is declared based on the previous sample, the current combined probability of speech in relation to the speech probability threshold and silence probability threshold parameters and the trailing silence parameter.

Enabling Silence Compressed Record

Use **dx_setparm( )** and the new parameter, **DXCH_SCRFEATURE**, to turn SCR on and off. Once enabled, voice record functions automatically record with SCR.

For **DXCH_SCRFEATURE**, the valid values are:

Define	Bytes	Read/Write	Default	Description
DXCH_SCRFEATURE	2	R/W	-	Silence Compressed Record (SCR). Valid values are: • **DXCH_SCRDISABLED** – SCR feature disabled • **DXCH_SCRENABLED** – SCR feature enabled

Note: This will override any board-level settings.

SCR Encoding parameters are supported via the CONFIG file. To change the parameter values for the SCR, you must add new parameters in the CONFIG file. The parameters listed below must be added in the [encoder] section of the CONFIG file. For details about these parameters, refer to the *Dialogic® Host Media Processing Configuration Guide*.

- AEnc_VADSpchHng (SCR Trailing Silence)
- AEnc_VADSpchPrO (SCR Speech Probability Threshold)
- AEnc_VADSpchPrC (SCR Silence Probability Threshold)
- AEnc_VADLoThr (SCR Low Background Noise Threshold)
- AEnc_VADHiThr (SCR High Background Noise Threshold)

Encoding Methods Supported in Silence Compressed Record

The following encoding algorithms and sampling rates are supported in SCR:

- OKI ADPCM, 6 kHz with 4-bit samples (24 kbps) and 8 kHz with 4-bit samples (32 kbps), VOX and WAVE file formats
- linear PCM, 8 kHz sampling 64 Kbps (8 bits), 8 kHz sampling 128 Kbps (16 bits)
- G.711 PCM, 6 kHz with 8-bit samples (48 kbps) and 8 kHz with 8-bit samples (64 kbps) using A-law or mu-law coding, VOX and WAVE file formats
- G.726 bit-exact voice coder at 8 kHz with 2-, 3-, 4-, or 5-bit samples (16, 24, 32, 40 kbps), VOX and WAVE file formats

1.136.2 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Voice API, refer to the following documents:

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

For more information about configuring Dialogic® HMP Software, refer to the following document:

- *Dialogic® Host Media Processing Configuration Guide*

1.137 Support for Advanced Network Services

This Service Update 90 verifies that Dialogic® HMP software can interoperate with Advanced Network Services (ANS) teaming software. Teaming allows a customer to group multiple physical network adapters (i.e., NIC interfaces) into *virtual* adapters. Establishing this kind of association between adapters makes it possible to support fault tolerance, load balancing, and virtual LAN (VLAN) tagging. Certifying that Dialogic® HMP software can operate in conjunction with ANS provides customers valuable tools to improve network reliability.

This section provides an overview of ANS Adapter Teaming. For a more complete description, refer to *http://www.dialogic.com/support/network/sb/CS-009747.htm*.

1.137.1 ANS Drivers

To determine if a server's network adapter has the ANS software and drivers installed, select a LAN and view the corresponding LAN Properties window. Then select an adapter and review its NIC Properties window, Teaming tab. If Teaming is enabled, it will be apparent.

ANS requires that servers have the appropriate ANS drivers installed. The latest driver version for each adapter type is available at *http://www.dialogic.com/support*.

1.137.2 Team Roles

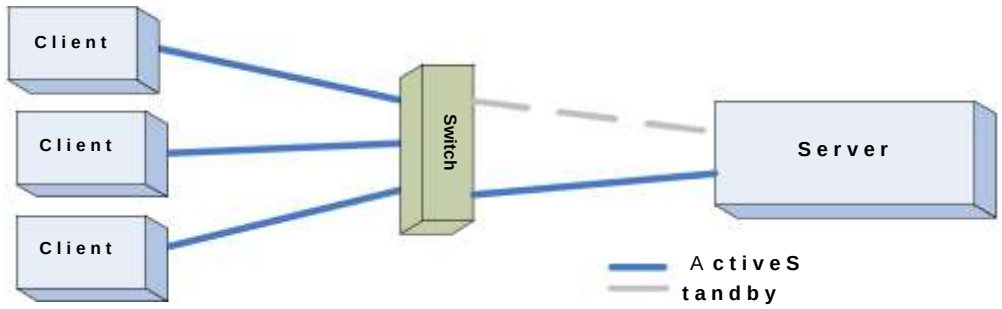
ANS supports teams consisting of from one to eight NIC interfaces, but this feature is limited to two-member teams. In a two-member team, one interface serves as the *primary* network interface and the other serves in a *secondary*, standby role. Both failover protection and load balancing are supported by assigning primary and secondary team roles.

1.137.3 Fault Tolerance

Failure protection is afforded through two types of fault tolerance, *Adapter Fault Tolerance* (AFT) and *Switch Fault Tolerance* (SFT). AFT is the default mode when a team is created. When AFT is operating, the secondary adapter automatically carries traffic when there is a connection failure of any type (e.g., cable, adapter, port, link partner) on the primary.

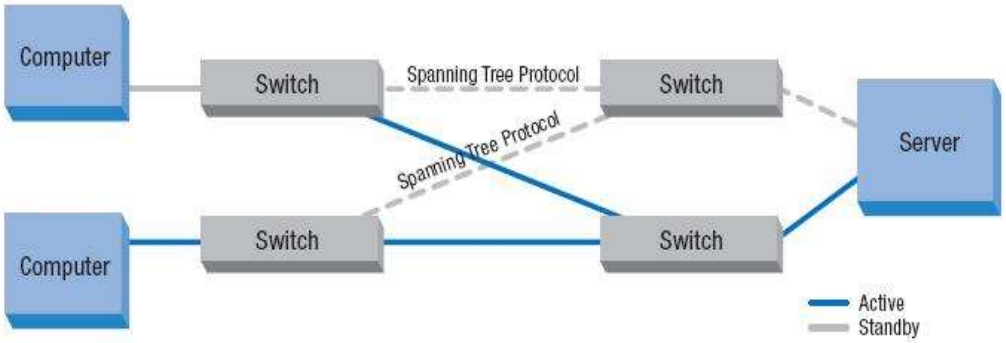
ANS passes the failed primary adapter MAC (Media Access Control) address and Layer 3 (IP) address to the secondary adapter to make this possible. In AFT, all adapters are connected to the same switch or hub, as in the following diagram.

Adapter Fault Tolerance



Switch Fault Tolerance, in contrast, supports a failover relationship between two adapters when each adapter is connected to a separate switch. In SFT, if one adapter, its cabling or the switch fails, the other adapter takes over, as in the following diagram.

Switch Fault Tolerance



ANS provides fault tolerance for different network architectures, with both single and multiple switch connections supported.

1.137.4 Load Balancing

Load balancing distributes the transmission and reception traffic among network adapters. This helps to maximize performance and improve reliability. Two types of load balancing are supported, Adaptive Load Balancing (ALB) and Receive Load Balancing (RLB). ALB supports transmission over from two to eight ports. Transmission to multiple destination addresses is supported and ALB incorporates Adapter Fault Tolerance. Receive Load Balancing is a sub feature of ALB and can only work in conjunction with it. RLB allows reception of network traffic on from two to eight ports from multiple client addresses. However, RLB only works with TCP/IP-based traffic.

Note: TCP/IP must be specified as the transport protocol in Global Call before RLB configuration can be run.

For instructions on how to set TCP/IP as the transport protocol, go to *Configuring TCP Transport* in the *Global Call for IP Technology Guide.*

1.138 Set the MIME Message in SIP 200 OK Response

With Service Update 90, you have the opportunity to specify and construct a MIME message body for inclusion in the SIP 200 OK response to an incoming remote user BYE request (200 OK-to-BYE). This feature is now the default behavior for the 200 OK-to-BYE message response.

1.138.1 Feature Description

Currently, Global Call operating in 1PCC mode supports the automatic 200 OK-to-BYE message response being sent when a BYE is received. However, in order to attach MIME information to a 200 OK-to-BYE response message, the MIME information must be pre-loaded **before** receiving the BYE and GCEV_DISCONNECT event. If it is not preloaded, Global Call will send the 200 OK-to-BYE response message without any MIME body.

With the new feature, the 200 OK is **not** sent back automatically when a BYE is received; the 200 OK is sent back when you issue a **gc_DropCall( )** function. This modification enables you to use the preloaded MIME feature as it is documented in the *Dialogic® Global Call IP Technology Guide* and set the MIME (using **gc_SetUserInfo( )**) for a 200 OK (for a BYE) **after** the incoming MIME attached along with a GCEV_DISCONNECTED event is received.

1.138.2 Documentation

The online bookshelf provided with Dialogic® PowerMedia™ HMP for Windows Release 3.0 contains information about all release features including features for application development, configuration, administration, and diagnostics.

For more information about Dialogic® Global Call APIs, refer to the following documents:

- *Dialogic® Global Call IP Technology Guide*

Release Issues 2

This chapter includes the following topics that relate to release issues:

2.1 Issues

The following table lists issues that can affect the software supported in Dialogic® PowerMedia™ HMP for Windows Release 3.0. 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:
  - Resolved – An issue that was resolved (usually either fixed or documented) in this release.
  - Known (permanent) – A known issue or limitation that will not be fixed in the future.
  - Known – A minor issue that affects the Dialogic® PowerMedia™ HMP for Windows Release 3.0. This category includes interoperability issues and compatibility issues. Known issues are still open but may or may not be fixed in the future.

Defect No.
: A unique identification number that is used to track each issue reported via a formal Change Control System.

PTR No.
: Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the defect number is used to track the issue.

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, typically one of the following:
  - a system-level component; for example, Host Media Processing Software
  - a software product; for example, the Global Call API Library
  - a software component; for example, Continuous Speech Processing

Description
: A summary description of the issue. For non-resolved issues, a workaround is included when available.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	HMP-1109	--	525	IP Host	Gc_OpenEx() fails when called on systems with only IPv4 addresses configured.
Resolved	HMP-1105	--	525	SSP	Audio quality issues when running HMP on VMWare systems.
Resolved	HMP-1012	--	525	IP Host	Sip authentication failure when processing BYE message.
Resolved	HMP-794	--	525	IP Host	HMP generates incorrectly formatted ISUP:ACM CANCEL messages.
Resolved	HMP-1075	--	520	Firmware	Service Update 395 crashes under load in *ssp.mlm.sys*.
Resolved	HMP-1013	--	520	Firmware	Application exception is reported when using MM streaming I/O interface under load for AMR-WB calls.
Resolved	HMP-842	--	520	Installation	The Visual C++ 2015 Redistributable x86 package fails to install when Visual C++ 2015 Redistributable x64 package is already present on the system.
Resolved	HMP-1016	--	520	IP Host	GCEV_TASKFAIL is returned after calling **gc_AcceptModifyCall( )** to respond to prior SIP re-INVITE received for active SIP call.
Resolved	HMP-964	--	520	IP Host	HMP crashes in the *libsipsigal* component when receiving INVITE from Cisco Unified Communications Manager.
Resolved	HMP-908	--	520	IP Host	No inbound or outbound calls are being processed as the SIP stack marks local address as unusable when TCP connection is reset by remote end point.
Resolved	HMP-801	--	520	IP Media	The **ipm_ModifyMedia( )** response is slow in multithreaded applications when called with EV_SYNC.
Resolved	HMP-997	--	520	OA&M	There is a vulnerability due to the FlexNet version.
Resolved	HMP-878	--	395	Call Control	The **gc_Start( )** function fails due to missing *pdkrt.dll*.
Resolved	HMP-850	--	395	Host Library	Service Update 393 contains some Global Call libraries which still have dependency of obsoleted Visual Studio 2005 Runtime package.
Resolved	HMP-859	--	395	IP Host	The l*ibsipsigal.dll* crashes multiple times per day.
Resolved	HMP-834	--	395	IP Host	When IP_PROXY_BYPASS is enabled, the SIP REGISTER message gets transmitted using wrong transport protocol.
Resolved	HMP-776	--	395	IP Media	T.38 Hairpinning fails due to no T.38 data passed between the two endpoint call legs.
Resolved	HMP-760	--	395	IP Media	The **ipm_ModifyMedia( )** function fails when called with native coders.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	HMP-759	--	395	IP Media	The IPMEV_NOTIFY_ENDPOINTID events are reported empty as 0.0.0.0:0 for IP address and port number.
Resolved	HMP-829	--	393	Host Library	Service Update 387 executable modules (.exe) have dependency of obsoleted Visual Studio 2005 Runtime package.
Resolved	HMP-800	--	393	Host Library	When using 64-bit library, a call to **dcb_open( )** function crashes.
Resolved	HMP-793	--	393	Host Library	DCM is unable to identify licensed NIC.
Resolved	HMP-825	--	393	Installation	The Visual C++ 2015 Redistributable x64 package fails to be installed as part of the Service Update 387 installation process.
Resolved	HMP-835	--	393	OA&M	When logged onto system as a non-admin user with admin rights, an attempt to start DCM hangs on splash screen.
Resolved	HMP-788	--	387	Installation	Service Update 382 fails to install the Visual C++ 2015 Redistributable packages when Visual Studio 2017 is installed on the system.
Resolved	HMP-787	--	387	Installation	During the Service Update 382 installation process, the *UpdatedlgDrvr.exe* generates an exception.
Resolved	HMP-758	--	385	Firmware	The **ipm_Stop( )** and **ipm_ModifyCall( )** functions are taking 100-500 ms to complete.
Resolved	HMP-741	--	385	Host Library	Unable to run two applications at once with “Error -4 communicating with License Server”.
Resolved	HMP-694	--	382	Firmware	Sporadic crash in ssp_mlm during CNF audio conferencing.
Resolved	HMP-673	--	382	Firmware	The **mm_StreamWrite( )** function gets hung/stuck during play when using MM streaming I/O interface with AMR-WB underload.
Resolved	HMP-655	--	382	Firmware	KMBC bridging component does not initialize successfully upon services startup on certain servers, which leads to incorrect timeslot information when retrieved from DNI2 board.
Resolved	HMP-672	--	382	Host Library	The **dx_resetch( )** function and its related definitions are missing from DXXXLIB.h header file.
Resolved	HMP-603	--	382	IP Host	SIP stack incorrectly sends crypto-based SDP content outgoing SIP messages in response to SIP INVITE.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	HMP-535	--	382	IP Host	The **gc_ReleaseCallEx( )** function fails in disconnect while in blind transfer scenario.
Resolved	HMP-262	--	382	IP Host	SIP transfer fails when RTF config is set to default levels.
Resolved	HMP-657	--	382	OA&M	Restart attempt of Dialogic services in DCM fails on Windows Server 2016.
Resolved	HMP-626	--	382	OA&M	HMP is not starting through DCM on Windows Server 2016.
Resolved	HMP-693	--	382	SSP	RFC 2833 digits with payload type of 102 are not properly propagated in native joins.
Known (permanent)	HMP-705	--	382	Firmware	The CR2 alarm indicator stays lit on DNI310TEPE2HMP or DNI610TEPE2HMP board with no alarm condition.
Resolved	IPY00118608	--	375	Firmware	HMP incorrectly sends NSS message in response to NSF for outbound fax call.
Resolved	HMP-632	--	375	Host Library	The 64-bit **dx_GetStreamInfo( )** function writes past the end of the DX_STREAMSTAT data structure.
Resolved	HMP-595	--	375	Host Library	A fault in runtime is encountered each time when receiving fax errors.
Resolved	IPY00118609	--	375	Host Library	When executing DEBG print during send fax operation, an application exception occurs in the DM3 Fax library.
Resolved	IPY00118578	--	375	IP Host	Outbound SIP calls fail to get to a connected state.
Resolved	IPY00118455	--	375	IP Host	When HMP sends SIP REGISTER with 2 From: and 2 Contact: headers, both the 2nd headers are empty.
Resolved	IPY00118443	--	375	IP Host	While in blind transfer scenario, the **gc_ReleaseCall ( )** function fails in disconnect.
Resolved	IPY00118315	--	375	IP Host	486 Busy is seen after using system for several days.
Resolved	IPY00118224	--	375	IP Host	Incorrect RFC 2833 payload type inserted into SDP when using G.729.
Resolved	HMP-619	--	375	OA&M	The About dialog box in DCM displays incorrect operating system information on Windows 2016 systems.
Resolved	IPY00118501	--	375	Tools	DCM cannot be started on HMP virtual machine.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00118527	--	372	OA&M	Lack of VC9 MSFT CRT redistributable causes a failure to launch *lmgrd.exe* during HMP License Manager Service installation, aborting the setup during HMP installation. When configured for delayed start, the HMP License Manager Service causes the Dialogic System Service hang on system restart.
Resolved	IPY00118416	--	371	Diagnostics	The *its_sysinfo.exe* tool fails to collect Default IP Address information during Installed Board Configuration step.
Resolved	IPY00118392	--	371	Diagnostics	The *its_sysinfo.exe* tool hangs at Processor Information step.
Resolved	IPY00118363	--	371	Firmware	When opening network (DTI) devices on DNI2 boards during application startup, there is a ConnectSCbus Failed error that occurs.
Resolved	IPY00118039	--	371	Firmware	The application called **ec_stream( )** but did not receive an completion event in return.
Resolved	IPY00118419	--	371	IP Host	The server returns 501 Not Implemented error in response to OPTIONS requests.
Resolved	IPY00118343	--	371	IP Host	The OpenSSL package delivered in HMP is susceptible to potential security vulnerabilities.
Resolved	IPY00118333	--	371	IP Host	The SIP stack incorrectly sends proprietary SDP content in outgoing SIP messages.
Resolved	IPY00118274	--	371	IP Host	A **gc_InvokeXfer( )** initiated, TCP-transport, SIP REFER blind transfer, fails to send a 200 OK after first received NOTIFY, even though the HMP app receives successful transfer complete event.
Resolved	IPY00118159	--	371	IP Host	When receiving an incoming call with precondition attributes in SIP INVITE, HMP responds with those SDP attributes in 200 OK without adding preconditions in supported header.
Resolved	IPY00117959	--	371	IP Host	SIP re-INVITE uses UDP after initial INVITE that used TCP.
Resolved	IPY00118367	--	371	OA&M	MMC detects error in *DM3Config.exe* entries added to the system event log.
Resolved	IPY00117965	--	371	OA&M	The *lmgrd.exe* licensing module delivered in HMP is susceptible to a buffer overflow vulnerability.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00118341	--	371	SSP	A bugcheck crash occurs in the SSP component when performing SIP calls with media activity under load on Windows 2008 (32-bit) systems.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known (permanent)	IPY00118380	--	371	IP Host	A TLS application initiating a transfer with **gc_InvokeXfer( )** must be aware of the following facts: 1. The call causes HMP to become a TLS Client and initiate a TLS handshake towards the Transferring or Transferor agent, which becomes the TLS Server. 2. HMP will obtain the Server's hostname from the Contact header of a previous incoming message (INVITE or ACK). 3. HMP will attempt to match the hostname with: • Server X.509 Certificate's DNS entry in Subject Alternative Name (SAN) extension, first. • Server X.509 Certificate's CN (commonName) attribute of the Distinguished Name (DN), second. • The IP Address entry in the Certificate's SAN is not used by HMP. 4. If neither 2.1 or 2.2 matches the hostname, TLS handshake will fail. These facts are particularly important when the remote Transferor/Transferring Agent uses an IP address, rather than a FQDN in its Contact header, while its Server Certificate's DNS entry in the SAN or CN attribute in the DN has no match. In that case, the **gc_InvokeXfer( )** will fail even if its called party information (in direction field of GCLIB_MAKECALL_BLK or numberstr function argument(s)) matches the remote Server's IP address. ***Workaround:*** • Make sure the Remote Agent's Certificate has a match, as described. • Use the existing HMP SIP TLS Post-Connect Callback Function to accept the post-Connect assertion, as required. Refer to Section 1.19, "SIP TLS Certificate Verification and Post-Connect Callback Functions", on page 61 for more information.
Resolved	IPY00117895	--	367	Dialogic® DNI Board	If the HMP system is shut down while Dialogic services are running, an NMI error (Uncorrectable PCI Express Error) occurs.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00118065	--	367	Firmware	A bugcheck crash occurs in the SSP component of the firmware/driver when using the MM File I/O with AMR-WB and the system is under load.
Resolved	IPY00117899	--	367	Firmware	The MMEV_PLAY event is returned after 1 second when using the MM streaming I/O interface, playing a 15 second prompt, and under load.
Resolved	IPY00117850	--	367	Firmware	The MMEV_RECORD_FAIL event is received after an active recording terminates (MAXTIME) when using the MM streaming I/O interface with AMR-WB and the system is under load.
Resolved	IPY00117732	--	367	Firmware	Outbound faxes fail with TFX_FAXERROR when HMP sends V.17 faxes via the DNI port.
Resolved	IPY00117627	--	367	Firmware	Sometimes when **ipm_GetSessionInfo( )** is called, the libipm_ipvsc module randomly crashes.
Resolved	IPY00117565	--	367	Firmware	When using MM Streaming I/O with AMR-WB under load, sometimes an mm_Record does not start and returns the MMEV_RECORD event immediately after execution.
Resolved	IPY00117560	--	367	Firmware	**ipm_GetSessionInfo( )** returns invalid QoS values.
Resolved	IPY00117541	--	367	Firmware	A bugcheck crash occurs in the SSP component of the firmware/driver when using the MM Streaming I/O with AMR-NB and the system is under load.
Resolved	IPY00117540	--	367	Firmware	When using MM Streaming I/O, there are audio glitches in G.722 and AMR-xx recordings.
Resolved	IPY00117539	--	367	Firmware	When using MM Streaming I/O to play back a recorded AMR-NB file that is 16 seconds, the MMEV_PLAY event is missing.
Resolved	IPY00117388	--	367	Firmware	A call to **gc_SetConfigData( )** to enable or disable the onboard echo canceller hangs indefinitely under certain circumstances.
Resolved	IPY00117163	--	367	Firmware	The tones generated using **dx_playtone( )** are too short and the completion event is delayed.
Resolved	IPY00117598	--	367	Global Call	When receiving a SIP call looking to use the G.726 codec, a GEV_TASKFAIL event is returned.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00117394	--	367	Host Library	CNG tones are not detected by HMP's CPA algorithms when using the voice method.
Resolved	IPY00117326	--	367	Installation	There is an "Unquoted Service Path Enumeration" vulnerability for Dialogic services registry entries.
Resolved	IPY00117927	--	367	Installation	When upgrading HMP 3.0 to a newer service update, some old files are not replaced by new ones. It is recommended to uninstall the existing HMP 3.0 software before installing the new HMP 3.0 service update.
Resolved	IPY00118005	--	367	IP Host	CPU usage is high after the remote peer stops responding to SIP messages from HMP.
Resolved	IPY00117658	--	367	IP Host	When the HMP sends a BYE after the application calls **gc_InvokeXfer( )** for a blind transfer, the "From" SIP header reverts to the default.
Resolved	IPY00117370	--	367	IP Host	If an outbound proxy is specified by its name, the DNS does not resolve the name and outbound calls return DISCONNECT.
Resolved	IPY00117192	--	367	OA&M	Dialogic® Services start but the underlying components and services fail.
Resolved	IPY00117252	--	367	PSTN Firmware	**gc_MakeCall** times out for all calls after a particular time on a single E1 use.
Resolved	IPY00117194	--	367	PSTN Firmware	Outgoing calls fail when the number digits do not match the calling, called, and redirection party digits.
Resolved	IPY00116675	--	367	SNMP	The OIDs in the MIBs do not correspond with all of the OIDs that the HMP sends in SNMP traps and the OIDs that can be queried with SNMP GET.
Known	HMP-300	--	367	OA&M	Sporadically, when a user repeatedly stops and starts the HMP services (without an operating system reboot), the HMP services can get into a condition where the services fail to start. Should this occur, a system reboot is required to recover this condition.
Resolved	IPY00117496	--	361	Host Library	TLS calls cannot be connected due to buffer size limit of 2048 bytes for server certificate.
Resolved	IPY00117494	--	361	Host Library	Maxcall-legs limit is encountered when processing multiple REFERs within single SIP call when using the feature: Support for SIP REFER for Adding a Participant in a Conference (RFC 4579, § 5.6).

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00117370	--	361	Host Library	HMP incorrectly resolves SIP proxy host name as SRV record rather than A record and outbound calls return DISCONNECT with IPEC_SIPReasonStatus503Service Unavailable return code; no SIP message is sent out of HMP.
Resolved	IPY00117258	--	361	Host Library	The SIP stack module is crashing and calls are not being processed.
Resolved	IPY00117513	--	361	Voice Library	**dx_addtone( )** call may fail with "Device busy" when creating multiple tone templates on high density systems.
Resolved	IPY00117456	--	360	Firmware	Missing events or API errors are seen at the application layer after 31 or more days of continuous operation. From a high level, an application may stop receiving asynchronous completion events during normal API operations, or when attempting to terminate active dx_ API operations when calling **dx_stopch( )** API. The issue may also manifest itself with errors being returned when executing API functions. Additionally, a dm3nk driver error would be observed in the Event Viewer.
Resolved	IPY00117066	--	360	Firmware	Robotic audio sound is heard in outgoing audio when two SIP calls are patched.
Resolved	IPY00116943 (IPY00116663)	--	360	Firmware	The audio RTP port base value cannot be changed.
Resolved	IPY00116918	--	360	Firmware	The RTP timeout event is not generated in cases where no RTP packets are received.
Resolved	IPY00117135	--	360	Global Call IP	Access violation occurs after calling **gc_util_copy_parm_blk( )** API to copy a gc_ParmBlk.
Resolved	IPY00117422 (IPY00117116)	--	360	Host Library	Application exception occurs when attempting ERR1 print in library stream source component.
Resolved	IPY00117299		360	Host Library	Unable to retrieve ptime from SDP of incoming SIP re-INVITE message.
Resolved	IPY00117289		360	Host Library	SIP call hold/retrieve is not working with specific IP PBX when session attribute is used.
Resolved	IPY00117287		360	Host Library	Incorrect frame size/ptime is used for TX side in RTP stream during outbound SIP calls.
Resolved	IPY00117243	--	360	Host Library	When using DNI + SPCI combination for signaling, Dlgcs7 crashes and stops logging.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00117452 (IPY00117433)	--	360	SIP Call Control	HMP is rejecting calls during high load with “SipSignaling::dropExcessCall” and “486 Busy Here” error messages.
Resolved	IPY00116942	--	360	SIP Call Control	SIP header in REFER is using stale NONCE value. Refer to Section 1.20, “Support for New SIP_STACK_CFG Data Structure User-Configurable Fields”, on page 67 for more information.
Resolved	IPY00116781	--	360	SIP Call Control	An error message “No more elements are available” is seen in the RTF logs and the application receives a SIP 503 message. This causes calls to be dropped from this point onwards until the HMP server is reset.
Resolved	IPY00116671	--	360	SIP Call Control	An attempt to set Max-Forwards header field in outgoing SIP REGISTER message results in two header fields sent within message.
Resolved	IPY00116605	--	360	SIP Call Control	Calling **gc_SetUserInfo( )** returns error when attempting to set Call-ID string of SIP header for outgoing SIP INFO message.
Known	IPY00117074	--	360	SIP Call Control	Call resource leak can be experienced in a peculiar case where the UA changed the To: tag header between “183 Session Progress” and “200 OK” responses; in this case a call resource is forked, but the original one might not be released if no further responses are received with the original To: tag header. ***Workaround:*** Set the forked1xxTimerTimeout field to zero (“0”) to avoid call-leg forking altogether. Refer to Section 1.20, “Support for New SIP_STACK_CFG Data Structure User-Configurable Fields”, on page 67 for more information.
Known	HMP-190	--	360	Firmware	Firmware may halt with QERROR_KILLTASK error message if dx devices are closed while tone templates are being added.
Resolved	IPY00116484	--	357	Board Download	If the HMP device fails to start, DCM still shows the HMP device as started with a green arrow.
Resolved	IPY00116560	--	357	Conferencing	Conference resources being exhausted overtime because of a race condition causing internal routing to fail.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00116351	--	357	PSTN Firmware	Sporadic cases of TS16 AIS alarm at initialization experienced on DNIxxxTEPE2HMP with E1 CAS protocol.
Resolved	IPY00116346	--	357	PSTN Firmware	DNIxxxTEPE2HMP configured for E1 CAS with no CRC triggers LOMF alarm detection and sending back TS16 RAI alarm.
Resolved	IPY00116752	--	357	SIP Call Control	Attempt to drop a call in progress prior to connection when using TCP via Proxy fails to send CANCEL message.
Resolved	IPY00116696	--	357	SIP Call Control	**gc_ReqModifyCall( )** is rejected on the second call consistently.
Resolved	IPY00116619	--	357	SIP Call Control	Failure to handle re-INVITEs in rapid succession leads to GCEV_REJECT_MODIFY_CALL errors in RTF logs.
Resolved	IPY00116613	--	357	SIP Call Control	On frequent multiple re-INVITEs, the HMP SIP stack returns incorrect “491 Request Pending” when they come in rapid succession and one is in process at the time of re-INVITE.
Resolved	IPY00116402	--	357	SIP Call Control	Application crash in SIP library caused by SIP digest authentication triggered by a race condition when changing the authentication parameters with the **gc_SetAuthenticationInfo( )** even when there is no update.
Known	IPY00116723	--	357	Voice	**dx_reciottdata( )** with TPT specifying DX_MAXSIL / TF_SETINIT stops immediately if initSilResume is “enabled” and dx_setevtmsk(SILON|SILOFF) has been set.
Resolved	IPY00116335	--	354	SIP Call Control	A crash was experienced in **gc_GetMetaEventEx( )** while retrieving TDM telephony data embedded in MIME buffers causing immediate dropped call.
Resolved	IPY00116305	--	354	SIP Call Control	**gc_CancelWaitCall( )** causes SIP resources to be lost, reducing the available SIP IP channels over time.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00116249	--	354	Dialogic® HMP	HMP crash (KILLTASK) might be observed during service start, causing HMP devices to be unavailable, when restarting the DSS multiple times. This is the result of system memory fragmentation which prevents proper allocation of HMP resources requiring contiguous blocks at HMP initialization time. To restore HMP to a functional state, the entire system must be restarted (reboot).
Known	IPY00116523	--	354	T.30 Fax	A higher than expected error rate was experienced while regression testing fax cases where T.30 is used with PSTN transport on only (DNIxxxxTEPEHMP).
Known	IPY00116501	--	354	Voice/Fax	D4/PCI-U voice resources do not correctly detect the fax tone in call progress.
Known (permanent)	IPY00116250	--	354	Dialogic® HMP	In a Windows 2003 32-bit with 2.5 GB system memory, the system runs out of resources and becomes unstable after HMP service started successfully with more than 400 IP RTP port license.
Resolved	IPY00115897	--	349	Dialogic® DNI Board	In a host-to-board streaming broadcasting case, when one host-to-board stream is disconnected, the other stream is also disconnected causing undesired noise in a conferencing scenario.
Resolved	IPY00102738	--	349	Dialogic® DNI Board	**dt_listen( )** on DNIxxxxTEPE2HMP fails to route network channels resulting in no audio heard whenever the network channel device is closed and re-opened in application.
Resolved	IPY00115565	--	349	Fax	Unable to fax a very large page document, fax send stops abruptly.
Resolved	IPY00115269	--	349	Fax	DIS/CSI are not detected on HMP, as those are mistaken as commands after an underrun condition on V.17 modem.
Resolved	IPY00115808	--	349	Global Call IP	HMP fails to send REFER message after **gc_InvokeXfer( )** was called due to parse errors seen with Refer-To header.
Resolved	IPY00115666	--	349	Global Call IP	Access violation seen when **gc_Stop( )** is called on system configured with 5000 licensed IP call control channels.
Resolved	IPY00057363	--	349	Global Call IP	Application exception happens in **gc_GetMetaEvent( )** during application startup.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00057362	--	349	Global Call IP (SIP)	Call transfer failed with 481 - failed to fetch Call leg for transaction.

Resolved	IPY00100042	--	349	T.38 Fax	HMP Fault Detector (FD) reports numerous T.38 fax firmware warnings due to an underrun condition; RTF errors from the FD might look like these which repeat after a few minutes if the underrun condition persists:
					`OAMSYSLOG ErrorEx`
					`DM3FDSP - <QError> Board:0 Processor:1`
					`UsrSeverity:2 ErrorTag:438 CauseTag:0`

Resolved	IPY00115304	--	349	Voice	*LIBDXXDM3.dll* crash caused by exhaustion of tone-tracking list and improper handling of board errors in this case.
Known	IPY00115905	--	349	Installation	The install option "Upgrade" does not properly update driver files on post-Vista Windows systems. In particular, it would expose strange/unexpected phenomenon. Especially when customer upgrades their existing system to SU 349, due to the changes made in core driver SW related adding of multiple RTP IP address support (multihomed), a customer performing an "Upgrade" might unexpectedly see a problem where RTP port's local IP address returned is 127.0.0.1 despite of setting in "IP Address" tab in DCM. ***Workaround:*** On any post-Vista Windows systems (e.g., Windows 2008), always perform a full removal of the existing HMP SU as documented, and perform a fresh install of the newer SU, as documented. Refer to the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide* for more information on installation procedures.
Known (permanent)	N/A	--	349	Fax	As of SU 349, 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.
Resolved	IPY00102864	--	347	Device Driver	SU 343 won't start on Windows 2012.
Resolved	IPY00102451	--	347	Global Call IP (SIP)	The SDP version number is not incrementing when refresh re-INVITE is received.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00057320	--	347	Global Call IP (SIP)	An access violation occurs within libgch3r when using dynamic outbound proxy.
Resolved	IPY00102111	--	343	Board Download	Dialogic System Service will not start or detect properly when multiple DNI2410TEPE2HMP boards are installed.
Resolved	IPY00102048	--	343	Board Download	HMP fails to start on some platforms.
Resolved	IPY00101970	--	343	Drivers	WMI service crashes on system startup.
Resolved	IPY00101839	--	343	Drivers	DNI board suddenly stops operating.
Resolved	IPY00094373	--	343	HMP Licensing	HMP license manager shows failure to activate the license.
Resolved	IPY00102354	--	343	PSTN	Once TS16 AIS is activated, the alarm will not go away with E1 CAS on DNI2410TEPE2HMP board.
Resolved	IPY00102219	--	343	PSTN	Line sync issues with E1 CAS on DNI2410TEPE2HMP board.
Resolved	IPY00102052	--	343	PSTN	RAI alarms does not get transmitted on DNI2410TEPE2HMP board.
Resolved	IPY00102042	--	343	PSTN Call Control	Sporadic **gc_MakeCall( )** failure with E1 CAS on DNI2410TEPE2HMP board.
Resolved	IPY00099638	--	343	PSTN Call Control	Calls with overlap receiving might receive duplicate digits in DNIS.
Resolved	IPY00102110	--	343	SIP IP	SIP registration fails after 6 to 7 days in auto refresh mode.
Resolved	IPY0057293	--	343	SIP IP	The “Connect Information” is not reported properly in IPSET_RTP_ADDRESS of type IPPARM_REMOTE after GCEV_REQ_MODIFY_CALL event in application.
Resolved	IPY00102241	--	343	SS7	Global Call SS7 does not work in HMP with Windows 2008.
Known	IPY00102428	--	343	Drivers	When operating in Windows 2012, D/4PCIU might observe “KMBC.SYS: CLOCK FAILED” sporadically, as a result of a bridge device failure; the system and board continue operating in a degraded mode.
Known (permanent)	IPY00091575	--	343	IP Media	**ipm_getLocalMediaInfo( )** needs to be initiated before calling **ipm_StartMedia( )** if the media type is different from the last call to **ipm_getLocalMediaInfo( )**; otherwise a failure starting a new media session will occur.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known (permanent)	IPY00100100	--	343	SIP IP	Setting session timers through **gc_SetUserInfo( )** with IPSET_SIP_SESSION_TIMER on a channel line device, outside of a dialog, under some circumstances might consume a SIP Call Control license, e.g. doing so upon **gc_DropCall( )** on an inbound channel that has been issued a **gc_WaitCall( )**. Unless an application has the necessity of changing session timers on every channel, it's recommended they are set prior to invoking the **gc_WaitCall( )**; alternatively if session timers must be updated dynamically, that can be done using the aforementioned **gc_SetUserInfo( )** and IPSET_SIP_SESSION_TIMER within the dialog by using the call's CRN instead.
Resolved	IPY00099952	--	338	Fax	When the fax resource does not send to the host a Phase D event for the RTN page, ATFX_PGXFER shows a skip in the page count.
Resolved	IPY00101287	--	338	IP Media	T.38 Hairpinning fails due to packets not being relayed intact between call legs.
Resolved	IPY00101863	--	338	SIP Call Control	An application crash when releasing heap memory is observed on servers with heavy use of SIP OPTIONS.
Resolved	IPY00101457	--	338	SIP Call Control	When the system receives more calls than it is ready to accept, the application crashes with exception at *libsipsigal.dll*.
Resolved	IPY00101443	--	338	SIP Call Control	The SDP received in 200 OK was not reported with GCEV_CONNECTED event.
Resolved	IPY00101299	--	338	SIP Call Control	SIP INVITE is sent to the address in the Request-URI header field as opposed to the original destination address specified. Refer to Section 1.36, "Special SIP INVITE Request-URI Overwrite", on page 124 for more information.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00057256	--	338	SIP Call Control	Un-handled Out of Dialog NOTIFY requests causes a resource leak. By default this feature is enabled; to disable receipt of out of dialog NOTIFY messages, the following field has been added to the IP_VIRTBOARD: E_SIP_OOD_NOTIFY_Access By default this field is enabled (ENUM_Enabled;) The application must explicitly disable the feature, if so desired. The following should be added before calling **gc_Start( )**. virtBoards[0].E_SIP_OOD_NOTIFY_Access = ENUM_Disabled;
Resolved	IPY00057220	--	338	SIP Call Control	Large incoming SIP messages with diversion header causing a GCEV_TASKFAIL event.
Resolved	IPY00101250	--	338	SIP IP	When HMP doesn't receive a response to a TCP SIP INFO with MIME body content within 30 seconds, it retransmits a UDP SIP INFO message with no MIME body content.
Resolved	IPY00101303	--	338	Voice	A TDX_PLAY event is not returned when digit is already present in buffer prior to **dx_play( )** being issued with MAXDTMF termination condition set to 1.
Resolved	IPY00099145	--	328	Conferencing	Conference tone clamping results in distorted audio.
Resolved	IPY00093730	--	328	Drivers	The voice device is unresponsive on initiating outbound calls.
Resolved	IPY00100893	--	328	Installation	During an uninstall, the error “Failed to Launch media_server.exe, parameters: -remove, Return Code: 0” is observed.
Resolved	IPY00100775	--	328	IP Media	SIP Overlap issues: • In 1PCC, GCEV_TASKFAIL on 484 Response to an HMP enbloc INVITE when the SIP Overlap feature is globally enabled via SEND_RECEIVE_TYPE_RFC_3578. • In 3PCC, Overlap **gc_MakeCall**(GCADDR_OVERLAP) generates GCEV_DISCONNECTED instead of GCEV_REQMOREINFO on receipt of 484 Response.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00100449	--	328	IP Media	Executing the **ipm_Stop( )** function in asynchronous mode degrades when stopping 500 channels simultaneously. This causes IPMEV_STOPPED events to be returned later than expected.
Resolved	IPY00100127	--	328	IP Media	Calling the **gc_MakeCall( )** function with invalid or corrupted IP addresses causes an immediate GCEV_DISCONNECT, and when the application tries to retrieve a cause with the **gc_ResultInfo( )** function, it fails with -1.
Resolved	IPY00057174	--	328	IP Media	HMP fails to handle two 100 Trying messages. With the second 100 Trying message, the application receives a GCEV_TASKFAIL event.
Resolved	IPY00057170	--	328	IP Media	A GCEV event notification is not sent to the application after a SIP 302 message is received in response to SIP NOTIFY.
Resolved	IPY00100886	--	328	SIP Call Control	The SIP Via header in an INVITE message cannot be altered and a second Via header is added instead.
Resolved	IPY00100857	--	328	SIP Call Control	"Call Control Programming Session-Expires" using **gc_SetUserInfo( )** prior to **gc_MakeCall( )** has no effect.
Resolved	IPY00100796	--	328	SIP Call Control	A single 10 ms click is present in the audio stream when the end of the file being played is reached.
Resolved	IPY00100534	--	328	SIP Call Control	After the application receives 31 subscription requests and responds using the **gc_Extension( )** function to accept the requests, HMP starts automatically sending 486 Busy Here for any subsequent SUBSCRIBE. While incoming calls are still accepted, any call to **gc_InvokeXfer( )** fails.
Resolved	IPY00099940	--	328	SIP Call Control	Early media streaming ports are not updated on subsequent SDP.
Resolved	IPY00100730	--	328	SIP Call Control	The **gc_AcceptModifyCall( )** function fails to return a GCEV_ACCEPT_MODIFY_CALL event when HMP receives a number of re-INVITE in a very short order.
Resolved	IPY00100486	--	328	SIP Call Control	Although required to send SIP registration with '*' in the Contact Header, the application receives an IPEC_REG_FAIL_INVALIDALIAS message.
Resolved	IPY00100385	--	328	SS7	Outbound calls contain incorrect CRN and DNIS.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00098898	--	328	Voice	Audio quality degrades when placing a high volume of SIP calls.
Resolved	IPY00093988	--	323	Conferencing	Tone clamping in conferences does not completely remove digits from the conference.
Resolved	IPY00098919	--	323	Drivers	An access violation crash occurs in the *libsrlmt.dll* library.
Resolved	IPY00099842	--	323	Fax	The application receives TFX_FAXERROR without TFX_PHASEB while receiving the fax using the fax pass thru protocol.
Resolved	IPY00099743	--	323	Fax	The **fx_rcvfax( )** function fails to return the TFX_FAXRECV event from an asynchronous call.
Resolved	IPY00099621	--	323	Fax	A Phase E Status of 103 (Bad Response to DCS, training') occurs when sending a T.38 fax.
Resolved	IPY00100032	--	323	IP Media	T.38 Hairpinning fails due to a conflict with DTMF packets.
Resolved	IPY00093233	--	323	Licensing	HMP fails to start when installed in a directory with a long path name.
Resolved	IPY00100305	--	323	SIP Call Control	The **gc_MakeCall( )** function fails shortly after a glare test is performed.
Resolved	IPY00100141	--	323	SIP Call Control	Setting multiple diversion headers of the same type on SIP outbound messages causes an immediate GCEV_DISCONNECT to the **gc_MakeCall( )** function.
Resolved	IPY00100045	--	323	SIP Call Control	Dialogic® HMP software sets multiple diversion headers of the same type on SIP outbound messages.
Resolved	IPY00100034	--	323	SIP Call Control	An application crash occurs when receiving SDP headers with a=rtpmap without parameters.
Resolved	IPY00099719	--	323	SIP Call Control	RequestURI header in INVITE is not setting the tel: format when using the **gc_makeCall( )** function.
Resolved	IPY00099956	--	323	Voice	Dialogic® HMP detects multiple digits when only one is entered.
Resolved	IPY00099495	--	323	Voice	Voice Enabling initial silence termination stops user-defined cadence tone detection.
Resolved	IPY00099980	--	320	Continuous Speech Processing (CSP)	CSP channels stop responding to the **ec_stopch( )** function and become unusable.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00099630	--	320	Fax	The **fx_stopch**(EV_SYNC) function returns before the fax channel goes into the IDLE state.
Resolved	IPY00093540	--	320	PSTN Call Control	The DPNSS protocol fails to send out the Diversion IE on the line.
Resolved	IPY00099861	--	320	SIP Call Control	Outbound SIP calls result in GCEV_TASKFAIL events when SIP 180 Ringing 200 OK responses carry "a=sendonly" media attribute as part of its SDP body for media capabilities.
Resolved	IPY00099736	--	320	SIP Call Control	The GCEV_REQ_MODIFY_CALL event is not returned to application when a re-INVITE is received.
Resolved	IPY00099028	--	320	SIP Call Control	Conflicts occur with the OpenSSL library and HMP.
Resolved	IPY00098933	--	320	SIP Call Control	SIP SDP Capability a=rtpmap:96 g726-32/8000 is not recognized by HMP.
Resolved	IPY00099807	--	320	Voice	Upon calling the **dx_open( )** and **ag_unlisten( )** functions, the D/4PCI-U board does not restore the frontend to its associated voice channel.
Resolved	IPY00093775	--	318	MSML	MSML becomes unstable when INVITEs and BYEs time out.
Resolved	IPY00092928	--	318	MSML	Major MSML failure occurs when using conferencing.
Resolved	IPY00056766	--	317	Conferencing	The conferencing (CNF) API library is allocating incorrect event data size for SRL event data.
Resolved	IPY00098827	--	317	Firmware	Noise is heard whenever a long RFC 2833 DTMF tone is detected or received.
Resolved	IPY00099133	--	317	Firmware	The **ipm_Stop( )** function takes time to return an event under a load condition.
Resolved	IPY00099572	--	317	Global Call IP (SIP)	The **gc_AcceptModifyCall( )** function returns an error in response to the GCEV_REQ_MODIFY_CALL event.
Resolved	IPY00099319	--	317	Global Call IP (SIP)	An application exception occurs due to an array overflow on INVITEs containing a From header larger than 128 bytes.
Resolved	IPY00099149	--	317	Global Call IP (SIP)	The re-INVITE 200 OK response does not contain the media direction attribute.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00098980	--	317	Global Call IP (SIP)	A GCEV_ACCEPT_MODIFYCALL event is not reported to the application after it issues a **gc_AcceptModifyCall( )**.
Resolved	IPY00056913	--	317	Global Call IP (SIP)	The application does not receive a GCEV_ACCEPT_MODIFY_CALL event after the **gc_AcceptModifyCall( )** function generates a 200 OK and the remote side ACKs.
Resolved	IPY00056779	--	317	Global Call IP (SIP)	In T.38 scenario, a GCEV_EXTENSION event to switch to T.38 is received prior to a GCEV_CONNECTED event.
Resolved	IPY00056770	--	317	Global Call IP (SIP)	A memory leak is observed during a during a call hold/transfer scenario.
Resolved	IPY00056719	--	317	Global Call IP (SIP)	When an INVITE is followed by a CANCEL, a GCEV_DISCONNECTED event is reported with a “5487” release code.
Resolved	IPY00099307	--	317	Global Call IP (SIP)	After calling the **gc_DropCall( )** function, a SIP release code 422 MinSE value is received from application.
Resolved	IPY00099177	--	317	Global Call IP (SIP)	When issuing a re-INVITE 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	IPY00099311	--	317	Installation	Blue Screens result when HMP is installed on a server with no NIC.
Resolved	IPY00099227	--	317	IP Media	The **ipm_Listen( )** function fails with an internal error; leaving the channel in an unusable state.
Resolved	IPY00099396	--	317	IP Media Session Control (RTP)	RTP decoding (jitter buffer) fails when large timestamp jumps are encountered.
Resolved	IPY00099150	--	317	IP Media Session Control (RTP)	RTP timestamp jumps occur intermittently.
Resolved	IPY00099200	--	317	PSTN Call Control	When a call is connected with Q.Sig protocol, the application is not able to send a facility message in an already connected NCAS call.
Resolved	IPY00099067	--	317	PSTN Call Control	DMV600BTEP board lock up caused by incorrect parsing of certain misconstrued High Layer Compatibility IEs received in the D-channel.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00094176	--	314	Conferencing	The CNF library allocates incorrect event data size for SRL event data.
Resolved	IPY00098952	--	314	Drivers	A Blue Screen occurs during system reboots when the debugangel service is enabled on systems containing Dialogic® DNI boards.
Resolved	IPY00094003	--	314	Fax	A Server crash occurs when using the Dialogic DNI board and the HP ProLiant DL380 G6.
Resolved	IPY00098947	--	314	IP Media	The **ipm_Listen( )** function stays in the wrong state if an error occurs in firmware; thus any attempt to issue a subsequent **ipm_Listen( )** or **ipm_UnListen( )** function will fail due to invalid state.
Resolved	IPY00094689	--	313	Board Download	HMP does not return to a stable state when errors occur during the HMP start sequence.
Resolved	IPY00094589	--	313	Board Download	An occasional Blue Screen results when the DNI board experiences clock issues.
Resolved	IPY00094197	--	313	Conferencing	If the customer repeats **cnf_OpenParty( )** > **cnf_CloseParty( )** > **cnf_OpenParty( )**, the function sequence actually reduces the number of parties available to the application.
Resolved	IPY00094591	--	313	Diagnostics	On 64-bit Windows platforms, DM3post returns an error when trying to retrieve the results of the last post and when trying to reset a board.
Resolved	IPY00093701	--	313	DM3 CSP	After running for approximately four hours, CSP devices get in a stuck state and the **ec_stream( )** function fails to start.
Resolved	IPY00094151	--	313	Drivers	A Blue Screen occurs once or twice a day due to an HMP driver issue.
Resolved	IPY00094030	--	313	Drivers	A Blue Screen occurs due to invalid stream IDs on driver requests.
Resolved	IPY00093913	--	313	Drivers	Errors occur when loading HPET drivers.
Resolved	IPY00094606	--	313	Fax	Fax channels remain in Fax-Server mode when IPVSC switches to Audio-Only.
Resolved	IPY00093416	--	313	Fax	Interoperability issues with certain gateways that extend the V.21 preamble beyond the T.30 specifications caused HMP to fail responding to the preamble sent by the receiving end.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00056756	--	313	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	IPY00094290	--	313	Installation	HMP fails to start when the Slipstream install option fails to upgrade some services correctly.
Resolved	IPY00093410	--	313	Installation	The Slipstream install option deletes the pdk.cfg file.
Resolved	IPY00093233	--	313	Licensing	HMP fails to start when installed in a directory with a long path name.
Resolved	IPY00093823	--	313	PSTN	The PSTN Diag tool fails to launch on a Windows Server 2008 system.
Resolved	IPY00094359	--	313	SIP Call Control	The application crashes while sending a NOTIFY ACCEPT when the SIP Call-ID is more than 63 characters long.
Resolved	IPY00056844	--	313	SIP Call Control	Header fields are incorrectly sent in the outgoing SIP INVITE message.
Resolved	IPY00056843	--	313	SIP Call Control	DTMF is not received when receiving a re-INVITE with SDP in the ACK message.
Resolved	IPY00056776	--	313	SIP Call Control	After the gc_MakeCall( ) function time-outs (when set to more than 180 seconds), HMP fails to send a CANCEL to far end. As a result, the call stays in ringing state at far end.
Resolved	IPY00056738	--	313	SIP Call Control	After the gc_MakeCall( ) function time-outs (when set to more than 180 seconds), HMP fails to send a CANCEL to far end. As a result, the call stays in ringing state at far end.
Resolved	IPY00098894	--	313	Voice	The **dx_reciottdata( )** function stops recording the file unexpectedly with a 2k buffer size set via the **dx_setchxfercnt( )** function.
Resolved	IPY00094154	--	313	Voice	When called synchronously, the dx_dial( ) function for call progress analysis always returns 0 instead of the predefined Call Result value.
Resolved	IPY00056539	--	310	Configuration	The desired MAC address in DCM cannot be selected after changes are made to a NIC and VLAN configuration.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00056601	--	310	Fax	The TIF image on the receiving fax side is corrupted. The sender indicates the fax was sent successfully, and no errors appear in the RTF logs on either side.
Resolved	IPY00093590	--	310	Global Call IP (SIP)	An INFO message in two different directions was flagged as an error and not passed to the application.
Resolved	IPY00056659	--	310	Global Call IP (SIP)	The **gc_InvokeXfer( )** function generates an exception.
Resolved	IPY00056642	--	310	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	IPY00093902	--	310	IP Media	The **ipm_DisableEvents( )** function does not work when the EVT_TELEPHONY value is specified for RFC 2833 tones.
Resolved	IPY00093971	--	310	MSML	MSML <createconference > requests, sent as payloads of SIP INFO messages with <audiomix> and no child element, result in the MSML media server crashing.
Resolved	IPY00094125	--	310	Multimedia	Calling **mm_Record( )** fails with an **mm_Play( )** error.
Resolved	IPY00094061	--	310	Voice	Calling the **dx_reciottdata( )** function with initial silence compression record enabled results in more than four seconds at the beginning of the file.
Resolved	IPY00093859	--	310	Voice	Play and record length values are not coming through MSML properly.
Resolved	IPY00093815	--	310	Voice	A blue screen occurs during heavy memory usage with no notification to the application. Refer to Section 1.49, "New TDX_DRVNOMEM Event", on page 137.
Resolved	IPY00093453	--	310	Voice	The **dx_wtring( )** function fails to return to the calling thread during a synchronous mode multithreading application.
Resolved	IPY00093361	--	307	Conferencing	The **cnf_OpenParty( )** function does not return an error when it exceeds the maximum number of parties defined by the HMP license.
Resolved	IPY00093375	--	307	Diagnostics	The its_sysinfo tool hangs when executed on 64-bit systems.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00093288	--	307	Fax	No termination event is sent following a call to the fx_stopch( ) function. This results in a hung port.
Resolved	IPY00093409	--	307	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	IPY00093385	--	307	Global Call IP (SIP)	When PRACK is enabled, the gc_AcceptCall( ) function does not generate a 18X response, causing the SIP call stop receiving the INVITE from the remote end.
Resolved	IPY00093384	--	307	Global Call IP (SIP)	The application receives the GCEV_TASKFAIL event as a response to gc_AcceptModifyCall( ) during a double re-INVITE.
Resolved	IPY00093270	--	307	Global Call IP (SIP)	A memory leak is observed during a SIP re-INVITE.
Resolved	IPY00092371	--	307	Global Call IP (SIP)	Call transfers return an error even though the transfers appear to succeed.
Resolved	IPY00056613	--	307	Global Call IP (SIP)	SIP calls are rejected with response “484 Address Incomplete'”.
Resolved	IPY00056610	--	307	Global Call IP (SIP)	The HMP SIP stack sends “400 Bad Request” in response to a second NOTIFY message received during a call transfer scenario.
Resolved	IPY00056607	--	307	Global Call IP (SIP)	An additional GCEV_TASKFAIL event is posted right after a GCEV_INVOKE_XFER_FAIL event when the gc_InvokeXfer( ) function is called.
Resolved	IPY00056559	--	307	Global Call IP (SIP)	The gc_InvokeXfer( ) function returns a GCEV_INVOKE_XFER_FAIL error even though the transfer appears to succeed.
Resolved	IPY00056458	--	307	Global Call IP (SIP)	The gc_acceptModifyCall( ) function fails on a transfer re-INVITE following an UPDATE.
Resolved	IPY00093672	--	307	Installation	The HMP Uninstall script does not complete successfully on systems configured with a mixture of DNI and JCT boards.
Resolved	IPY00093501	--	307	IP Media	The sr_getboardcnt(DEV_CLASS_IPT, &nIPTBoards) function returns nIPTBoards=0 instead of actual number of IP boards configured in GC_START_STRUCT in the IPCCLIB_START_DATA data structure.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00093232	--	307	MSML	SIP stack prints out an error when the direction mode is set in the SDP multiple times for the same session.
Resolved	IPY00093627	--	307	PSTN Call Control	Issues occur when sending Diversion IE. Scenario: An outbound call is made and the switch informs that the number is diverted. HMP needs to make the second call AND set the DIVERSION_IE to tell the new switch what number was originally dialed.
Resolved	IPY00056584	--	307	SIP Call Control	A formatted Uniform Resource Identifier (URI) is automatically replaced with a sip: prefix when trying to format a Request-URI with a tel: prefix.
Resolved	IPY00093013	--	307	Voice	Incorrect signal detector behavior caused sporadic missing digits.
Resolved	IPY00093246	--	303	Configuration	HMP does not start when a system has a Flex-10 ethernet module as its only NIC.
Resolved	IPY00093097	--	303	DCB Conferencing	A memory leak occurs while using the DCB API to add or remove a conference party.
Resolved	IPY00093128	--	303	Dialogic® HMP Software	A system crash with a blue screen occurs with the initial silence termination (TF_SETINIT) feature enabled.
Resolved	IPY00092139	--	303	Installation	When uninstalling HMP, the install script attempts to stop the SNMP service even though SNMP was not selected during installation.
Resolved	IPY00056482	--	303	Installation	When using the response file to do silent install, HMP creates ProgramData on the C drive even though it is configured to be on the D drive.
Resolved	IPY00093215	--	303	IP Media	Improper video jitter buffer handling caused an exception in HMP.
Resolved	IPY00093117	--	303	IP Media Session Control (RTP)	No audio (RTP packets) are sent from HMP when establishing a SIP call using Global Call 3rd-party call control and Secure RTP.
Resolved	IPY00093089	--	303	SIP Call Control	HMP incorrectly responds to a refresh SUBSCRIBE message with a 405 rather than a 481.
Resolved	IPY00056505	--	303	SIP Call Control	A parser error occurs when URN is used in the Request-URI header of an outbound INVITE.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00056289	--	303	SIP Call Control	The GCEV_TASKFAIL event is received after issuing gc_AcceptModifyCall( ) followed by gc_ReqModifyCall( ) to change the channel into an inactive state.
Resolved	IPY00092995	--	299	Configuration	The its_sysinfo tool cannot find RTF logs on 64-bit versions of Windows 7 and higher systems.
Resolved	IPY00092835	--	299	Firmware	Dialogic® HMP fails to start on a system with 24 cores.
Resolved	IPY00092725	--	299	Global Call IP (SIP)	The RTF logs show incorrect IP Call Control event types.
Resolved	IPY00056281	--	299	Licensing	When running Device Map Dump (devmapdump) on a 1000+ port license, the output infinitely loops.
Resolved	IPY00092965	--	299	MSML	The MSML Media Server service crashes and requires a manual restart.
Resolved	IPY00092794	--	299	SIP Call Control	Two Refer-To fields generate when the application attempts to redefine the Refer-To header before calling the **gc_InvokeXfer( )** function.
Resolved	IPY00091574	--	299	SIP Call Control	SDP session ID and revisions are incremented by 1 when sending a Session Timer INVITE.
Resolved	IPY00091274	--	299	SIP Call Control	When a SIP UPDATE is received after answer with SDP, it is not reported to the application. Instead, the underlying stack responds with a 200 OK without an SDP_ANSWER.
Resolved	IPY00056119	--	299	SIP Call Control	Dialogic® HMP software only allocates a maximum of 50 (hardcoded) SIP SUBSCRIBEs.
Resolved	IPY00056234	--	299	Voice	Under certain race conditions, after the **dx_reciottdata( )** function is issued followed by the **dx_stopch( )** function, the next call to **dx_reciottdata( )** terminates with a termination condition immediately, and prematurely.
Resolved	IPY00092629	--	297	Global Call IP	An issue occurs with **gc_GetMetaEvent( )** where memory associated with extraEvtdatap is not released when a device is closed.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00092511	--	297	Global Call IP (SIP)	When a SIP UPDATE is received after answer with SDP, it is not reported to the application. Instead, the underlying stack responds with a 200 OK without an SDP_ANSWER.
Resolved	IPY00092276	--	297	Global Call IP (SIP)	The **gc_ResultInfo( )** function does not return consistent error codes when an inbound call is rejected after issuing **gc_AnswerCall( )**.
Resolved	IPY00056135	--	297	Global Call IP (SIP)	Dialogic® HMP software incorrectly sends unsupported ptime media attribute in an outgoing SIP 200 OK message.
Resolved	IPY00079902	--	297	IP Media	Distorted voice results when recording RTP extension - G.711 A-law.
Resolved	IPY00080769	--	297	IP Media Session Control RTP	When RTP containing Extension Headers is received (correctly) and hairpinned, the x bit is set, but the remainder of the packet does not contain the extensions.
Resolved	IPY00092242	--	297	PSTN Call Control	NFAS issues occur with the backup D-channel when connecting to the Verizon switch.
Resolved	IPY00056096	--	297	Voice	When a TDX_CST event is reported, DX_CST data can be retrieved via the **sr_getevtdatap( )** function.The DX_CST.cst_event indicates either DE_SILON, DE_SILOFF, corresponding to Silent / Non-Silent detected. DX_CST.cst_data should indicate timestamp information, however, when using HMP software the data field always equals zero (0).
Resolved	IPY00092148	--	296	IP Media Session Control (RTP)	When the RTP timestamp is sharply shifted, Dialogic® HMP software stops decoding RTP packets and puts silence on the TDM bus stream.
Resolved	IPY00056036	--	296	SIP Call Control	SDP information is missing from the second INVITE after the stack authenticates the resend of the initial INVITE.
Resolved	IPY00056033	--	296	SIP Call Control	No 48x response to a SIP INVITE message is received after the licensed channel limit is reached.
Resolved	IPY00056010	--	296	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.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00092090	--	291	Conferencing (DCB)	When the application calls the **dcb_delconf( )** function followed by the **dcb_remfromconf( )** function, the **dcb_delconf( )** function blocks for about 10 seconds before it returns a -1 failure.
Resolved	IPY00092209	--	291	Fax	Inbound faxes are received with some of the pages cut off. The missing part appears blank when displayed with an image viewer.
Resolved	IPY00091862	--	291	PSTN Call Control	The application is unable to retrieve CLI from DPNSS SSRM(I).
Resolved	IPY00055976	--	291	SIP Call Control	When the MIME feature is turned on, the wrong INVITE message is sent after CANCEL within same TCP connection. As a result, Dialogic® HMP Software rejects incoming calls with "getFreeBuffer: no buffer available."
Resolved	IPY00091855	--	289	Conferencing	The third party in a conference hears audio breaking up while two other parties talk to each other.
Resolved	IPY00091795	--	289	Conferencing	The **cnf_OpenParty( )** function returns the CNFEV_OPEN_PARTY event prematurely, causing issues in multithreaded applications.
Resolved	IPY00055504	--	289	Conferencing	The CNF_ADD_PARTY event returns the incorrect length from sr_getevtlen.
Resolved	IPY00091982	--	289	Configuration	A TDX_RECORD completion event is not received after calling the **dx_stopch( )** function to terminate record activity.
Resolved	IPY00091502	--	289	Configuration	The use of the NCM API to restart the Dialogic® Service results in a potential hang during service shutdown.
Resolved	IPY00055503	--	289	Device Management	The dev_Disconnect( ) function hangs without returning an error if the same device is specified by another call to dev_Connect( ).
Resolved	IPY00091783	--	289	Drivers	A Blue Screen of Death (BSOD) results after setting the Signal Detector minimum energy parameter (0x070b).
Resolved	IPY00091867	--	289	Global Call (IP)	A GCEV_ACCEPT_MODIFY_CALL was incorrectly sent to the application in a supervised transfer scenario where the media was disconnected while the call was moving into a disconnected state.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00091103	--	289	Global Call IP (SIP)	The application receives two termination events, GCEV_TASKFAIL and GCEV_DISCONNECTED, after calling the **gc_MakeCall( )** function.
Resolved	IPY00055898	--	289	Global Call IP (SIP)	After setting the IPPARM_SIGNALING_DEFERRED parameter to defer SIP messages, the **gc_ReleaseCallEx( )** function fails to return an event when the call transfer (**gc_InvokeXfer**) completes.
Resolved	IPY00055887	--	289	Global Call IP (SIP)	The application receives a GCEV_TASKFAIL event if the remote end sends an empty re-INVITE during a fax session.
Resolved	IPY00091084	--	289	IP Media	Incorrect payload type packets break the native hairpinned video stream.
Resolved	IPY00081381	--	289	PSTN Call Control	The `gctload` command shuts down after 10 seconds of not being able to establish the communication with the SS7G21 Signaling Server. (See the Documentation Update in the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide.)
Resolved	IPY00091932	--	289	Voice	Access Violation was seen when calling the **dx_reciottdata( )** function due to a corruption in RTF.
Resolved	IPY00091878	--	289	Voice	There is a short delay before TDX_PLAY events are returned to the application.
Resolved	IPY00091823	--	289	Voice	Using the **dx_setsvcond( )** function for pause/resume play and volume adjustment on detection of the specified DTMF digit fails.
Resolved	IPY00091609	--	286	Configuration	The DCM automatic startup menu choices for Windows Server 2008 are greyed out.
Resolved	IPY00091507	--	286	Drivers	An IRQ conflict causes the Dialogic® HMP Software installation to fail on an NEC Express Server with Windows Server 2008 (32 bits).
Resolved	IPY00091521	--	286	Global Call (SIP)	In 1PCC mode, a re-INVTE with multiple m-lines does not report all capabilities. The GCEV_REQ_MODIFY_CALL should report capabilities from all m-lines, but only capabilities from the first m-line get reported.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00091411	--	286	Global Call (SIP)	For a session timer refresh which contains multiple "m=" lines in the SDP, only the capabilities from the second line are reported. In this case, since the second line has T.38 capabilities specified, the application logic rejects the re-INVITE with a 488 message.
Resolved	IPY00091349	--	286	Global Call (SIP)	Although Dialogic® HMP Software receives SIP UPDATE, it does not use the updated Contact Address as expected.
Resolved	IPY00055808	--	286	Global Call (SIP)	When SDP is present in the 18x SIP message, it does not appear in the parmblk block attached to the GCEV_EXTENSION message.
Resolved	IPY00091782	--	286	Voice	Calling the **dx_playiottdata( )** function fails to return a result value due to an access violation within its process to get the play started.
Resolved	IPY00091303	--	283	Conferencing	An established conference fails when attempting to remove a monitor from the conference.
Resolved	IPY00091361	--	283	Global Call IP (SIP)	An application error (offset 000d959, librtfmt.dll) occurs.
Resolved	IPY00091348	--	283	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	--	283	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	IPY00091278	--	283	IP Media Session Control RTP	Dialogic® HMP software does not capture all DTMFs from the RTP stream.
Resolved	IPY00091360	--	283	Voice	Channels get stuck after a TDX_ERROR event is returned from **dx_reciottdata( )** in user I/O mode.
Resolved	IPY00091247	--	283	Voice	Multiple race conditions occur.
Resolved	IPY00091219	--	283	Voice	The system is not able to receive new calls.
Resolved	IPY00055624	--	279	Device Management	An error occurs when the application calls **dev_Connect( )** / **dev_Disconnect( )** from multiple threads.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00091137	--	279	Driver	Download fails after a few attempts. The size of the license appears to affect how soon the download will fail.
Resolved	IPY00090867	--	279	Driver	A BSOD may occur during an out of memory condition in the dlgcmpd driver.
Resolved	IPY00091140	--	279	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	--	279	Fax	HMP Fax send/receive failures occur causing all fax channels to be busy.
Resolved	IPY00091138	--	279	Fax	The fax application locks when sending a fax.
Resolved	IPY00091093	--	279	Fax	A fax termination issue occurs while using T.30 FAX with ECM enabled (error correction mode) on HMP.
Resolved	IPY00090932	--	279	Fax	Phase B fails when the remote side responds with a Digital Identification Signal (DIS) to an initial DIS from HMP.
Resolved	IPY00082125	--	279	Fax	The fax server freezes sporadically and the HMP fax application stops responding. Only a system reboot temporarily resolves the problem.
Resolved	IPY00091023	--	279	Global Call	The **gc_DropCall( )** function fails and causes a hung port.
Resolved	IPY00091162	--	279	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	--	279	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	IPY00055507	--	279	Global Call IP (SIP)	In 1PCC mode, SIP re-INVITE Codec Reporting appears to be wrong or corrupted.
Resolved	IPY00055506	--	279	Global Call IP (SIP)	HMP transfer disconnects after 200 OK is received for NOTIFY when attempting to make an outbound transfer using NOTIFY messages to track the status to the transferring party.
Resolved	IPY00091132	--	279	IP Media	Audio transcoding between G.711 to G.729 is not successful.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00055625	--	279	IP Media	Outbound 1PCC SIP call connects as G729AB but reports the receive GCEV_EXTENSION as G729A.
Resolved	IPY00091081	--	279	Voice	The **dx_play( )** function hangs when receiving 92 calls simultaneously.
Resolved	IPY00091050	--	279	Voice	The application crashes when calling the **dx_pause( )** function.
Resolved	IPY00047710	--	279	Voice	Stuck channels result on HMP devices during record.
Resolved	IPY00047709	--	279	Voice	Multiple channels get stuck channel when the application uses UIO (User-defined Input/Output) and UIO write returns -1.
Resolved	IPY00082301	--	275	CSP	During a CSP load, ec_reciottdata( ) 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. The following parameters can be added to the .config file to adjust the number of Tone Templates and Defines. [sigDet] SetParm=0x0700,128 ! SD_ParmMaxTnTmplts (default=128) SetParm=0x0701,128 ! SD_ParmMaxTnDefs (default=128)
Resolved	IPY00081518	--	275	DM3	DIVERSION_IE and DIVERSION_VALIDATION_IE are not received when calling the **gc_GetSigInfo( )** after receiving GCEV_OFFERED and GCEV_PROCEEDING events.
Resolved	IPY00090705	--	275	Firmware	A fatal firmware crash was observed upon a request to stop a playback operation (dx_playiottdata) due to a race condition to change its echo reference port, which rendered the HMP board unoperational.
Resolved	IPY00055439	--	275	Global Call IP (SIP)	Dialogic® HMP Software cannot issue a re-INVITE on a channel after rejecting a previous remote re-INVITE.
Resolved	IPY00090969	--	275	Voice	Although the **dx_reciottdata( )** function issues successfully, it does not return an event and the channel remains in a record state.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00081873	--	275	Voice	The **dx_recordiottdata( )** function fails internally, and never returns an event to the application.
Resolved	IPY00080914	--	272	Conferencing (DCB)	An error is received when calling **dcb_evtstatus( )** function.
Resolved	IPY00080842	--	272	Firmware	Customer observes different QoS Alarm default parameter values than what is expected.
Resolved	IPY00090750	--	272	Global Call IP (SIP)	No Response is sent to invalid INVITE.
Resolved	IPY00090646	--	272	Global Call IP (SIP)	In 1PCC mode, the SDP answer differs for 18x and 200 OK.
Resolved	IPY00090645	--	272	Global Call IP (SIP)	Dialogic® HMP Software SIP stack incorrectly sends a “486 Busy Here” message in response to SIP PRACK message received.
Resolved	IPY00082165	--	272	Global Call IP (SIP)	Dialogic® HMP software failed to generate multiple PRACK responses.
Resolved	IPY00081732	--	272	Global Call IP (SIP)	A GCEV_REQ_MODIFY_CALL event is reported while a previous **gc_AcceptModifyCall( )** is in progress.
Resolved	IPY00081695	--	272	IP Media Session Control (RPT)	Dialogic® HMP software fails when the application calls the **ipm_ModifyMedia( )** function to change both RTP/RTCP port numbers.
Resolved	IPY00082081	--	271	Licensing	Dialogic® HMP software fails with a DongleManager error in the Windows event log.
Resolved	IPY00081235	--	271	Licensing	License activation fails when using ethernet ports. Any subsequent attempt to start services fails with an "invalid host id " error message.
Resolved	IPY00081529	--	270	Configuration	The Dialogic® HMPDM3Config service fails to install correctly.
Resolved	IPY00082126	--	270	Dialogic® HMP software	Event log errors are reported when installing Dialogic® HMP software without Dialogic® DNI board support.
Resolved	IPY00055436	--	270	Drivers	Channels hang when the driver does not respond to API calls. A restart of Dialogic® Services is required to recover the system.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00055290	--	270	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	IPY00090707	--	270	Installation	During uninstalled dlgcmpd.sys and dlgcmcd.sys drivers were not removed from the system.
Resolved	IPY00081251	--	270	Installation	The dlgcleanup.bat tool takes a long time to scan the system disk and delete log files.
Resolved	IPY00082088	--	270	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	IPY00055020	--	267	Global Call IP (SIP)	Proxy-Authentication for registration and re-registration encounters problems with multi-user registrations.
Resolved	IPY00081665	--	267	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	IPY00054991	--	265	IP Media Session Control (RTP)	RTP receipt notification is missing when audio hairpinning.
Resolved	IPY00081465	--	265	Voice	Memory increases over time when issuing the **dx_reciottdata( )** function with a GSM codec.
Resolved	IPY00081096	--	262	Voice	Voice device audio is not heard from the octal-span Dialogic® DNI board.
Resolved	IPY00081159	--	262	Conferencing (DCB)	The **dcb_delconf( )** function fails without returning an error message.
Resolved	IPY00081284	--	262	Global Call IP (SIP)	Session Timer settings appear to be reset upon a re-INVITE to the Dialogic® HMP software.
Resolved	IPY00081264	--	262	Global Call IP (SIP)	UAS Session Timer programming does not survive the application's re-INVITE request.
Resolved	IPY00081142	--	262	Global Call IP (SIP)	The Proxy Bypass feature is not working properly.
Resolved	IPY00081132	--	262	Global Call IP (SIP)	The SIP hold/retrieve call scenario using the **gc_ReqModifyCall( )** and **gc_AcceptModifyCall( )** functions does not perform as expected.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00081061	--	262	Global Call IP (SIP)	The SIP stack does not negotiate correctly when VAD is specified as DON'TCARE on the local side. VAD-specific coders in an incoming SDP are not accepted.
Resolved	IPY00080944	--	262	Global Call IP (SIP)	Inbound calls are rejected due to maximum call legs allocated even though there is sufficient capacity on the node.
Resolved	IPY00054970	--	262	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	--	262	Global Call IP (SIP)	A Call Transfer issue causes REFER to fail after receipt of an INVITE.
Resolved	IPY00080918	--	262	IP Media	The application does not wrap the user=phone inside the < > delimiters in the To Header.
Resolved	IPY00080396	--	256	Configuration	The application cannot select the E1 line side CAS protocol in DCM.
Resolved	IPY00080694	--	256	Dialogic® HMP software	The *RvSdpLogFile.log* file continues to grow on a per call basis using up disk space.
Resolved	IPY00080407	--	256	Dialogic® HMP software	The apicdrvio timer driver is disabled in Dialogic® HMP software.
Resolved	IPY00080860	--	256	Fax	Fax device fails to respond to specific inbound Digital Command Signals (DCS) sent by the remote-end.
Resolved	IPY00080800	--	256	Global Call IP (SIP)	A duplicate "a=inactive" line occurs in an SDP message when a re-INVITE is sent using the **gc_ReqModifiyCall( )** function.
Resolved	IPY00080772	--	256	Global Call IP (SIP)	183 messages cannot be PRACKed if they generate a GCEV_PROGRESSING event.
Resolved	IPY00080693	--	256	Global Call IP (SIP)	The application cannot add Session Description Protocol (SDP) to a SIP UPDATE message.
Resolved	IPY00080468	--	256	Global Call IP (SIP)	Dialogic® HMP software is unable to correctly register the application on a SIP Proxy.
Resolved	IPY00080308	--	256	Global Call IP (SIP)	The **gc_ModifyCall( )** function fails with SIP Session Timers due to a SIP re-INVITE collision issue.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00080011	--	256	Global Call IP (SIP)	A GCEV_TASKFAIL event is returned on receipt of a LowBitRate HOLD re-INVITE 200 OK.
Resolved	IPY00080010	--	256	Global Call IP (SIP)	An inbound SIP BYE from a SNOM phone is rejected with a "501 unimplemented" message.
Resolved	IPY00080640	--	256	PSTN Call Control	Although a call connects, the switch does not send audio because the appropriate message is not sent when making the call (missing CAM/NAM and CCM).
Resolved	IPY00080125	--	255	Fax	A race condition in T.38 fax caused send and receive errors.
Resolved	IPY00080124	--	255	Fax	When the application receives more than 100 pages of fax, the page number restarts from zero (0).
Resolved	IPY00080152	--	253	Installation	Problematic issues occur using the Advanced Programmable Interrupt Controller (APIC) timer.
Resolved	IPY00080031	--	253	IP Media	Loss of voice occurs while receiving RTP (G.711) with long inter-packet gaps (DTX), and time stamps "skewed" against the system clock.
Resolved	IPY00079937	--	253	Licensing	A problem occurs activating a license for HMP and DNI when starting media.
Resolved	IPY00080086	--	253	Voice	A Blue screen occurs when producing EED logs.
Resolved	IPY00045481	--	251	Configuration	The number of frame timers (1024) specified in the PCD file is not enough for a 1000 channel license.
Resolved	IPY00078684	--	251	Global Call	The application hangs while waiting for a GCEV_UNBLOCKED event.
Resolved	IPY00044285	--	251	IP Media	The **ipm_Listen( )** function fails with the error 0xa=Synchronization object timeout.
Resolved	IPY00079186	--	251	MSML	The MSML media server deadlocks when dialog exists at the same time a dialog end request is received.
Resolved	IPY00043527	--	251	MSML	A CNF party level resource receives a conference level event.
Resolved	IPY00079866	--	251	PSTN	No response is sent back to the application upon receipt of a user-to-user service 1 or 2 request.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00079825	--	251	PSTN	When receiving an IAM with the continuity check indicator set to spare (illegal value), the application handles it as if a "continuity check required" indicator was received.
Resolved	IPY00045524	--	248	Device Management	The **dev_connect( )** function fails when used between M3G and DNI devices.
Resolved	IPY00079551	--	248	PSTN	IAM messages exceed the maximum allowed.
Resolved	IPY00079764	--	248	Voice	The **dx_playiottdata( )** function does not return a TDX_PLAY event when playing *.wav* file.
Resolved	IPY00079353	--	248	Voice	Audio is missing at the end of recorded files.
Resolved	IPY00079678	--	246	Dialogic® HMP software	Echo cancellation on DNI boards does not function properly.
Resolved	IPY00079435	--	246	Fax	A high percentage of inbound faxes are received as blank or partly cut-off.
Resolved	IPY00079763	--	246	IP Media	Synchronization object timeout errors are reported.
Resolved	IPY00079655	--	245	Device Management	RFC2833 digits are not resent when native hairpinned using the **dev_PortConnect( )** function.
Resolved	IPY00079601	--	245	Global Call IP (SIP)	No response is returned when an INVITE with a parse error is received.
Resolved	IPY00079396	--	241	Conferencing	After running for some time, the application thread hangs after adding one party to a conference using the **ms_addtoconf( )** function.
Resolved	IPY00079346	--	241	Fax	After calling the **fx_stopch( )** function at Phase C of the fax receive side, there is no TFX_FAXERROR event report nor is the state of fax channel IDLE.
Resolved	IPY00079344	--	241	Fax	Dialogic® HMP software stores received faxes with a non-metric resolution calibration.
Resolved	IPY00079530	--	241	Global Call IP (SIP)	The application is unable to place a call on hold and then retrieve it when using G.723.
Resolved	IPY00079518	--	241	Global Call IP (SIP)	After receiving the GCEV_TASKFAIL event, the **gc_ResetLineDev( )** function does not return the GCEV_RESETLINEDEV event.
Resolved	IPY00079509	--	241	Global Call IP (SIP)	The Min-SE header is present in the outgoing INVITE even though session timers on the channel are disabled.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00079483	--	241	Global Call IP (SIP)	Cannot initiate sendOnly or recvOnly calls using parameter IP_CAP_DIR_LCLSENDONLY or IP_CAP_DIR_LCLRECVONLY.
Resolved	IPY00079466	--	241	Global Call IP (SIP)	The supported 'replaces' header has a [space] before carriage return line feed (CRLF) causing an ingress call to fail.
Resolved	IPY00079393	--	241	Global Call IP (SIP)	The SIP Allow header is omitted in response messages to inbound calls.
Resolved	IPY00079374	--	241	Global Call IP (SIP)	Dialogic® HMP software does not obtain the origination point in multi-step diversions.
Resolved	IPY00044782	--	237	Dialogic® HMP software	A race condition was discovered while decrementing a reference count.
Resolved	IPY00079254	--	237	Fax	A sporadic failure occurs with the Dialogic® HMP software.
Resolved	IPY00079177	--	237	Fax	The fax port does not receive T.38 messages from the remote end.
Resolved	IPY00079123	--	237	Fax	The application receives a TFX_FAXRECV message, but the TIF file contains a "black" image.
Resolved	IPY00079251	--	237	Global Call IP (SIP)	The SIP Allow header does not include the 'INFO' method.
Resolved	IPY00079125	--	237	Global Call IP (SIP)	The application is unable to send SDP in 183 using the **gc_SipSessionProgress( )** function.
Resolved	IPY00079108	--	237	Global Call IP (SIP)	IPPARM_OFFERED_FASTSTART_CODER/ GCSET_CHAN_CAPABILITY reporting for SIP G723.1 is always chosen at the default bit rate of 6.3k if remote side does not specify the bit rate.
Resolved	IPY00079161	--	237	IP Media	The sequence number restart is not correctly handled by Dialogic® HMP software.
Resolved	IPY00079116	--	237	IP Media	G723.1 rule does not allow the receipt of 6.3 frames in 5.3 frame mode.
Resolved	IPY00079160	--	237	PSTN Call Control	No ANI is returned by the **gc_GetCallInfo( )** function when the numbering plan is private.
Resolved	IPY00079068	--	237	PSTN Call Control	The **gc_SetConfigData( )** function fails when when 96 IP media devices are bridged to 96 DNI devices.
Resolved	IPY00045369	--	232	Global Call IP (SIP)	A returned IP Address of 0.0.0.0 in a 183 message causes the media devices to fail until a reboot is performed.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00040428	--	232	IP Media	The SRL high priority handler for IP Media library events is not enabled after the second call to **gc_Start( )**.
Resolved	IPY00078754	--	232	Voice	Calling the **dx_stopch( )** function does not stop the channel from playing.
Resolved	IPY00078720	--	232	Voice	TDX_CST and/or TDX_PLAYTONE events are missing.after the application calls the **dx_playtone( )** function.
Resolved	IPY00078639	--	229	Dialogic® HMP software	Driver verifier failures occur and the system reboots with a cstream locking problem.
Resolved	IPY00078626	--	229	Dialogic® HMP software	Event viewer errors appear at random times.
Resolved	IPY00043492	--	229	Dialogic® HMP software	Sound quality issues result when both speakers talk at the same time with Non Linear Processing (NLP) clipping low level audio.
Resolved	IPY00045058	--	229	IP Media	A stack corruption occurs with the IP Media server demo in the file *CVoiceStateMachine.cpp*.
Resolved	IPY00078827	--	228	CSP	During Dialogic® HMP load testing, the 67th CSP channel fails ec_stream callback.
Resolved	IPY00078606	--	228	Fax	When the fax channel begins to receive, the application randomly stops the channel, and all of the fax channels hang up.
Resolved	IPY00078742	--	228	Global Call IP (SIP)	The application cannot set a 183 response in **gc_AcceptCall( )** on a per call basis.
Resolved	IPY00078334	--	228	Global Call IP (SIP)	When an ingress call that reaches alerting is disconnected by the originator, Dialogic® HMP sends an SDP in the CANCEL 200 OK message.
Resolved	IPY00045313	--	226	Conferencing (DCB)	The **dcb_addtoconf( )** function returns -1 when a multi connect is attempted.
Resolved	IPY00078489	--	226	Global Call IP (SIP)	A 1PCC SDP response causes interaction issues (RTP encryption).
Resolved	IPY00045458	--	226	Installation	The Install module, *InstallDrv.exe*, does not correctly install the Dialogic® HMP virtual device.
Resolved	IPY00044215	--	226	IP Media	A Dr.Watson event occurs, causing the application to fail.
Resolved	IPY00045502	--	225	Dialogic® HMP software	The system crashes during login after installing the Dialogic® HMP Software Release 3.0 SU199.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00045371	--	225	Fax	The fax channel fails to respond after calling the **fx_stopch( )** function.
Resolved	IPY00045006	--	225	Fax	An exception in the Dialogic® HMP fax component causes resources to get stuck in a non-idle state.
Resolved	IPY00044978	--	225	Fax	The **fx_rcvfax( )** function fails with -1 error.
Resolved	IPY00044750	--	225	Fax	The **fx_stopch( )** function does not return an expected termination event.
Resolved	IPY00044620	--	225	Fax	The Fax resource remains stuck in a STOP state.
Resolved	IPY00045374	--	225	Global Call IP (SIP)	When GCMSK_PROGRESS is not enabled and a 180 and 183 response is received, two GCEV_ALERTING events are sent to the application resulting in a GCEV_TASKFAIL event.
Resolved	IPY00045174	--	225	Global Call IP (SIP)	While using an Outbound Proxy configuration, the Route Header from the SIP message is not being used for routing the call.
Resolved	IPY00045130	--	225	Global Call IP (SIP)	EVENT_MODIFY_START occasionally fails when re-INVITE message arrive quickly.
Resolved	IPY00044853	--	225	Global Call IP (SIP)	Currently, any attempt to pass more than 255 bytes to the **gc_InvokeXfer( )** function will cause the function to fail.
Resolved	IPY00078449	--	225	IP Media	The **ipm_GetQoSThreshold( )** function fails when the QoS Parameter ID specified is invalid.
Resolved	IPY00045260	--	225	IP Media	The IP device stops transmitting audio towards the TDM bus after certain a parameter is applied. As a result, the **dx_getdigit( )** function does not terminate and keeps waiting for a digit.
Resolved	IPY00045231	--	225	IP Media	When ipm_StartMedia is performed in SEND_ONLY mode, IP devices fail.
Resolved	IPY00044995	--	225	IP Media	When two separate applications are running with different IP Addresses, both applications work fine with only G.711 specified in the SDP. As soon as application 1 specifies G.729 in the SDP, it crashes when it receives the first IP call.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00045362	--	225	MSML	The media server stops responding to incoming calls when a conference party is reset before a dev_disconnect event is received.
Resolved	IPY00045209	--	225	Voice	The **dx_distone( )** function does not disable user-defined ascii tone detection.
Resolved	IPY00045207	--	225	Voice	An unexpected noise clip is heard before the application calls the **dx_play( )** function to play the audio file.
Resolved	IPY00045164	--	225	Voice	Voice data is not being read properly.
Resolved	IPY00045120	--	225	Voice	The **dx_playiottdata( )** function does not complete successfully (in either synchronous or asynchronous mode) when certain prompt offsets and lengths are set within the play structure.
Resolved	IPY00044630	--	221	Dialogic® HMP software	An "Unhandle exception" is found during the **gc_Start( )** function when running the application through VC++ .Net 2003 debug mode.
Resolved	IPY00044822	--	218	IP Media	In a "hairpinned" situation, the **ipm_ModifyMedia( )** function fails to switch codecs; resulting in garbled speech.
Resolved	IPY00044700	--	218	Dialogic® HMP software	The Radvision asn.1 library crashes.
Resolved	IPY00044705	--	218	Global Call IP (SIP)	The Dialogic® HMP provisional timer expires and generates an internal GCEV_DISCONNECT event.
Resolved	IPY00044633	--	218	Global Call IP (SIP)	After initializing the outbound proxy SIP port to 7070, setting the IP Header "Route: <sip:gw1.dialogic.com:7070;lr>\0" causes the SIP port to change to 5060.
Resolved	IPY00044404	--	218	MSML	The Dialogic® HMP MSML server initialization generates too many messages for the driver's orphan message queue.
Resolved	IPY00044976	--	218	Voice	HeapCreate is failing but memory is still being used.
Resolved	IPY00044366	--	214	Conferencing	"Dead air" occurs after a few successful conference calls.
Resolved	IPY00044684	--	214	Conferencing (DCB)	The **dcb_removefromconf( )**, **dcb_deleteconf( )** and **dcb_deleteallconf( )** functions do not wait User Mode Bridge Controller (UMBC) completion events.
Resolved	IPY00044437	--	214	Fax	Fax channels hang intermittently.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00044578	--	214	Global Call IP (SIP)	A GCEV_ALERTING event is not generated for an ingress 183 Session Progress response.
Resolved	IPY00044488	--	214	Global Call IP (SIP)	When the CGPN and CDPN optional IE's are not present in an incoming setup message, the H323 stack also does not put anything in.
Resolved	IPY00044273	--	214	Global Call IP (SIP)	The application cannot issue a re-INVITE from audio to T.38 fax using the **gc_ReqModifyCall( )** function in T38_MANUAL_MODIFY_MODE.
Resolved	IPY00044219	--	214	Global Call IP (SIP)	Manual transfer ends with a disconnected event.
Resolved	IPY00044516	--	214	IP Media	Audio quality issues occur after implementing native hairpinning.
Resolved	IPY00044145	--	214	IP Media	False QoS alarms of RTPTIMEOUT are sent to the application when SSRC switching occurs from same endpoint.
Resolved	IPY00044101	--	214	IP Media	Setting unFramesPerPacket to zero with G729 causes a Blue Screen of Death (BSOD).
Resolved	IPY00043787	--	214	IP Media	The ec_stream hangs when using FILE_FORMAT_WAVE.
Resolved	IPY00043716	--	210	CSP	ec_stream timers confusion occurs when the MaxSil timer is applied to stopping the **ec_reciottdata( )** function.
Resolved	IPY00044310	--	210	Dialogic® HMP software	Incomplete receipt of DTMF after the system has operated for a period of time.
Resolved	IPY00043860	--	210	Dialogic® HMP software	When Dialogic® HMP is connected to the Avaya switch, the switch sends an unexpected empty Terminal Capabilities Set (TCS) and an Open Logical Channel (OLC) in the middle of the call. As a result, the state machine does not handle the signaling connect event in the media.
Resolved	IPY00043399	--	210	Global Call IP (SIP)	When the Session Timers (MinSE and Session-Expires) were set to zero, the request and response messages contained the "timer" parameter in the Supported header and Min-SE:0.
Resolved	IPY00043595	--	202	Dialogic® HMP software	A change in the SSRC number in the middle of an RFC 2833 transmission results in the software corrupting outgoing audio.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00043209	--	202	Dialogic® HMP software	A Blue Screen of Death (BSOD) occurs when Dialogic® Services are stopped.
Resolved	IPY00043543	--	202	Voice	Data stream corruption results when the Dialogic® HMP Interface board rate interrupts are missing.
Resolved	IPY00043149	--	202	Voice	An RTF log is not generated for the **dx_setchxfercnt( )** function.
Resolved	IPY00043536	--	201	Dialogic® HMP software	An attempt to stop the device using the "Safely Remove Hardware" window fails.
Resolved	IPY00043210	--	201	IP Media	If there are several hairpinned calls in progress and the application asserts and shuts down, those hairpinned connections are still in progress even when the application is restarted.
Resolved	IPY00043620	--	201	Voice	The **dx_stopch( )** function does not return or progress.
Resolved	IPY00043320	--	201	Voice	A kernel access violation causes TEC_STREAM to be lost.
Resolved	IPY00042993	--	199	CSP	The **ec_getblkinfo( )** returns incorrect info when the stream buffer size is 4096 bytes or larger.
Resolved	IPY00043261	--	199	Global Call IP (SIP)	In 1PCC, a HOLD re-INVITE using the **gc_AcceptModifyCall( )** function fails for low bit rate codecs.
Resolved	IPY00043043	--	199	Global Call IP (SIP)	An access violation occurs when trying a SIP re-INVITE.
Resolved	IPY00042960	--	199	Global Call IP (SIP)	SIP Transport method is not included in Contact for 200 OK or 1xx messages.
Resolved	IPY00042671	--	199	Global Call IP (SIP)	Error occur with Dialogic® HMP Software after calling the **gc_InvokeXfer( )** function.
Resolved	IPY00043035	--	199	IP Media	There are no RTCP receiver reports about received RTP.
Resolved	IPY00042654	--	199	Voice	Routing two Dialogic® DTI devices together using the **dx_mreciottdata( )** function only records from one device.
Resolved	IPY00042651	--	199	Voice	The **dx_getparm( )** function is not working correctly for DXCH_XFERBUFSIZE.
Resolved	IPY00042893	--	195	Global Call IP (SIP)	Ingress INVITEs are rejected with a "Requires: 100rel" header.
Resolved	IPY00042742	--	195	Global Call IP (SIP)	The application cannot set PRACK capability in egress INVITE.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00042805	--	195	IP Media	When the **ipm_ModifyMedia( )** function is called with the CODER_TYPE_NATIVE flag specified, the payload type is incorrect on outbound RTP and the content is scrambled.
Resolved	IPY00042235	--	195	Licensing	The **gc_Start( )** function fails when the application is launched as a service and the user is Network Service (non Administrator).
Resolved	IPY00040101	--	192	Dialogic® HMP software	Windows 2003 Server SP2 crashes due to an APICDRV.sys issue.
Resolved	IPY00041686	--	192	Fax	When using manual modify mode, H.323 functionality breaks with regard to transitioning from audiio to fax.
Resolved	IPY00042461	--	192	Global Call IP	A “486 Busy Here” error occurs if all codecs are enabled on Eyebeam.
Resolved	IPY00042737	--	192	Global Call IP (SIP)	GCEV_ALERTING is the last event received after executing a **gc_MakeCall( )**.
Resolved	IPY00042550	--	192	Global Call IP (SIP)	Response messages for an Ingress INVITE without "Supported:100rel" contain a "Supported:100rel" header when PRACK is enabled.
Resolved	IPY00042424	--	192	Global Call IP (SIP)	When the application attempts to make "late media" SIP calls, no SDP is added to the SIP ACK message, and the remote end disconnects the call.
Resolved	IPY00042407	--	192	Global Call IP (SIP)	SIP calls get rejected with a "486 Busy Here” error.
Resolved	IPY00042329	--	192	Global Call IP (SIP)	When programmed in 1PCC GCCAP_DONTCARE, the SIP G.729AB ingress call sets an incorrect RX media codec.
Resolved	IPY00042300	--	192	Global Call IP (SIP)	IPT will not switch to T38 if it rejected a previous audio re-INVITE in 1PCC.
Resolved	IPY00042204	--	192	Global Call IP (SIP)	IPT IP calls appear to 'forget' local media information.
Resolved	IPY00041894	--	192	Global Call IP (SIP)	The TO header from the Dialogic® HMP software is incomplete when responding to a BYE message.
Resolved	IPY00041789	--	192	Global Call IP (SIP)	There is no checking of UPDATE in the Allow Header of the Ingress INVITE/ 200 OK.
Resolved	IPY00041876	--	192	IP Media	H.323 transfer fails with the RTF log ERR1 message, EVENT_CONNECT_SIGNALING failed.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00042246	--	192	MSML	The Dialogic® HMP software is sending a TEL_INFOTYPE_EVENT of 255, which is not a valid digit.
Resolved	IPY00042072	--	182	Configuration	CONFIG files were not updated with increased memory usage after the addition of a multimedia component caused the memory increase during IP calls.
Resolved	IPY00042011	--	182	Configuration	DCM and the DCM system tray has inconsistent system status.
Resolved	IPY00040358	--	182	Dialogic® HMP software	A Blue Screen of Death (BSOD) occurs when rebooting the system while Dialogic® Services are running.
Resolved	IPY00041047	--	182	Fax	Phase D Reply information in not returned in a timely manner when performing a Receive Fax operation.
Resolved	IPY00041014	--	182	Installation	Default CONFIG, FCD and PCD files (qsige1) for the Dialogic® HMP Interface Board are not regenerated during the build or packaging process.
Resolved	IPY00042397	--	182	Multimedia	An MMEV_OPEN event is not returned when opening an mmB1 device using the **mm_Open( )** function.
Resolved	IPY00041847	--	176	Fax	A Blue Screen of Death (BSOD) occurs on a system running heavy amounts of faxing activity.
Resolved	IPY00042016	--	176	Voice	The **ATDX_BUFDIGS( )** function returns an incorrect value.
Resolved	IPY00041112	--	175	Conferencing	The application and drivers do not respond when attempting to change the attributes on a party device that is already in a conference.
Resolved	IPY00041160	--	175	Dialogic® HMP	There are missing digits when using RFC 2833 for DTMF.
Resolved	IPY00041674	--	175	Dialogic® HMP software	When endpoints send Comfort Noise packets and then stop RTP, Dialogic® HMP software interprets this as a jitter event which causes the jitter buffer to expand to try to compensate for latency.
Resolved	IPY00041570	--	175	Fax	The Fax resource causes the driver to lockup.
Resolved	IPY00041582	--	174	Global Call IP (SIP)	Egress SIP calls fail due to order of 'm=' line in SDP.
Resolved	IPY00041300	--	174	Global Call IP (SIP)	SIP calls are rejected with a "486 Busy Here" message.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00041404	--	174	Global Call IP (SIP)	The PRACK response contains an unwanted SDP OFFER.
Resolved	IPY00041294	--	174	Global Call IP (SIP)	The wrong incoming Session Description Protocol (SDP) type is being set by the SIP signal.
Resolved	IPY00041405	--	174	IP Media	When the **gc_Listen( )** function is executes in async mode, an IMPEV_LISTEN event is received instead of a GCEV_LISTEN event.
Resolved	IPY00041801	--	174	IPMedia	A few different exceptions and errors occur in libipm_ipvsc and dm3low when running two applications using different IPM resources simultaneously.
Resolved	IPY00040866	--	172	H323 Call Control	The **gc_InvokeXfer( )** function fails when interfacing with the Avaya IP PBX configured for TCP transport.
Resolved	IPY00041157	--	171	IP Media	The **ipm_Listen( )** function is returning incorrect return codes.
Resolved	IPY00041254	--	171	Voice	Two inits of the same CSP Channel result when two **dx_open( )** calls are made to the same channel.
Resolved	IPY00040029	--	170	Conferencing (DCB)	A memory leak occurs when a conference is established and then deleted quickly and frequently.
Resolved	IPY00041268	--	170	Configuration	Incorrect mutex causes memory corruption.
Resolved	IPY00041076	--	170	Device Management	The **dev_PortConnect( )** function fails while running a hairpin test.
Resolved	IPY00040859	--	165	IPML	An RTP socket binding failure occurs if two customer applications are started/invoked in a certain order.
Resolved	IPY00040630	--	165	MSML	If you use <deleteWhen=never> when creating conferences, then destroy them manually using <destroyconference>, Dialogic® HMP may stop responding to requests after a number of hours.
Resolved	IPY00040826	--	165	Voice	The **dx_playiottdata( )** function does not return an error when playing a .WAV file with an unsupported format. Instead, the voice resource becomes stuck.
Resolved	IPY00040895	--	162	Licensing	Addition of Call Control license change causes the Global Call Basic Call Model demo to fail.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00039536	--	162	Configuration	D-Channel tracing with Dialogic® HMP DNI boards time stamps are not aligned with the system time.
Resolved	IPY00041234	--	162	Global Call IP (SIP)	DNS queries sent by Dialogic® HMP use a destination UDP port of 13568 instead of 53.
Resolved	IPY00040961	--	162	Global Call IP (SIP)	Session Timers Issue. Dialogic® HMP reports a TASK_FAILURE when calling the **gc_Answer( )** function.
Resolved	IPY00040440	--	162	Global Call IP (SIP)	A fall back occurs when DNS resolution fails while making an SIP HMP call.
Resolved	IPY00039823	--	162	Global Call IP (SIP)	When modifying the "ReferTo" header field, the Global Call API only adds content to the end of the SIP header instead of replacing the contents of the pre-existing header field. This results in two "ReferTo" header fields sent in the outgoing SIP REFER message when invoking the call transfer.
Resolved	IPY00040419	--	162	Installation	Optimal logging levels for IPT (and Dialogic® HMP).
Resolved	IPY00040382	--	157	MLM Firmware	An intermittent issue occurs regarding DTMF detection. Related to IPY00040071.
Resolved	IPY00040031	--	157	Conferencing (DCB)	The **dcb_SetParm( )** function disables Active Talker Detection on only the first conference.
Resolved	IPY00040082	--	157	Configuration	When stopping Dialogic® Services via DCM, the control panel indicates stop a few seconds before DCM completes the task.
Resolved	IPY00040071	--	157	MxML	An intermittent issue occurs regarding DTMF detection. Related to IPY00040382.
Resolved	IPY00040038	--	155	Global Call	Two TASKFAILS result when a BUSY is returned during a blind transfer in response to switch NOTIFY messages.
Resolved	IPY00039965	--	155	Global Call (IP)	An outbound IP call fails with "IPEC_SIPReasonStatus503ServiceUnavailable" when the hostname is passed for the destination address in the dialstring.
Resolved	IPY00040033	--	155	Global Call IP (SIP)	For an incoming re-INVITE, a “488” response is not sent automatically when re-INVITE support is disabled.
Resolved	IPY00039822	--	155	Global Call IP (SIP)	The PRACK response contains an unwanted SDP offer.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00039707	--	155	Global Call IP (SIP)	A "glare" condition resulting in a disconnected call occurs during an automatic SIP re-INVITE when media switches from audio to fax.
Resolved	IPY00039325	--	155	Global Call IP (SIP)	In a two party SIP call (1PCC or 3PCC), Dialogic® HMP sends a 'blank' RTP when not connected to the CT bus.
Resolved	IPY00040365	--	155	IPML	The system freezes when calling the **ipm_Listen( )** function to listen to voice device after an MMEV_PLAY completion event. This same call will succeed if called after an MMEV_PLAY_ACK event.
Resolved	IPY00039985	--	155	IPML	Although the QOS fault threshold range is 50 - 1200 (x 100ms), the application can set the fault threshold to a range of 1 - 1200 (1000ms).
Resolved	IPY00039964	--	155	Voice	The Recorder component has DTMF back suppression parameters in millisecond units, while the Encoder uses sample units.
Resolved	IPY00039701	--	153	Configuration	Toggling the Dialogic® Service start up state between Manual and Semi-Automatic mode also toggles the board download state provided by **NCM_GetSystemState( )**.
Resolved	IPY00037336	--	153	Configuration	The dm3nk INFO and ADVISORY messages posted in the event log cause confusion.
Resolved	IPY00039308	--	153	Dialogic® HMP	The application experiences one-way audio if the endpoint sends a different SRTP “ssrc” to the same Dialogic® HMP port.
Resolved	IPY00039855	--	153	Fax	Implement better handling of the invalid Data Modulation Bits from a DIS frame.
Resolved	IPY00039874	--	153	Voice	A TDX_VAD event is not generated to the application when A-law code is used for an RTP connection.
Resolved	IPY00037547	--	152	Configuration	Dialogic® Services do not start if the applications uses the maximum number of resources in the license configuration on a dual processor, dual core, hyper-threading, 3.8 GHz system, with 2 GB RAM memory.
Resolved	IPY00039675	--	152	Fax	A system crash occurs in the *ssp.mlm.sys* file.
Resolved	IPY00039649	--	152	Fax	The application fails to perform an **fx_listen( )** on V.17 fax sessions.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00039505	--	152	Fax	A request to switch to T.38 fax is rejected, but a gcev_taskfail event is not received.
Resolved	IPY00039639	--	152	Global Call	The **gc_SetAlarmParm( )** function causes an assert in *msvcrt.dll*.
Resolved	IPY00039446	--	150	Conferencing (DCB)	The **dcb_unmonconf( )** function unmonitors the wrong conferee (party). Instead of unmonitoring the MONITOR conferee (party), the most recent added party is being UNMONITORED.
Resolved	IPY00039451	--	150	Global Call	The gc_AcceptModifyCall( ) function fails to return a termination event.
Resolved	IPY00039179	--	150	Global Call	When the SYNC function **gc_ReleaseCall( )** is called during a glare scenario, an unexpected GCEV_RELEASECALL event is returned.
Resolved	IPY00037841	--	150	Global Call	A deadlock occurs between an application thread and the DM3 worker thread if the application issues **gc_GetLineDevState( )** from one thread before the **gc_OpenEx( )** function returns from another thread.
Resolved	IPY00039564	--	150	Global Call IP (SIP)	A GCEV_EXTENSION event (IPSET_SWITCH_CODEC, IPPARM_READY) is not received for T.38 call.
Resolved	IPY00039401	--	150	Global Call IP (SIP)	The "Record-Route" field of the SIP header message gets incorrectly reported as the "Route" field in an incoming SIP message when using the Global Call API.
Resolved	IPY00039333	--	150	Global Call IP (SIP)	The SIP INVITE fast-start codec G.729A should default to annex B.
Resolved	IPY00039315	--	150	Global Call IP (SIP)	An Access Violation (0xC0000005) occurs in *libsipsigal.dll* when receiving a SIP REFER message containing a header field parameter with content greater than 255 bytes.
Resolved	IPY00039225	--	150	Global Call IP (SIP)	G.729ab appears in the SIP 200 OK SDP instead of the G.729 codec.
Resolved	IPY00039488	--	150	Installation	The one port IP call control verification license per marketing requirements was missing. SU 150 supports the new default license file name, *1r1v0e0c0s0f1i_ver.lic*, for Dialogic® HMP Software Release 3.0 only.
Resolved	IPY00038998	--	150	PSTN Call Control	Bipolar violation alarms are reported in the LineAdmin, but not reported to either the DTI API program or GCAMS program.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00039492	--	150	Runtime Tracing (RTF)	A memory leak is occurring,
Resolved	IPY00039455		150	Voice	Continuous TDX_HIGHWATER events occur, causing the library to stop consuming data from the application (through the **dx_PutStreamData( )** function).
Resolved	IPY00039321	--	144	Fax	The preamble DIS frame does not respond during a fax.
Resolved	IPY00039232	--	144	Global Call	The application does not receive a GCEV_ANSWERCALL event.
Resolved	IPY00039206	--	142	Fax	The **fx_sendfax( )** function fails with a TFX_FAXERROR response.
Resolved	IPY00039033	--	142	Fax	The **fx_sendfax( )** function hangs in PHASE D.
Resolved	IPY00039036	--	142	Global Call IP (SIP)	The application returns a "486 Busy Here" response when closing and reopening a device in 3PCC mode.
Resolved	IPY00038802	--	142	MSML	MSML INFO message is rejected by "486 Busy Here" after receiving 200 OK on previous INFO transaction.
Resolved	IPY00039170	--	142	Voice	The **dx_SetWaterMark( )** function ignores the values indicated by the application.
Resolved	IPY00038869	--	137	Configuration	Dialogic® HMP fails to start when Non Linear Processing (NLP) is defined in the configuration file.
Resolved	IPY00038997	--	137	Dialogic® HMP	RFC2833 digits are not detected when RFC2833 packets have a volume of 18 and start the first packet with duration value of 240.
Resolved	IPY00038856	--	137	Fax	A Fax resource error occurs after applying a patch for IPY00038205.
Resolved	IPY00037654	--	137	Fax	After a successful call to **fx_rcvfax( )** in T.30 mode, the entire received TIFF file gets the wrong image contents when ECM is used and retransmission occurs.
Resolved	IPY00038848	--	137	Global Call	When **gc_DropCall( )** returns a cause code of 5801 from the application, an invalid response code appears in the RTF log.
Resolved	IPY00038992	--	137	Global Call IP (SIP)	The application is missing a termination event after executing the **gc_InvokeXfer( )** function followed by a disconnect from the receiving end.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00038868	--	137	Global Call IP (SIP)	A GCEV_CONNECTED event is not returned from outgoing calls using SIP.
Resolved	IPY00038827	--	137	Global Call IP (SIP)	The **gc_AnswerCall( )** function does not receive a response when a SIP CANCEL is received after a SIP INVITE.
Resolved	IPY00038208	--	134	Global Call	When an incoming ISDN PROGRESS message contains a cause IE, the Call Control library does not forward it along with the GCEV_PROGRESSING event, or clear it. Instead this cause value is associated with the GCEV_DISCONNECTED event.
Resolved	IPY00038696	--	134	Global Call IP	Improper sequence numbers are generated when sequence number index handling in srtpGetIndex fails to initialize on the first packet.
Resolved	IPY00038732	--	134	Global Call IP (SIP)	Dialogic® HMP does not choose the bit rate specified by the application if the remote codec fails to specify the bit rate.
Resolved	IPY00038288	--	134	Media	An attempt is made to free a NULL pointer after a memory allocation failure.
Resolved	IPY00038747	--	134	Voice	The channel hangs when calling the **dx_AdjSv( )** function while play is in progress.
Resolved	IPY00038218	--	133	Device Management	An exception occurs in the Device Management library when the **dev_SetError( )** function keeps data in local thread storage.
Resolved	IPY00038513	--	133	Dialogic® HMP	UIO offset does not account for data already played from a previous buffer.
Resolved	IPY00038313	--	133	Fax	The Fax application does not return any event when the channel hangs after calling the **fx_sendfax( )** function.
Resolved	IPY00038610	--	133	Global Call IP (SIP)	A simultaneous re-INVITE causes a GCEV_TASKFAIL event to be generated after calling the **gc_ReqModifyCall( )**.
Resolved	IPY00038580	--	133	Global Call IP (SIP)	An access violation occurs in *libsipsigal.dll*.
Resolved	IPY00038572	--	133	Global Call IP (SIP)	Unable to determine the IPPARM_MGSTYPE value for an incoming SIP message.
Resolved	IPY00038555	--	133	Multimedia	The **mm_ErrorInfo( )** function output is incorrect.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00038638	--	133	Voice	A TDX_PLAY event is generated prematurely when using the **dx_PutStreamData( )** function to populate the play buffer.
Resolved	IPY00038475	--	133	Voice	The **dx_dial( )** function does not always a return a TDX_CALLP event when stopped with the **dx_stopch( )** function.
Resolved	IPY00038474	--	133	Voice	The voice resource gets stuck in a busy state when a TDX_CALLP event is not returned from the **dx_dial( )** function.
Resolved	IPY00038549	--	133	Voice	Ports are stuck in a CS_STOPD state when a caller hangs up during playback.
Resolved	IPY00038470	--	130	Global Call IP	A GCEV_CONNECTED event is missing from the RTF log.
Resolved	IPY00038365	--	130	Global Call IP (SIP)	Egress SIP calls work briefly, then omit SDP (G711A codecs) in egress INVITE messages.
Resolved	IPY00038343	--	130	Global Call IP (SIP)	When creating registrations, the customer runs into a limit and gets an error unless he unregisters a previous user.
Resolved	IPY00038342	--	130	Global Call IP (SIP)	Simultaneous disconnects cause an error at RTL after a number of calls.
Resolved	IPY00038240	--	130	Global Call IP (SIP)	A thread in the API is asserting when an inbound re-INVITE is received before GCEV_REQ_MODIFY_CALL.
Resolved	IPY00038476	--	130	IPML	Benchmarking on Dialogic® HMP Software Release 3.0 and conferencing experiences latency in adding and removing conferees as the number of conferences increases.
Resolved	IPY00037789	--	130	Runtime Tracing	RTF logs are not generated if the application is executed as a service and the user is "Network Services."
Resolved	IPY00038217	--	128	Conferencing (DCB)	The **dcb_setcde( )** function fails with an "invalid board" error.
Resolved	IPY00038049	--	128	Conferencing (DCB)	The **dcb_estcnf( )** function fails when using Dialogic® DNI boards. This does not occur when using the dcb API with IP or when using the CNF API either over IP or with Dialogic® DNI boards.
Resolved	IPY00038233	--	128	Dialogic® HMP	A quality issue relating to poor audio on the RTP connection occurs intermittently.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00037756	--	128	Fax	Dialogic® HMP fails to deliver a termination event for “send fax” when the **fx_stopch( )** function is called randomly
Resolved	IPY00037659	--	128	Firmware	The thin blade access kernel routines cause a black screen in certain situations.
Resolved	IPY00037918	--	128	Global Call	The RSI link goes down intermittently.
Resolved	IPY00038150	--	128	Global Call IP	A simultaneous re-INVITE from both sides of a call causes a TASKFAIL for the **gc_ReqModifyCall( )** function.
Resolved	IPY00038060	--	128	Global Call IP (SIP)	A rvSdpStringGetData crash occurs when there are no media attributes containing "rtpmap" in a SIP INVITE.
Resolved	IPY00033127	--	128	Installation	The application is unable to get an unblocked event while using the PDK_US_LS_FXS_IO protocol.
Resolved	IPY00038092	--	128	IPML	The **ipm_GetCapabilities( )** function returns a Coder Type of 12, which is not a valid value for eIPM_CODER_TYPE.
Resolved	IPY00038050	--	128	IPML	The HMP Coder table listed coders twice, causing **ipm_GetCapabilities( )** to return an incorrect value.
Resolved	IPY00037888	--	123	Device Management	The **dev_ErrorInfo( )** function does not return EDEV_INVALIDDEVICEHANDLE when an invalid device handle is passed to the **dev_Connect( )** function.
Resolved	IPY00037704	--	123	Device Management	The **dev_Connect( )** function does not return immediately when running in asynchronous mode.
Resolved	IPY00037358	--	123	Dialogic® HMP	The receipt of RFC 2833 digits after the G.711 Comfort Noise packet causes Dialogic® HMP to double detect the first RFC 2833 digit incorrectly.
Resolved	IPY00035675	--	123	Dialogic® HMP	Server locks up (freezes) during CAS Bulk Call test under Dialogic® HMP 2.0 Software Release using QUAD span.
Resolved	IPY00037795	--	123	Global Call	In the RTF logs, gc_h3r reports a loss of buffers for the Ext Evt Pool, after which almost all **gc_MakeCall( )** function calls return a failure.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00037422	--	123	Global Call	When using Dialogic® DNI boards running ISDN and encountering frame slips on the line, calls to the **gc_SetConfigData( )** function to change line properties either hang completely or return after an extremely long time.
Resolved	IPY00034254	--	123	Global Call	The **gc_SetConfigData( )** function hangs when changing line encoding/framing at runtime.
Resolved	IPY00037699	--	123	Global Call IP (H.323)	The **gc_Stop( )** function fails to return control to the application when called after the LAN cable is disconnected.
Resolved	IPY00037778	--	123	Global Call IP (SIP)	The Session Description Protocol (SDP) is missing from SIP re-INVITE after calling the **gc_ReqModifyCall( )** function.
Resolved	IPY00037737	--	123	Global Call IP (SIP)	The **gc_MakeCall( )** function returns a GCEV_TASKFAIL event when frames per packet are set to 20.
Resolved	IPY00036779	--	123	Global Call IP (SIP)	Assertion faults are received when “DON’T CARE” is used with the G.726 coder.
Resolved	IPY00037733	--	123	Host Interface	A switch is responding to outbound calls with a DISCONNECT and cause code 0x19 (== 25). This cause code is not defined in the Dialogic® HMP header file.
Resolved	IPY00037776	--	123	PSTN Call Control	When using a Quad Span thin blade, ISDN firmware running NI2 quickly disconnects a normal outbound call that has progressed through the ALERTING/PROCEEDING/PROGRESSING states.
Resolved	IPY00037777	--	123	Voice	The callback function fails after calling the **dx_stopch( )** function to stop **dx_playiottdata( )**.
Resolved	IPY00036634	--	117	Configuration	Cannot open a board device on a thin blade twice.
Resolved	IPY00037708	--	117	Diagnostics	The `its_sysinfo` tool, a standalone utility used to collect system data, is not collecting a full memory dump.
Resolved	IPY00037632	--	117	Global Call	A delay in the SS7 service picking up messages from the IPC queue results in an error message, ERROR_IO_PENDING, and the SS7 library terminates the IPC.
Resolved	IPY00037686	--	117	SIP	A "486 Busy Here" message is received on various occasions.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains “--” and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00037746	--	117	Voice	An exception results when calling ATDX_CPERROR with RTF logging enabled.
Resolved	IPY00037655	--	115	Fax	In T.30 mode, the **fx_rcvfax( )** function does not terminate when there is a busy tone.
Resolved	IPY00037656	--	115	Host Interface	The DM3 time slot crashes.
Resolved	IPY00037603	--	115	Install	During System Release installation, the SNMP component gets installed regardless of whether or not it was selected as an install option.
Resolved	IPY00037633	--	115	PSTN Call Control	Call transfer fails when using protocol pdk_sw_e1_ssls_io.
Resolved	IPY00037541	--	115	SIP	When a GCEV_REQ_MODIFY_CALL event is returned, RTP port-address info in the event data stays unchanged until the **gc_AcceptModifyCall( )** function is called and a GCEV_ACCEPT_MODIFY_CALL event is received.
Resolved	IPY00037532	--	115	SNMP	System error on Dialogic® HMP from *CosNaming405.dll*, related to SNMP.
Resolved	IPY00037432	--	115	Voice	The **dx_clrdigbuf( )** function overwrites the area of threads stack causing the application to crash.
Resolved	IPY00036658	--	112	Global Call	The **gc_GetMetaEvent( )** function is failing with an Internal (SRL) failure. Assert in libgch3r.
Resolved	IPY00037618	--	112	Global Call ISDN	The header file is missing the GCIS_PARM_CODINGSTANDARD and GCIS_PARM_TRANSFERCAP ISDN parameters.
Resolved	IPY00037481	--	112	MSML	A problem exists with the **<unjoin>** command on the Dialogic® HMP media server.
Resolved	IPY00037338	--	112	SIP	A CANCEL message from the **gc_MakeCall( )** function timeout does not contain the SIP header information.
Resolved	IPY00037222	--	112	SIP	Dialogic® HMP does not return a GCEV_DROPCALL event until 32 seconds after calling the **gc_DropCall( )** function. This occurs when both sides of the call execute **gc_DropCall( )** at the same time.
Resolved	IPY00036943	--	110	Conferencing (CNF and DCB)	Clicks are heard when a party is removed from the conference before the **dev_Disconnect( )** function can be called.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00037170	--	110	Host Interface	An App exception occurs if an incorrect handle is passed to the **dev_Disconnect( )** function. [Observed on SU 80]
Resolved	IPY00036292	--	110	Host Interface	Issuing the **dx_stopch( )** function with the mode parameter set to EV_NOSTOP does not return a TDX_NOSTOP event when called after the **dx_getdig( )** function.
Resolved	IPY00036930	--	110	IP Media	There is an excess byte (trailing '\0') put in the NonStandardData field of the Q931 Facility message sent by the **gc_Extension( )** function. Changed non-standard data handling in the library. IPSET_NONSTANDARDDATA / IPPARM_NONSTANDARDDATA_DATA specified by application is now being handled as binary data. This allows the application to send byte 0x00 within IPSET_NONSTANDARDDATA / IPPARM_NONSTANDARDDATA_DATA.
Resolved	IPY00036867	--	110	Multimedia	There is a missing completion event for the **mm_play( )** function when audio is rerouted.
Resolved	IPY00033753	--	110	RTP	Dialogic® HMP Software Release has the same Blue Screen of Death (BSOD) as seen on the SU 37 servers.
Resolved	IPY00037361	--	110	SIP	The application receives a "486 Busy Here" message on various occasions.
Resolved	IPY00037386	--	110	Voice	There is always size discrepancy between the size specified in the iott.io_length and the file size created in the **dx_reciottdata( )** function.
Resolved	IPY00037225	--	110	Voice	Issuing the **dx_stopch( )** function with the mode parameter set to EV_NOSTOP on an idle channel does not return a TDX_NOSTOP event.
Resolved	IPY00036086	--	103	Installation	Alternate method to silent uninstall Dialogic® HMP Software Release fails to remove two components.
Resolved	IPY00037181	--	103	Integration Project	The following files are missing: *pdk_sw_e1_ntmd_io.arm.hot* *pdk_sw_e1_ntmd_io.hot*.
Resolved	IPY00036515	--	103	Integration Project	AN XML tag is missing from responses to msml commands.
Resolved	IPY00036209	--	103	Integration Project	Orbacus services fail on startup when using certain scenarios.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00035685	--	103	Integration Project	Memory leak in **NCM_GetFamilyDeviceByAUID( )**.
Resolved	IPY00035405	--	103	IP	Any delays or failures in obtaining an IP address by the Windows DHCP service on bootup could cause the DlgPnPObserverService to fail in the detection phase. This failure makes it impossible to start Dialogic® Services until a reboot is performed and DHCP executes properly.
Resolved	IPY00036790	--	103	IP Media	A blue screen results when testing Dialogic® HMP Software Release 2.0 SU 125.
Resolved	IPY00035821	--	103	IP Media	**ipm_ResetQoSAlarmStatus( )** fails to reset QoS alarms, resulting in the same device not being able to receive the same alarm again.
Resolved	IPY00035767	--	103	IP Media	The jitter buffer parameter number changed from Dialogic® HMP Software Release 2.0 to Dialogic® HMP Software Release 3.0.
Resolved	IPY00036914	--	103	MSML	Problems result when collecting RFC 2833 DTMF digits using Dialogic® HMP Software Release 3.0 SU 87 and SU 91.
Resolved	IPY00036618	--	103	SIP	Dialogic® HMP rejects Register messages after several days of taking calls.
Resolved	IPY00036332	--	99	CNF	Added beep tone function on conferee entry or exit.
Resolved	IPY00036283	--	99	FAX	Fix HMPFax firmware to properly handle T.30 RTN ("Retrain Negative" response).
Resolved	IPY00034315	--	99	FAX	Time display in fax header of AM or PM is wrong for certain time period.
Resolved	IPY00035374	--	99	Licensing	**gc_start( )** fails under Dialogic® HMP Software Release 3.0 Build 78 with Dialogic® DNI601TEPHMP.
Resolved	IPY00036075	--	99	Voice	If a user attempts to play forever or until EO is reached (specifying io_length = -1) with UIO plays on DM3/HMP, there is still a hard upper limit on the number of bytes that can be played, which is approximately equal to 2.147GB (~2 to the 31 bytes).
Resolved	IPY00036312	--	97	Global Call	**gc_ReqModifyCall( )** receives a GCEV_MODIFY_CALL_REJ and then various GCEV_TASKFAILs occur. The system application does not recover at that point and an application restart is necessary.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00036451	--	97	IP Media	**ipm_ModifyMedia( )** fails to change the direction of the media stream, tested on Dialogic® HMP Software Release 3.0 SU 81.
Resolved	IPY00036236	--	97	IP Media	**gc_MakeCall( )**fails with GCEV_TSAKFAIL. Error occurs when RTL calls **ipm_StartMedia( )** during call establishment. It happens under 96 channel load after a couple of hours of execution.
Resolved	IPY00035843	--	91	IP Media	**ipm_ModifyMedia( )** fails to change the direction of the media stream, tested on Dialogic® HMP Software Release 3.0 SU 81.
Resolved	IPY00035795	--	90	Conferencing (CNF)	Sporadically, CNFEV_ADD_PARTY arrives with the wrong party handle on multi-core or hyper-threaded systems.
Resolved	IPY00035507	--	90	Conferencing (CNF)	CNF API does not provide a way to recover CNF based devices used for ongoing conferences after application is terminated due to abnormal exit. See **cnf_ResetDevices( )** function in Section 3.4.4, "Dialogic® Conferencing API Library Reference", on page 368.
Resolved	IPY00035822	--	90	Global Call IP	HMP 2.0 and 1.3 do not respond to SIP 407 Proxy Authentication Required messages.
Resolved	IPY00034679	--	87	Global Call IP	SIP Call Control-fax transmission fails when using new xxxModifyCall Method of handling re-INVITE.
Resolved	IPY00035760	--	87	Multimedia	Application crashes when nonexistent video file is added into MM_PLAY_INFO.
Resolved	IPY00035371	--	85	Configuration	In Dialogic® HMP Software Release 3.0 Build 78, DCM cannot start if Dialogic® DNI601 is configured as E1CC.
Resolved	IPY00034049	--	85	Installation	License - Application fault occurs at system startup after Dialogic® HMP is installed.
Resolved	IPY00035589	--	85	SNMP	dlgagent.log file increasing in size as *snmp.exe* writes same entry in this file every 10 seconds: INVALID_HANDLE_VALUE - 2.
Resolved	IPY00035584	--	85	Voice	Play of multiple wave files from a DX_IOTT data structure on Dialogic® HMP Software Release 3.0 fails.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00035489	--	81	Conferencing (CNF)	Access violation when using **cnf_Open( )** to open invalid board. See **cnf_OpenEx( )** function in Section 3.4.4, "Dialogic® Conferencing API Library Reference", on page 368.
Resolved	IPY00033894	--	81	Conferencing (CNF and DCB)	There is a limit of 254 parties per conference. If this limit is exceeded, no error message is returned and the application may fail.
Resolved	IPY00034064	--	81	Dialogic® HMP	The warning message "G729 decoder: Warning unknown frame type 7" may appear when using G.729 coders to interoperate with the T1/E1 IP Media Gateway.
Resolved	IPY00034435	--	81	FAX	V.17 fax call fails after a successful T.38 fax call on the same fax resource.
Known	IPY00006325	36540		Licensing	If the license file is located on a network drive, the licensing service may not start correctly. Recommend that license file be stored on local system.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description

Known	IPY00100540	--		Board Download	Intermittent download failures are observed on systems with more than one DNI601TEPHMP boards. The Dialogic System Service might fail upon system reboot; and the System Event viewer shows an entry similar to the below in the System logs:
					
					`General fault: MC_ERROR_CODE_MOVERS: An error occurred while moving the Kernel Image to the board.`
					`Board Number: 1`
					`Instance: 0`
					`Additional Data: Image Name = (null) ,CmdLine = C:\Program Files (x86)\Dialogic\HMP\bin\srecload_16.exe, ErrorCode = 0`
					The board number and instance might be different depending on your configuration and your specific Windows operating system version.
					Workaround: Restore the board's DCM parameter settings to default values. From the DCM Main Window, right click on the HMP_Software device from the DCM Device submenu and select **Restore Device Defaults**. Refer to the *Dialogic® Host Media Processing Configuration Guide* and the *Dialogic® Host Media Processing Administration Guide* for more information.

Known	IPY00092500	--		Conferencing	Background noise is sometimes heard during conferences when using the Coach/Pupil feature and parties from D/4PCIUFEQ or D/4PCIU4SEQ boards.
Known	IPY00006356	36412		Conferencing	If the conferencing application is abnormally stopped during conferencing, HMP service may fail to start because of not being stopped correctly. None
Known	IPY00006121	36098		Conferencing	The **ipm_getxmitslot( )** and **dcb_addtoconf( )** functions will timeout on 400 channels. Use no more than 254 parties per conference on a single conference.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known (permanent)	--	--		Conferencing	The **dcb_setbrdparm( )** function fails when attempting to set MSG_ALGORITHM to ALGO_LOUD and ATDV_ERRMSGP returns "Bad global parameter value". Do not set this parameter. By default, the algorithm uses the Loudest Talker.
Known	IPY00032436	--		Conferencing (CNF)	The active talker list is not empty when no party is opened. None
Known	IPY00034363	--		CSP	A CSP application may miss TEC_STREAM events when full RTF logging for the "Cheetah" HMP library component is enabled. Reduce RTF logging to default level.
Known	IPY00006156	36002		CSP	When using CSP, the **ec_stopch( )** function may hang in some streaming to board tests. None
Known (permanent)	IPY00006236	35116		DCM	Pop-up messages occur when the Country tab is selected in DCM. Do not select the Country tab. It is not supported nor required.
Known (permanent)	--	--		Demo	Demos are intended to be run using a maximum of 4 channels.
Known	IPY00006332	36534		Demos	The following demos may not compile if moved from the default directory location: conferencing ipmediaserver sitest speechprocessing Leave these demo files in the default location following installation of the HMP software.
Known	IPY00005961	31271		Device Management	If an error occurs when executing a Device Management API library function, a value of -1 is returned. When SRL function **ATDV_LASTERR( )** is called, it returns 0 to indicate no error. This is not a problem if an error occurs within a subsystem (e.g., IP Media library) as **ATDV_LASTERR( )** will return the correct error code.
Known	IPY00006055	36034		Diagnostics	Some features of the Listboards utility do not work. None

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00033614	--		Dialogic® DNI Board	When performing **gc_MakeCall( )** on both sides with symmetrical flag set, you may incorrectly get an OFFERED event on both sides. Do not set symmetrical flag.
Known	IPY00033382	--		Dialogic® DNI Board	When using R2MF protocols, calls may sometimes disconnect with a reason "Event caused by protocol error." None
Known	IPY00033243	--		Dialogic® DNI Board	An application may not receive the TDX_CST event for DE_DIGITS after adding a user defined tone with **dx_addtone( )** and associating the tone with an optional digit. This occurs with both Dialogic® HMP provided tone resources and with the tone resources provided by the board. None
Known	IPY00006295	35985		Dialogic® DNI Board	When the d-channel comes back up and channels try to make calls at the same time, only the first channel will correctly connect. Allow a few seconds of delay before making calls.
Known	IPY00006252	35312		Dialogic® DNI Board	When the logical ID of the Dialogic® HMP board is set to a high number, the download may fail. Only use logical IDs less than 255.
Resolved	IPY00028507	36610		Dialogic® DSI Board	Dialogic® DSI single board start/stop may not work consistently on some systems.
Known	IPY00006375	36608		Dialogic® DSI Board	At startup, multiple Dialogic® DSI boards may be assigned the same logical ID in DCM. Using DCM, modify the logical IDs of the Dialogic® DSI boards such that each board has a unique logical ID.
Known	IPY00006374	36607		Dialogic® DSI Board	Removal of a Dialogic® DSI board after installation may result in subsequent startup failures. Reboot the system.
Known	IPY00006367 (IPY00011030, IPY00006366)	36591 (36590, 36589)		Dialogic® DSI Board	Starting a Dialogic® DSI board as the clock slave may occasionally result in poor voice quality. Restart HMP service.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00006338	36546		Dialogic® DSI Board	Starting a previously stopped clock master, which takes over as the master (due to clock priority precedence) does not work properly. Do not change the default clock priority (6) for Dialogic® DSI Boards. Additionally, it is recommended that a Dialogic® DNI board be selected as the clock master.
Known	IPY00006304	36175		Dialogic® DSI Board	The *libMeaSiapi.dll* will fail when **si_Open( )** and **si_Close( )** are called concurrently from multiple threads on the same station in the same process. This failure can occur in more than one context (UMBC callback and SI station thread). Open all SI stations at the beginning of the application and close them before exiting the application.
Known	IPY00006215	36611		Dialogic® DSI Board	System densities of 128 DSI ports are not supported. Limit the total number of DSI ports to 48.
Known	IPY00037739	--		Dialogic® HMP	Message audio is not always played.
Known	IPY00034444	--		Dialogic® HMP	When attempting to start Dialogic® HMP with a third party call control license that contains no RTP resources (for example, 0r0v0e0c0s0f1500i0m_host_pur.lic), startup fails with a general error message and no information about the cause. Activate the third party call control license but do not start Dialogic® HMP.
Known	IPY00034389	--		Dialogic® HMP	If the "Cheetah" HMP library component has errors, it may fill up the default size RTF logging files. Increase the RTF file size to greater than the default.
Known	IPY00034388	--		Dialogic® HMP	The **dx_getfeaturelist( )** function may not accurately return all supported features and occasionally returns unsupported features. None

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known (Permanent)	IPY00033977	--		Dialogic® HMP	At higher channel densities (above 400 channels) using G.711, lower voice quality may be observed, possibly due to packet loss at the network interface card (NIC). To improve voice quality, you can adjust performance parameters for the NIC. These parameters can usually be changed on the Properties page in the Device Manager. Before making these changes, you may need to install the latest drivers for the NIC. To get to your network interface card's Properties page, right-click on My Computer and go to **Manage**. The Computer Management screen is displayed. In the left pane under System Tools, click on **Device Manager**. Device Manager is displayed in the right pane. In Device Manager, expand **Network adapters**. Right-click on your network adapter and click **Properties**. For example, on the **Properties** page for an Intel PRO/1000 MT Desktop Adapter, the Advanced tab includes a selection for **Performance Options**. Under this option, increase the Receive Descriptors and Transmit Descriptors to 1024. Increasing these values can help performance, but will use more system memory.
Known	IPY00033893	--		Dialogic® HMP	If, for some reason, the pdk portion (cas/r2mf) of HMP startup fails, HMP will still appear to start successfully even though ISDN calls will fail. None
Known (permanent	IPY00028239	32740		Dialogic® HMP	After several cycles of starting and stopping the Dialogic System Service, or a single device (HMP), the Dialogic System Service may hang if any Debug View client (DBMON or Debug View) is active.
Known (permanent)	IPY00010888	36193		Dialogic® HMP	Operating Dialogic® HMP with Black Ice* installed will result in degraded system performance (e.g. voice quality). Do not operate Dialogic® HMP with Black Ice* on the system.
Known (permanent)	IPY00010577	36051		Dialogic® HMP	Some machines may briefly freeze during download before completion.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00006336	36435		Dialogic® HMP	If the autostart option is changed while HMP service is running, a failure may result when HMP service is stopped. Do not change the autostart option while HMP service is running.
Known	IPY00006256	35630		Dialogic® HMP	Stopping the board through the Safely Remove Hardware Window fails. Do not use the Safely Remove Hardware Window.
Known	IPY00006178 (IPY00005995, IPY00006012, IPY00006039, IPY00006044, IPY00006085)	35039 (35009, 35004, 35003, 35002, 34958)		Dialogic® HMP	If any change is made to the system's IP address after HMP installation (e.g., disconnecting network cables), HMP will fail to download unless a system reboot is performed. Reboot the system after any post-HMP installation IP address change.
Known	IPY00006153	35033		Dialogic® HMP	Occasionally, during repetitive downloads of 240 channel configurations, HMP download will fail. Repeat download.
Known	IPY00006082	34176		Dialogic® HMP	If a kill task with error code 0x3901f occurs, the system requires rebooting before HMP can be restarted successfully. Reboot system
Known (permanent)	--	--		Dialogic® HMP HMP	High I/O activity during heavy HMP activity may cause an increase in error rates, such as degraded digit detection and voice quality.
Known (permanent)	--	--		Dialogic® HMP HMP	HMP may not run properly if Advanced Configuration and Power Interface (ACPI) is installed. See the *APIC Timer and HMP* topic in the *Compatibility Notes* section.
Known (permanent)	--	--		Dialogic® HMP HMP	The HMP system does not operate with some Dialogic® board level products, or with Dialogic System Release software installed on the same machine.
Known (permanent)	--	--		Dialogic® HMP HMP	When using Dialogic® HMP, advanced power management features, such as hibernation, are not supported.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00101008	--		Dialogic System Service	In Windows 2008 or Windows 7, the Dialogic System Service might fail to consistently start when set to AutoStart mode on system boot-up (System/Device Autostart from the DCM Settings pull-down menu). Starting services manually afterwards works with no issues. By default the Dialogic System Service is set to manual start-mode, thus this problem would not be exhibited. ***Workaround:*** A workaround is currently in place in SU 338 onwards, by means of forcing the Dialogic System Service in DelayedAutoStart mode. Note that if the end user has a Windows Service which is set to depend on the Dialogic System Service, then this Service must **also** be set to DelayedAutoStart mode in order to completely circumvent the problem. Please refer to the *Dialogic® Host Media Processing Configuration Guide* for DCM-specific information. Documentation about Windows Service DelayedAutoStart mode can be obtained from the Microsoft Windows help.
Known	IPY00006676	30626		Fax	The TO: field in the fax header for polled faxes is empty. None
Known	IPY00006575	34131		Fax	If using Debug View when making T.38 fax calls with a Cisco AS5300, there may be an occasional "QT38[3]: ExecIFPInProcessHSHDLC: unexpected ind: 0" error message. None is required. Ignore this message.
Known	IPY00006546	33789		Fax	If using Debug View when making T.38 fax calls, there may be an occasional "FC3_ReadFrame call FAILED" error message. None required as there is no error. Ignore the message.
Known (permanent)	--	--		Fax	For fax applications, the *srllib.h* header file must appear before *faxlib.h* in the #include directive.
Known	IPY00080175	--		Global Call IP	IPT TaskFailure and onStartMedia failure with error code 503.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known (permanent)	--	--		Global Call IP	G.711 µ-Law and A-Law IP Encoding/Decoding only support 64 Kbps.
Known (permanent)	--	--		Global Call IP	If a call is made to Dialogic® HMP using NetMeeting* from a machine that does not have a sound card, the coder negotiation will fail and the call will be disconnected before a GCEV_ANSWERED event is generated.
Known (permanent)	--	--		Global Call IP	If a call is made to Dialogic® HMP using NetMeeting, the "secure outgoing calls" option must not be selected.
Known (permanent)	--	--		Global Call IP	For T.38 calls only, applications should call **gc_SetUserInfo(IPPARM_T38_CONNECT)** after GCEV_OFFERED, but before the next GC function call. Otherwise, the call will fail and the application will get a GCEV_TASKFAIL event.
Known (permanent)	--	--		Global Call IP	Host applications should always clean up resources before exiting. If the application terminates irregularly (resources are not cleaned up as described below), the Dialogic® HMP devices should be restarted using DCM to stop and then start the devices. If the application is using Global Call for call control, the application should terminate all calls by issuing **gc_DropCall( )** followed by **gc_ReleaseCallEx( )**, or use **gc_ResetLineDev( )**. Any open devices should then be closed using **gc_Close( )**. The Global Call Libraries should then be stopped using **gc_Stop( )** to release reserved resources. All other resource types should be stopped and closed using the appropriate **xx_stop( )** and **xx_close( )** API function. (For conferencing applications, **dcb_delconf( )** should be issued on existing conferences before closing the DCB device handle).
Known (permanent)	--	--		Global Call IP	When running Global Call H.323 applications on Dialogic® HMP with Cisco 2600 Gatekeeper (IOS Version 12.3 (1a)), the Cisco 2600 sends packets to the wrong destination port after switching a voice call to T.38 fax. This can be avoided by enabling tunneling on both sides.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known (permanent)	--	--		Global Call IP	Network conditions may cause UDP packets to be lost, some of which may contain the SIP "100 Trying" or "180 Ringing" non-reliable response messages that correspond to the GCEV_PROCEEDING and GCEV_ALERTING events respectively. Because SIP does not require retransmission of 1xx response messages, these events will not be generated when a UDP packet containing the corresponding 1xx response is lost. Applications should be written to continue processing a call when GCEV_CONNECTED is received, regardless of whether GCEV_PROCEEDING or GCEV_ALERTING was received.
Known (permanent)	--	--		Global Call IP	When making 240 calls to the same destination in burst mode, the application may receive GCEV_DISCONNECTED with error IPEC_Q931TransportError. This occurs when the remote side is not able to keep up with the socket connections (WSAECONNREFUSED 10061). The application should drop and release the call.
Known (permanent)	--	--		Global Call IP	During a SIP call termination, the Global Call application may fail to receive a BYE message from a remote user agent due to excessive network errors. If this occurs on your network with your SIP application, the application should be designed to include detection of RTP timeout alarms. When an RTP timeout alarm is received and the resource has not received a GCEV_DISCONNECTED event, the resource should drop the call and proceed as if it had received the GCEV_DISCONNECTED event.
Known (permanent)	--	--		Global Call IP	In SIP Digest Authentication, Global Call only responds to one WWW-Authenticate or Proxy-Authenticate header (if there are multiple) in a 401 or 407 response.
Known (permanent)	IPY00036934	--		Global Call IP (SIP)	Dialogic® HMP starts RTP after the reception of ACK to 200 OK. This may cause the ICMP error "Destination Unreachable" at the remote endpoint if the endpoint starts transmitting RTP immediately after receiving 200 OK. This issue occurs intermittently and only with a limited set of endpoints.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00034051	--		Installation	Silent install cannot find the iss file when using command lines setup /s /f1"setup.iss" or setup /s /f1"hmp.iss" Always specify the full path to the iss file even if the file is in the root directory of the build media or the %winddir%.
Known	IPY00006040 (IPY00006047)	34690 (34255)		Installation	A warning message from the operating system about Dialogic® HMP drivers not being digitally signed may be displayed during installation. Ignore the warning and continue installation. For information on disabling the warning message, refer to Section 4.5, "Disabling the Windows Driver Signing Check" in the *Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide*.
Known	IPY00006308	36489		IP	The Event Viewer shows an information message during startup stating that the default IP address has changed, even though it has not changed. None. Ignore this message.
Known	IPY00006281	36474		IP Gateway (Global Call) Demo	Call failures occur in the IP Gateway (Global Call) Demo when calls are made in the IP to PSTN direction. Stagger the calls in the IP to PSTN direction.
Known	IPY00006280	36472		IP Gateway (Global Call) Demo	The IP Gateway (Global Call) Demo may fail when more than six calls are made in the PSTN to IP direction when there is no staggering between calls. Stagger the calls in the PSTN to IP direction.
Known (permanent)	IPY00006275	36265		Licensing	Macrovision LMTOOLS utility may fail on machines running Windows XP Service Pack 2 if the System Settings tab is selected. The utility can be restarted and will work as normal. The tab in question is not necessary to activate HMP licensing on the system.
Resolved	IPY00006259	36149		Licensing	The lmpath utility does not work correctly, preventing CLI license activation from working as documented. This defect has been closed because lmpath is no longer recommended.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Resolved	IPY00006174	36261		Licensing	When using the Macrovision lmpath utility CLI, download may fail with the following error messages: \Device\SSP, LBRAC, 800C8, qkernerr.h, 800C9, qkernerr.h. This defect has been closed because lmpath is no longer recommended.
Known	IPY00092165	--		Licensing	When using HMP on a system with more than five NICs, the following error is seen "Exception: The FlexLM function lc_checkout( ) returned the error "(-9) Invalid Host." The host ID of this system does not match the host specified in the license file". Refer to the documentation updates in the Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide and the Dialogic® Host Media Processing Administration Guide for information about dealing with this problem. (IPY00091711)
Known	IPY00093028	--		Multimedia	Video play/record fails to allocate memory on 64-bit Windows systems. Observed behavior: the **mm_Play( )** and **mm_Record( )** functions return success, but then generate a failure event, (MMEV_*_ACK_FAIL). The reason code is EMMRC_MEMALLOC_POOLNOTFOUND. This is not an issue on 32-bit Windows systems.
Known	IPY00034238	--		Multimedia	**mm_Play( )** will not complete normally if the IPML session is stopped. Use the **mm_Stop( )** call to terminate the play operation.
Known	IPY00033481	--		Multimedia	If a video application exits abnormally while sessions are still streaming, error messages may print in dbgview during HMP stop. None
Known	IPY00032644	--		Multimedia	After exiting a multimedia application without stopping the mm plays and records, using **dev_Disconnect( )**, or closing the devices, subsequent runs of the application may not return completion events for multimedia play and record. None
Known (permanent)	IPY00032630	--		Multimedia	The **mm_Reset( )** function does not set parameters back to the default values. None

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00032597	--		Multimedia	Unable to run **mm_Play( )** or **mm_Record( )** on files that reside on network drives. Maintain files on local system.
Known (permanent)	IPY00032396	--		Multimedia	Calling **dev_Connect( )** immediately after calling **ipm_Listen( )** may result in missing completion events from mm play and record. Wait a short period of time following **ipm_Listen( )** before calling **dev_Connect( )**.
Known	IPY00034220	--		Multimedia Demo	The Multimedia Demo may not complete the **ipm_Stop( )** function when users hang up during the call or recording. None
Known	IPY00006305	36495		Station Interface	The Station Interface API function **si_GetDisplayCursorPos( )** does not fail if a station does not have a carrier. None
Known	IPY00081927 (IPY00081721, IPY00091533)	--		Voice	Voice resources do not always receive audio due to an issue with non-Windows 7 versions of the operating system and the Dell PowerEdge R610. To be sure that HMP software functions properly, the C-State power management must be disabled in the BIOS. Refer to Intel Xeon E5500 Nehalem with QuickPath Interconnects in Section 2.2, "Compatibility Notes", on page 350 for more information.
Known	IPY00039309	--		Voice	A TDX_LOWWATER event fails to return while running streaming to board. When a stream buffer is declared and filled in, a play is initiated. Under expected operation, a TDX_LOWWATER is received, and the application starts filling in the stream buffer until a TDX_HIGHWATER event is received. In this scenario, the play is active, but no indication of end of stream, in the form of a TDX_LOWWATER event, is received.
Known	IPY00038972	--		Voice	The application does not receive TDX_PLAY after playing a file (until EOF). A PAUSE is issued after about one second, followed by a RESUME, but TDX_PLAY is never received.
Known	IPY00033331	--		Voice	HMP may detect a 1605/1735 Hz dual tone signal even though it is set to only allow a 20Hz detection deviation. None

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN

Issue Type	Defect No.	PTR No.†	SU No.	Product or Component	Description
Known	IPY00006093	34242		Voice	Under heavy load, the TDX_UNDERRUN event may not occur when stream data is played out. None is required. The application should recover.
Known (permanent)	--	--		Voice	The **dx_reciottdata( )** and **dx_playiottdata( )** functions do not support recording or playing in WAVE format to/from memory; only VOX format.
Known (permanent)	--	--		Voice	Global Tone Detection (GTD) does not support detecting user defined tones as digits via the digit queue; the tones are only detected as tone events via the event queue.
Known (permanent)	--	--		Voice	For both single and dual tone definition, the frequency deviation that is defined in the tone template for each frequency must be not less than ± 30 Hz.
Known (permanent)	--	--		Voice	The number of tone templates which can be added to a voice device and enabled for detection is limited. The maximum number of events for each instance is 10.
Known (permanent)	--	--		Voice	The **dx_dial( )** function only detects CED tones for CR_FAXTONE event. CNG tones cannot be detected.
Known (permanent)	--	--		Voice	The **dx_playtoneEx( )** function does not terminate after the time specified by the DX_MAXTIME termination parameter has elapsed.
Known (permanent)	--	--		Voice	DTMF digits not processed by the user application with **dx_getdig( )** or other means remain in the device after it is closed with **dx_close( )**, and remain with the device even after the application terminates. To ensure that the buffer is empty, clear the digit buffer using **dx_clrdigbuf( )**.
Known (permanent)	--	--		Voice	The **dx_getctinfo( )** and **fx_getctinfo( )** functions return incorrect values for Device Family and Bus Mode.

†Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding Change Control System (CCS) defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the CCS number is used to track the issue.

2.2 Compatibility Notes

Adjusting the TimedWait State

Running ONLY call control on 10 or more time slots may cause the following error:

"IPEC_Q931Cause18NoUserResponding"

Each IP call uses a Windows socket which binds the call to a unique TCP address/port. The Global Call stack uses these ports starting at port address 20000. When an IP call is completed, the socket associated with that call closes and then enters into a TimedWait state, during which the socket's associated address/port is not available for use until the time expires. The default time for this TimedWait state is 240 seconds.

If an application is stopped and then immediately restarted before the TimedWait state expires, as may be the case during application development and debugging, calls may fail. Reducing the duration of the TimedWait state can alleviate this problem.

Another problem that may result from the TimedWait state duration is when a server experiences a high call rate. Even though the maximum number of TCP connections that can be opened simultaneously is large, in a high call rate scenario the potential exists for hundreds of TCP sockets to be in the TimedWait state causing the system to reach the maximum number of TCP connections. Again, reducing the duration of the TimedWait state can alleviate this problem.

Changing the TimedWait state to a value less than the 240 second default requires a change to the registry:

System Key: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters

Parameter Name: TcpTimedWaitDelay

Value Type: REG_DWORD (DWORD Value)

Valid Range: 30 - 300 seconds

Also, see the following Microsoft information at these links:

- Windows XP*http://support.microsoft.com/default.aspx?scid=kb;en-us;314053*
- Windows 2003 Server*http://support.microsoft.com/default.aspx?scid=kb;en-us;120642*

APIC Timer and Dialogic® HMP

Dialogic® HMP Software uses the Advanced Programmable Interrupt Controller (APIC) timer for its high resolution timer. The APIC driver may conflict with the processor throttling feature of power management. To prevent errors or degraded quality, verify that processor throttling is disabled. To determine if processor throttling is disabled, run the command `powercfg /query`:

```
Field Description         Value
-----------------         -----
Name                      Portable/Laptop
Numerical ID              1
Turn off monitor (AC)     Never
Turn off monitor (DC)     Never
Turn off hard disks (AC)  After 20 mins
Turn off hard disks (DC)  After 30 mins
System standby (AC)       After 25 mins
System standby (DC)       After 60 mins
System hibernates (AC)    Never
System hibernates (DC)    After 120 mins
Processor Throttle (AC)   ADAPTIVE
Processor Throttle (DC)    ADAPTIVE
```

If the command returns the following values, then an error may occur.

```
Processor Throttle (AC)    ADAPTIVE or DEGRADE
Processor Throttle (DC)     ADAPTIVE or DEGRADE
```

To prevent errors, there are two methods that can be used to change the power management scheme.

Method 1: Under Windows 2003, select Control Panel > Power Options Properties window, and then select Always On for the power scheme. This will disable processor throttling.

Method 2: Use the powercfg utility. This method disables processor throttling while keeping all other power management options. Copy the current power management scheme then use the command:

```
powercfg /change "Portable/Laptop" /processor-throttle-ac
NONE /processor-throttle-dc NONE
```

Note: The quotes around the scheme name are necessary for the Portable/Laptop scheme.

Intel Xeon E5500 Nehalem with QuickPath Interconnects

Some systems using Intel Xeon E5500 Nehalem processors with QuickPath Interconnects may experience irregular RTP packet generation. The issue is caused by the incorrect mapping of the Local APIC Timer's power C-state. Disabling C-State power management in the BIOS or updating the machines BIOS is necessary for correct operation. This issue is related to problem Microsoft has documented at *http://support.microsoft.com/kb/2000977*. It has also been seen a Dell PowerEdge R610 and a NEC Express 5800 system. Both systems operated normally after disabling C-state power management in the BIOS. Refer to IPY00081721, IPY00081927, and IPY00091533 in Table , "Issues Sorted by SU No., Dialogic® Host Media Processing Software Release 3.0WIN", on page 278.

Configuring UDP/RTP Port Range

Note: In addition to the following procedure, you may also use the structure IPM_PARM_INFO, associated with the **ipm_GetParm( )** and **ipm_SetParm( )** API functions, to configure the UDP/RTP port range. For more information, refer to Chapter 4, "Data Structures" in the *Dialogic® IP Media Library API Programming Guide and Library Reference*.

The HMP system currently defaults to UDP/RTP ports 6000 through 6100 and UDP ports 49152 through 49xxx for RTP streaming, where 49xxx = 49152 + twice the maximum number of channels purchased under the licensing agreement. For example, if 120 channels were purchased, 49xxx would be 49392.

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

1. Stop the system service using DCM.
2. Locate the .config file in the *C:\Program Files\Dialogic\HMP\data* directory that matches the FCD/PCD files associated with your licensed configuration.
3. Using a text editor such as NotePad or WordPad, open the .config file.
4. Locate the [IPVSC] section in the .config file.
5. In the [IPVSC] section, locate the line

   ```
   setparm 0x4005, 49512      !set the rtpPortBase on IPVSC
   ```

   The number 49512 is the default value for this parameter. You may change the beginning of the UDP/RTP port range by first editing this value and saving the .config file.
6. After you have saved the .config file with the new UDP/RTP port value, open the Command Prompt window.
7. From the Command Prompt, change the directory to *C:\Program Files\Dialogic\HMP\data*.

8. Execute fcdgen as follows:

```
..\bin\fcdgen -f <input filename>.config -o <output filename>.fcd
```

   The resulting FCD file is created in the *C:\Program Files\Dialogic\HMP\data* directory. If the -o option is omitted from the command, the default output FCD file will have the same filename as the user-modified input .config file, but with an .fcd extension.

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

Configuring T.38 Service Port Range

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

1. Stop the system service using DCM.
2. Locate the .config file in the *C:\Program Files\Dialogic\HMP\data* directory that matches the FCD/PCD files associated with your licensed configuration.
3. Using a text editor such as NotePad or WordPad, open the .config file.
4. Locate the [0xe] section in the .config file.
5. In the [0xe] section, locate the line:

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

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

6. After you have saved the .config file with the new T.38 service port value, open the Command Prompt window.
7. From the Command Prompt, change the directory to *C:\Program Files\Dialogic\HMP\data*.
8. Execute *fcdgen* as follows:

```
..\bin\fcdgen -f <input filename>.config -o <output filename>.fcd
```

   The resulting FCD file is created in the *C:\Program Files\Dialogic\HMP\data* directory. If the -o option is omitted from the command, the default output FCD file will have the same filename as the user-modified input .config file, but with an *.fcd* extension.

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

H.323 Coder Negotiation with Third Party Stacks

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

Example:

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

SIP Call Control

If the remote site does not respond to an outgoing INVITE sent from HMP, the **gc_MakeCall( )** function will time out after 32 seconds and generate a GCEV_DISCONNECTED event. In this scenario, if the application attempts to drop the call before the 32 second timeout is reached, a CANCEL will be sent by HMP to the remote site. If there is no response by the remote site to the CANCEL, there will be an additional 32 second timeout, at the end of which, a GCEV_DISCONNECTED event will be reported.

Avoiding UPD/RTP Port Conflicts

When using Windows 7 or Windows Server 2008, with Dialogic® HMP Software, a UDP ports conflict between HMP media (starting at 49152 by default) and other applications or software components may occur. As a result, the **ipm_StartMedia( )** function fails internally.

The reason for the conflict is that the operating system allocates the UDP/TCP port automatically with the dynamic port pool range changed in Windows 7 or Windows Server 2008. By default, the new port pool range is between 49152 and 65535. When an application attempts to use a UDP socket by specifying UDP port number 0, Windows automatically allocates one from the dynamic port number pool. With previous version of Windows, such as Windows 2003, the dynamic port range was between 1025 and 5000.

Proceed as follows to avoid any potential UDP/RTP port number conflict between HMP and other applications or processes:

1. Use the netsh command to notify Windows to use alternative range for dynamic ports. This example specifies start port number 49520 for 1000 ports:

   ```
   netsh int ipv4 set dynamic udp
   start=49520 num=1000
   ```

   Review the current setting using the following command:

   ```
   netsh int ipv4 show dynamicportrange udp
   ```

2. Modify HMP RTP port starting number in following line under the [IPVSC] section of the *.config* file:

   ```
   SetParm=0x4005,49152
   ```

3. Generate the *.fcd* file using FCGGEN.

For additional information, refer to *http://support.microsoft.com/kb/929851* and the UDP Port Base for Audio RTP parameter in the [IPVSC] IP Media Parameters section of the *Dialogic® Host Media Processing Configuration Guide*.

Documentation Updates 3

This chapter contains information on updates and corrections to the documents included in Dialogic® PowerMedia™ HMP for Windows Release 3.0.

The documentation updates are divided into the following sections, which correspond to the top level categories used in the online document navigation page:

3.1 Release Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide

3.1.1 Dialogic® Host Media Processing Software Release for 3.0WIN Release Guide

In Service Update 367, under section 2.2 “Basic Software Requirements” add the following note:

If using Windows 7 or Windows Server 2008 R2, Windows security updates 3033929 and 3035131 must be installed prior to HMP installation.

In Service Update 328, under section 2.1 “Basic Hardware Requirements” add a subsection (2.1.1) and add the following:

Data Execution Prevention (DEP)

Due to the nature of DEP, it must be disabled for Dialogic kernel-level objects since they may access protected memory locations. This is irrespective of the target Operating System version; however, the procedure and DEP settings may be different for the various Operating System versions supported. Run the following commands to control DEP settings for Windows 7 or Windows Server 2008 operating systems:

```
bcdedit /set nx [Optin |AlwaysOff]
```

AlwaysOff disables DEP and any attempts to enable DEP selectively are ignored. **Optin** enables DEP only for operating system components, including the Windows kernel and drivers. Administrators can enable DEP on selected executable files by using the Application Compatibility Toolkit (ACT). Any other setting might cause erratic behavior on the Dialogic System Services. For example:

```
bcdedit /set nx Optin
```

Note: The **AlwaysOff** option is more restrictive; the decision to use one or another option is left to the discretion of the system administrator.

Please consult the Microsoft documentation regarding DEP settings for the specific Windows Operating System version in the target system.

3.2 Installation and Configuration Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide
- Dialogic® Host Media Processing Configuration Guide
- Dialogic® Global Call CDP Configuration Guide
- Dialogic® Springware Architecture on Windows Configuration Guide
- Dialogic® DCM Configuration Manager Online Help

3.2.1 Dialogic® Host Media Processing Software Release 3.0WIN Software Installation Guide

Installation Package Policy

Dialogic® PowerMedia™ HMP for Windows Release 3.0 is an InstallShield-based installation package delivered as a zipped .zip file for installation on an existing Microsoft Windows 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 343, the Microsoft Intelligent Platform Management Interface (IPMI) Compatibility Device in Windows Server 2012, if enabled, may cause audio degradation to HMP media engine.

This has been observed at random times over long periods of continuous HMP operation. The degradation maybe coming in short spurs of audio clicks which would occur on all media channels at once which will shortly subside. The effect of this degradation depends on the media being streamed, i.e. voice, multimedia, and/or tones.

The IPMI Compatibility Device is Microsoft's Intelligent Platform Management Interface implementation for Windows Server 2012.

The problem is likely rooted in a conflicting behavior the Windows IPMI subsystem might have on the HMP real-time timing devices, such as HPET, in the case of an IP-only system, or the HW interrupts from a Board-based system. Because HMP media processing depends on a stable timing source, when the source is compromised, so is the quality of media streamed between the external interfaces (such as IP or TDM), and the core HMP real-time SW running on the host CPU.

On Board-based systems, if real-time timing from the HW is compromised, Clock Gap warnings would be observed if an application that lets you monitor debug real-time system kernel output on your local system, such as DebugView, or a similar system debugger. In such case HMP will log entries such as:

```
"KMBC.SYS: CLOCK GAP (mid=nnn,count=xxxx,gap=yy)"
```

And/or:

```
"KMBC.SYS: POSSIBLE LOSS OF CLOCK (mid=nnn)"
```

Where "nnn", "xxxx", and "yy" may vary from system to system, and on each entry.

On IP-only systems these entries would not exist; there might not be any specific logging indicating a timing (clock) gap existed.

While it might be possible to configure the Microsoft IPMI Compatibility Device to avoid the timing device clocking, in order to safely eliminate it, is recommended to disable the Microsoft IPMI Compatibility Device in Windows Server 2012. This can normally be done via the Windows Device Manager; under System Devices find "Microsoft IPMI Compatibility Device", then device right-click on it and select Disable. Please consult the Windows Server 2012 help for specific information that pertains to your Windows installation.

Note: The above description applies to the Microsoft IPMI subsystem for Windows 2012; however your system manufacturer might also provide their own IPMI subsystem implementation and interface. In such cases, if HMP audio degradation is also observed it is recommended disabling the IPMI subsystem through the platform-specific interface as well.

With Service Update 240, 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 Configuration Guide

Update to **Configuration Overview** section

The following note should be added:

Note: Dialogic® PowerMedia™ HMP for Windows Release 3.0 supports only one IP IPv4 address and one IPv6 address assigned to a network interface.

In Service Update 328, update to **section 4.9 "Preserving Data in User Configuration Files"** (IPY00100643)

Add the following note under Step 1:

Note: For **64-bit** Windows 7 or Windows Server 2008 operating systems, the *Hmp.Uconfig* file should be created in data directory under INTEL_DIALOGIC_DATA_DIR pointed to by the environment variable.

3.2.3 Dialogic® Global Call CDP Configuration Guide

A new version of this document is now available on the bookshelf.

3.2.4 Dialogic® Springware Architecture on Windows Configuration Guide

This document is added to the bookshelf to support the following analog devices:

- Dialogic® D/80PCIE-LS
- Dialogic® D/4PCIUFEQ
- Dialogic® D/4PCIU4SEQ

See Section 1.60, "Support for Dialogic® D/4PCIU Boards", on page 143 and Section 1.67, "Support for the Dialogic® D/80PCIE-LS Media Board", on page 157 for more information.

3.2.5 Dialogic® DCM Configuration Manager Online Help

There are currently no updates to this document.

3.3 OA&M Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Dialogic® Host Media Processing Administration Guide
- Dialogic® SNMP Agent Software Administration Guide
- Dialogic® Host Media Processing Diagnostics Guide
- Dialogic® Event Service Programming Guide
- Dialogic® Event Service Library Reference
- Dialogic® Native Configuration Manager API Programming Guide
- Dialogic® Native Configuration Manager API Library Reference

3.3.1 Dialogic® Host Media Processing Administration Guide

Add a link for FLEXnet documentation in Section 3.4.

FLEXnet documentation describing how to Manage Licenses from Multiple Vendors can be found at *http://www.globes.com/support/utilities/flexnet_licensing_end_user_guide.pdf*. Refer to Chapter 2, Managing Licenses from Multiple Vendors, in that document.

In Service Update 291, update to Chapter 6, “Displaying and Changing the Dialogic® HMP Software Default IP Address”

In the Introduction section, Note 1. should be replaced with the following text: “With Windows Server 2008, an inactive NIC may be selected as the primary NIC when the NIC instance to be used by the DCM "Default IP Address" tab is not explicitly specified. As a result, Dialogic® HMP Software may not start or function properly. With both Windows Server 2003 and Windows Server 2008, on systems that have multiple NICs with multiple IPs, it is possible that the default IP address picked by the Dialogic® HMP Software may not be the IP address best suited for your purposes. Therefore, **after installation and before starting the Dialogic® HMP Software**, you should check the default IP address and change it, if necessary, to suit your specific needs. The system will remember the selected default IP address and associated NIC controller so you do not need to check the address on subsequent reboots unless you want to assign a different default IP address.”

In Service Update 289, update to Chapter 6, “Displaying and Changing the Dialogic® HMP Software Default IP Address” (IPY00091711)

Add the following note to the Introduction section:

Note: There is a restriction when assigning Dialogic® HMP Software licenses to a NIC in systems with more than five NICs. The HMP license must be tied to one of the first five NICs listed in the NIC priority list, otherwise the license will fail to activate. The NIC priority list can be changed under Advanced Settings of Manage Network Connection in the Windows Control Panel, allowing any NIC to be used for the HMP license.

Update to Chapter 12, “Troubleshooting” (IPY00044223)

A new section on “TDM Bus Messages to Ignore When Using Dialogic® HMP Software Without Dialogic® HMP Interface Boards” should be added in this chapter.
The Dialogic HMP Software system without Dialogic HMP Interface Boards is essentially an IP-based product and does not use the TDM bus. If the Dialogic HMP Software system configuration is reset to default settings, for example via DCM, the following messages may be generated and displayed in the Windows Event Viewer log. These messages are related to the TDM bus technology and can be ignored.

```
TDM bus user settings were rewritten because of bus configuration
failure 0x20000060. This will happen if the resources have changed
and is usually not cause for alarm.
```

Explanation: This message correctly shows that the system was reset to default settings (restore default settings).

```
During bus configuration, the user selected media type was not supported
by all FRUs, or the FRUs don't support a common media type.
```

Explanation: This message correctly shows that the default TDM Bus settings are not supported by all FRUs (boards). Since the HMP Software system is a virtual board, this message correctly reports that the default TDM bus media type is not supported by the HMP virtual device (board).

For systems that include Dialogic HMP Interface Boards, these messages should not be ignored as they may indicate a problem.

Update to Chapter 12, “Troubleshooting” (IPY00044247)

The section on “Errors Using Dialogic® HMP Software with RealSpeak 4.0” is renamed “Errors Caused by Third-Party Software Conflicts with FLEXnet Licensing”. The first paragraph is revised to cover third-party software in general because this error may occur with other software. The revised text is in italics.

Using Dialogic® HMP Software and *certain third-party software (such as RealSpeak 4.0 from Nuance Communications Inc. or Crystal Reports from Business Objects, an SAP company)* in the same chassis may result in an error *due to possible conflicts* with the FLEXnet Licensing System from Macrovision Corp. Because the port IDs are not specified for each system, the FLEXnet Licensing System automatically uses the first available TCP/IP port in the range of 27000 – 27009. An error results because Dialogic HMP Software and *the third-party software system* are assigned to use the same first 27000 port.

3.3.2 Dialogic® SNMP Agent Software Administration Guide

There are currently no updates to this document.

3.3.3 Dialogic® Host Media Processing Diagnostics Guide

Update to **Chapter 20, QScript Reference**

With Service Update 393, QScript utilities support has been deprecated.

Update to **Chapter 21, Runtime Trace Facility (RTF) Reference** (IPY00037518)

The following information about using binary should be added to **Section 21.3.2, Logfile Tag**:

For installations with high channel densities, or which have enabled all or most RTF trace levels, the volume of logging may result in an increased CPU utilization by the RtfServer executable as a result of the increased volume of log messages.

As shipped, the RTF log files are generated in ASCII text mode. There is a configuration parameter in the RTF configuration file (*RtfConfigWin.xml* for Windows, *RtfConfigLinux.xml* for Linux) that allows log files to be generated in either "text" or "binary" format. Testing on high channel density systems with most or all of the RTF trace levels enabled has shown that the generation of binary format RTF log files has less of an impact on CPU usage than does the generation of text format RTF log files. If the volume of logging results in high CPU usage, then using binary format will reduce the usage.

Enabling Binary Format RTF Log Files

The XML file contains the following line, which allows changes to log file parameters to be made:

```
<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="300"
maxbackups="10" preserve_size="300" preserve_maxbackups="10"
duplicate_to_debug_console="0" log_format="text" />
```

The "log_format" value controls the type of log files that are written. Valid values for this parameter are "text" and "binary". Once a change has been made to the XML file, it must be reloaded using the rtftool reload command.

Converting Binary Format RTF Log Files to Text Format

In order for binary log files to be examined, they must be converted into text format. This can be done by using the rtftool export command.

```
rtftool export [-d source_dir | -s
source_file] [-f [dest_file] | -m dest_dir]
```

By default, the name of the text format files generated by this command will be *EXPORT-<RTF binary log file name>*. For example, if the binary format file is named *rtflog-LOCAL-20070306-15h09m26.506s.txt*, then the default name of the generated text format file will be *EXPORT-rtflog-LOCAL-20070306-15h09m26.506s.txt.* This behavior can be overridden using the -f command line option.

The rtftool utility is a stand-alone program, and it is not necessary to have the Dialogic System Release installed on the system in order to convert RTF log files from binary to text format.

Note: When generating large binary files with RTF, do not split the single large binary file and then use the individual split files with the rtftool utility. Rtftool will not work with chopped binary files.

3.3.4 Dialogic® Event Service Programming Guide

There are currently no updates to this document.

3.3.5 Dialogic® Event Service Library Reference

There are currently no updates to this document.

3.3.6 Dialogic® Native Configuration Manager API Programming Guide

There are currently no updates to this document.

3.3.7 Dialogic® Native Configuration Manager API Library Reference

Update to **NCM_StartSystem( )** function:

The note in the Description section is invalid and should be removed. The function description should read as follows:

After a system reboot, if the Dialogic® HMP software system is set to manual mode, the **NCM_StartSystem( )** function starts the Dialogic system services and starts all boards. If the Dialogic® HMP software system is set to semi-automatic mode, the **NCM_StartSystem( )** function starts all boards only (the Dialogic system services already are running).

If the Dialogic® HMP software system was stopped, the **NCM_StartSystem( )** function starts the Dialogic system services and starts all boards.

Update to **NCM_StartDlgSrv( )** function:

The second note in the Description section is invalid and should be removed. The function description should read as follows:

The **NCM_StartDlgSrv( )** function initiates the entire Dialogic system and starts all boards.To start only one board, use **NCM_StartBoard( )** function.

Note: A successful completion code for this function (NCM_SUCCESS) only indicates that a start message was sent to the Dialogic system. Use **NCM_GetDlgSrvState( )** or **NCM_GetDlgSrvStateEx( )** to determine whether or not the system actually started.

Update to **NCM_StopDlgSrv( )** function:

The second note in the Description section is invalid and should be removed. The function description should read as follows:

The **NCM_StopDlgSrv( )** function stops the entire Dialogic system and stops all boards. On HMP, the Boardserver service (BoardServer.exe for the SNMP Agent Software functionality) will be stopped as well. If the Boardserver service was running prior to stopping the entire Dialogic system, Boardserver must be restarted manually. If the Boardserver service is set to start automatically when the system starts, it will start automatically for the new session. The Boardserver service is not a part of the **NCM_StartDlgSrv( )** or DCM GUI startup sequence. To stop only one board, use the **NCM_StopBoard( )** function.

Note: A successful completion code for this function (NCM_SUCCESS) only indicates that this function attempted to stop the system. Use **NCM_GetDlgSrvState( )** or **NCM_GetDlgSrvStateEx( )** to determine whether or not the system was actually stopped.

Update to **NCM_StopSystem( )** function:

The note in the Description section is invalid and should be removed. The function description should read as follows:

The **NCM_StopSystem( )** function stops all boards in the system and, and in some cases stops the Dialogic system service. If your system is running in semi-automatic mode, the **NCM_StopSystem( )** function will stop all boards in the system, but will not stop the Dialogic system service. If your system is running in automatic or manual mode, the **NCM_StopSystem( )** function will stop all boards and stop the Dialogic system service.

3.4 Programming Libraries Documentation Updates

This section contains updates to the following documents:

- Dialogic® Audio Conferencing API Programming Guide
- Dialogic® Audio Conferencing API Library Reference
- Dialogic® Conferencing API Programming Guide
- Dialogic® Conferencing API Library Reference
- Dialogic® CSP API Programming Guide
- Dialogic® CSP API Library Reference
- Dialogic® Device Management API Library Reference
- Dialogic® Digital Network Interface Software Reference
- Dialogic® Fax Software Reference
- Dialogic® Global Call API Programming Guide
- Dialogic® Global Call API Library Reference
- Dialogic® Global Call Analog Technology 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® Learn Mode and Tone Set File API Software Reference
- Dialogic® Multimedia API Programming Guide and Library Reference
- Dialogic® SRL API Programming Guide
- Dialogic® SRL API Library Reference
- Dialogic® Station Side Interface API Library Reference
- Dialogic® Voice API Library Reference
- Dialogic® Voice API Programming Guide

3.4.1 Dialogic® Audio Conferencing API Programming Guide

Update to Section 9.2, “Input Volume Automatic Gain Control”

Removed the following note: “This feature is **not** supported on Dialogic® Host Media Processing (HMP) Software.”

3.4.2 Dialogic® Audio Conferencing API Library Reference

Update to Chapter 2, “Function Information”

Add two functions **dcb_setcnfparm( )** and **dcb_getcnfparm( )** for conference tone notifications. See Section 1.135, “Enhancements to CNF and DCB Notification Tone”, on page 256, for more details.

3.4.3 Dialogic® Conferencing API Programming Guide

Update to Section 5.3, “Creating a Conference” (IPY00090909)

Add the following sentence to item number 3:

Please refer to the note following these steps for information about resource limitations.

Add the following note after Step 15:

Note: When a license has voice, RTP, and conferencing devices, there are cases where not all of the number of conference parties as mentioned in the license will be opened due to an internal limitation of 2048 transmit time slots on Windows. This is because the Dialogic Windows stack has an internal limitation on the maximum number of time slots it can allocate system wide, currently 2048 time slots. This limitation of 2048 is only on Windows operating systems due to memory constraints. The Dialogic stack allocates the transmit time slots for voice and RTP devices when the devices are created during board download, while the transmit time slots for the audio conference party is allocated when a party is added to a conference.

Each device requires a different number of time slots, hence Dialogic stack may not be able to open the exact number of devices mentioned in the license if the time slot usage exceeds the 2048 time slot limit.

Consider the following examples provided herein. Note that each voice device and each RTP requires one transmit time slot each, while each conference party requires two transmit time slots internally.

```
License of 520r520v0e520c0s0f0i0m_host.lic namely 520 G.711, 520
Voice and 520 Conference Parties
```

At board download, all voice and RTP channels together will consume 1040 transmit time slots as 520 voice devices will require 520 transmit time slots and 520 RTP devices will require 520 transmit time slots. Thus, 2048 minus 520 twice will leave 1008 transmit time slots for conference. As each conference party requires two transmit time slots per party, the Dialogic Stack will allow only 504 parties (1008/2) and the addition of 505th party will fail. Even when the license allows up to 520 conference parties, the combination of voice, RTP and conference party devices limits the number of parties to 504.

```
License of 510r510v0e510c0s0f0i0m_host.lic namely 510 G.711, 510
Voice and 510 Conference Parties
```

At board download, all voice and RTP channels together will consume 1020 transmit time slots as 510 voice devices will require 510 transmit time slots and 510 RTP

devices will require 510 transmit time slots. Thus, 2048 minus 510 twice will leave 1028 transmit time slots for conference. As each conference party requires two transmit time slots per party, the Dialogic Stack will allow 514 parties (1028/2). In this scenario, the maximum number of conference parties specified in the license (510 parties) can be achieved.

3.4.4 Dialogic® Conferencing API Library Reference

Service Update 317, updates the **cnf_SetDTMFControl( )** description in Chapter 2, “Function Information”

The **cnf_SetDTMFControl( )** function description is incorrect. Instead of “returns information about the DTMF digits used to control the conference behavior” it should read “sets information about the DTMF digits used to control the conference behavior.”

Update to Chapter 2, “Function Information”

Conferencing (CNF) has been modified to include a new parameter for the **cnf_SetAttribute( )** and **cnf_GetAttribute( )** functions: **ECNF_CONF_ATTR_NOTIFY**. See Section 1.135, “Enhancements to CNF and DCB Notification Tone”, on page 256, for more details.

Add the following new API (IPY00035489):

cnf_OpenEx( )

Name SRL_DEVICE_HANDLE cnf_OpenEx (a_szBrdName, a_pOpenInfo, a_pUserInfo, a_usMode)

Inputs:

const char * a_szBrdName	• pointer to virtual board device name
CPCNF_OPEN_INFO a_pOpenInfo	• reserved for future use
void * a_pUserInfo	• pointer to user-defined data
unsigned short a_usMode	• synchronous/asynchronous mode specifier

Returns: Virtual board SRL device handle if successful
CNF_ERROR on failure

Includes: cnflib.h

Category: Initialization

Mode: synchronous/asynchronous

■ Description

The **cnf_OpenEx( )** function opens a virtual board device and returns a unique SRL handle to identify the device. The naming convention for a virtual board device is "cnfBx", where x is the board number starting at 1. All subsequent references to the opened device must be made using the handle until the device is closed.

The **cnf_OpenEx( )** function allows you to choose synchronous or asynchronous mode. If you require operation in synchronous mode, use **cnf_OpenEx( )** instead of **cnf_Open( )**.

Parameter	Description
a_szBrdName	points to a virtual board device name
a_pOpenInfo	reserved for future use. Must be set to NULL.

Parameter	Description
a_pUserInfo	points to user-defined data. If none, set to NULL.
a_usMode	specifies synchronous/asynchronous mode. Valid values are: • EV_SYNC • EV_ASYNC ***Note:*** There is no default setting for mode.

Synchronous Mode

If this function is called in synchronous mode, then if successful, the returned SRL handle is a valid handle that can be used to further communicate with the board device.

Asynchronous Mode

If this function is called in the asynchronous mode, then if successful, the returned SRL handle will not be valid until the CNFEV_OPEN event is reported on the SRL handle to indicate successful initialization of the device. If a failure occurs, the device is not opened and the CNFEV_OPEN_FAIL event will be reported on the SRL handle returned from **cnf_OpenEx( )**.

■ Termination Events

The following is a list of events that can be returned as a completion to this request when used in asynchronous mode.

CNFEV_OPEN
: indicates successful completion of this function; that is, a virtual board device was opened Data Type: NULL

CNFEV_OPEN_FAIL
: indicates that the function failed
Data Type: NULL

Note: Application must call **cnf_Close( )** to clean up if CNFEV_OPEN_FAIL is received.

■ Cautions

- In applications that spawn child processes from a parent process, the device handle is not inheritable by the child process. Make sure devices are opened in the child process.
- The **a_pOpenInfo** parameter is reserved for future use and must be set to NULL.
- The same virtual board device can be opened in multiple processes; one process can delete a conference running on another process on the same virtual board device. It is up to you to synchronize access to the same virtual board device from multiple processes.

■ Errors

If this function fails with CNF_ERROR, use **cnf_GetErrorInfo( )** to obtain the reason for the error. Refer to **cnf_GetErrorInfo( )** for a list of possible error values. Refer to Chapter 5, "Error Codes" in the *Conferencing API Library Reference* for a list of error codes.

Synchronous Code Example

```
#include <cnflib.h>

int main(int argc, char *argv[])
{
    SRL_DEVICE_HANDLE BrdDevice; /* Virtual board device handle. */
    if ((BrdDevice = cnf_OpenEx("cnfB1", NULL, NULL, EV_SYNC)) == CNF_ERROR)
    {
         cout << "cnf_OpenEx failed!!" << endl;
         /* process error */
         return 0;
    }
    else
    {
         cout << "cnf_OpenEx Successful..." << endl;
         /* Open successful - May now use BrdDevice handle. */
    }
```

Asynchronous Code Example

```
#include <cnflib.h>

int main(int argc, char *argv[])
{
    SRL_DEVICE_HANDLE BrdDevice; /* Virtual board device handle. */
    if ((BrdDevice = cnf_OpenEx("cnfB1", NULL, NULL, EV_ASYNC)) == CNF_ERROR)
         cout << "cnf_OpenEx failed !!" << endl;
         /* process error */
         return 0;
    }
    else
    {
         if (sr_waitevt(10000) == -1)
         {
             cout << "sr_waitevt TIMEOUT failure" << endl;
             /* process error */
             return 0;
         }
         else
         {
             unsigned int unEvent = sr_getevttype();
             switch(unEvent)
             {
                 case CNFEV_OPEN:
                     /* Open successful - May now use BrdDevice handle */
                     break;

                 case CNFEV_OPEN_FAIL:
                     /* Open failed - Process failure and must close device */
                     cnf_Close(BrdDevice, NULL);
                     break;

                 default:
                     /* Received some other event - Process this event */
                     break;
             };
         }
    }
}
```

See Also

- **cnf_Close( )**

Add the following new API (IPY00035507):

cnf_ResetDevices( )

Name: CNF_RETURN cnf_ResetDevices(SRL_DEVICE_HANDLE a_BrdHandle, CPCNF_RESET_DEVICES_INFO a_pResetInfo, void *a_pUserInfo)

Inputs:

a_BrdHandle	• SRL handle to the virtual board device
a_pResetInfo	• reserved for future use
a_pUserInfo	• pointer to user defined data

Returns: CNF_SUCCESS for success
CNF_ERROR for failure

Includes: cnflib.h

Category: Initialization

Mode: Asynchronous

Description

The **cnf_ResetDevices( )** function resets all devices that may have been opened and not closed by a previous process for the specified board. This function should only be used to recover conference and party devices that were not properly closed due to an abnormal or improper shutdown of some process, and should not be used otherwise.

Parameter	Description
a_BrdHandle	specifies an SRL handle to the virtual board device
a_pResetInfo	reserved for future use. If none, set to NULL.
a_pUserInfo	points to user-defined data

Events

If CNF_SUCCESS is returned, the user is notified of the completion status of this request via one of the events listed below, otherwise CNF_ERROR will be returned.

CNFEV_RESET_DEVICES
Reset devices successful or no devices to recover

CNFEV_RESET_DEVICES_FAIL
Reset devices failure

Cautions

- This function should only be used to recover previously opened devices that were not closed due to an abnormal shutdown of a process. The most common use of this function is to call it at the beginning of an application in order to make sure that the firmware conferencing resources are properly reset. The function will return the CNFEV_RESET_DEVICES event if it successfully recovered one or more CNF devices, or if there were no devices to recover.

Errors

If this function fails with CNF_ERROR, use **cnf_GetErrorInfo( )** to obtain the reason for the error. Refer to **cnf_GetErrorInfo( )** for a list of possible error values.

Example

```
#include <cnflib.h>
int main(int argc, char *argv[])
{
    SRL_DEVICE_HANDLE BrdDevice; /* Virtual board device handle. */

    if ((BrdDevice = cnf_Open("brdB1", NULL, NULL)) == CNF_ERROR)
    {
         cout << "cnf_Open failed !!" << endl;
         /* process error */
         return 0;
    }
    else
    {
         if (sr_waitevt(10000) == -1)
         {
              cout << "sr_waitevt TIMEOUT failure" << endl;
              /* process error */
              return 0;
         }
         else
         {
              unsigned int unEvent = sr_getevttype();
              switch(unEvent)
              {
                  case CNFEV_OPEN:
                      /* Open successful - May now use BrdDevice handle */
                      break;

                  case CNFEV_OPEN_FAIL:
                      /* Open failed - Process failure and must close device */
                      cnf_Close(BrdDevice, NULL);
                      exit(0);
                      break;

                  default:
                      /* Received some other event - Process this event */
                      break;
              };
         }
    }

    /**
     * We could use the cnf_GetDeviceCount( ) function to determine if we have
      * any allocated conference or party devices that need to deallocated or
     * we could decide to always reset the board devices by default. If so,
     * we use the cnf_ResetDevices to force a deallocation of these devices.
```

```
 */

if ((cnf_ResetDevices(BrdDevice, NULL, NULL)) ==
CNF_ERROR {
    cout << "cnf_ResetDevices failed !!" << endl;
    /* process error */
    return 0;
}
else
{
   if (sr_waitevt(10000) == -1)
   {
       cout << "sr_waitevt TIMIEOUT failure" << endl;
       /* process error */
       return 0;
   }
   else
   {
       unsigned int unEvent = sr_getevttype();
       switch(unEvent)
       {
            case CNFEV_RESET_DEVICES:
                 /* Reset devices successful */
                 break;

            case CNFEV_RESET_DEVICES_FAIL:
                 /* Reset devices failure - lets use SRL to find reason */
                 break;

            default:
                 /* Received some other event - process this event */
                 Break;
       };
   }
}
}
```

- **See Also**

None.

3.4.5 Dialogic® CSP API Programming Guide

Update to Section 4.1.2, “Reserving Extra Time Slots for Streaming to TDM Bus” The following API functions are not supported by Dialogic® HMP:
ec_getxmitslot( )
ec_rearm( )

3.4.6 Dialogic® CSP API Library Reference

Update to Chapter 2, “Function Information”

The following API functions are not supported by Dialogic® HMP:
ec_getxmitslot( )
ec_rearm( )

Updates to the **ec_getblkinfo( )** reference page (IPY00043040)

Remove the third bullet item under the Caution heading.

Replace the current example with the following new example code:

```
int stream_cb(int chDev, char *buffer, UINT length)
{

int rc;
   EC_BLK_INFO blkInfo;
    rc = ec_getblkinfo(&blkInfo);
    if (rc == 0){
       printf("type %d: flags %d: size %d: elapsed time %d\n",
              blkInfo.type,blkInfo.flags,blkInfo.size,blkInfo->timestamp);
   }else{
   printf("Error in ec_getblkinfo(). Err Msg = %s, Lasterror =
          %d\n", ATDV_ERRMSGP(chDev), ATDV_LASTERR(chDev)); }
    }
    /* Write recorded streaming data to file. */
    rc = _write(RecordFile[ecdev_to_channel[chDev]], buffer, length);
}
```

3.4.7 Dialogic® Device Management API Library Reference

Service Update 291, update to the **dev_Connect( )** function in support of analog devices.

Add the following connType value to the parameter table:

DM_ANALOG_INTF - Specifies that the Springware voice device handle passed as either devHandle1 or devHandle2 is treated as an analog front-end. This parameter should be ORed with either DM_FULLDUP or DM_HALFDUP.

Add the following to Supported Connections (after the existing connections):

Analog Front-End and CNF Audio Conference Party

A full-duplex or half-duplex connection between an audio conferencing party device (CNF API) and a Springware front-end device on board. Requires a valid audio conferencing party device handle obtained through the cnf_OpenParty( ) function and a valid voice device handle obtained through the dx_open( ) function. Both synchronous and asynchronous modes are supported. In the half-duplex connection, either type of device can listen to the other.

Springware Voice and CNF Audio Conferencing Party

A full-duplex or half-duplex connection between an audio conferencing party device (CNF API) and a Springware voice device on board. Requires a valid audio conferencing party device handle obtained through the cnf_OpenParty( ) function and a valid voice device handle obtained through the dx_open( ) function. Both synchronous and asynchronous modes are supported. In the half-duplex connection, either type of device can listen to the other.

Analog Front-End and HMP Voice Device

A full-duplex or half-duplex connection between a HMP voice device (Voice API) and an analog front-end device on board. Requires a valid HMP voice device handle obtained through the dx_open( ) function and a valid analog voice device handle obtained through the dx_open( ) function.

Both synchronous and asynchronous modes are supported. In the half-duplex connection, either type of device can listen to the other.

Note: Do not use the **dev_Connect( )** function to connect two analog devices. One of the device handles passed to the function must me an HMP device. If two analog device handles are passed to **dev_Connect( )**, the function will return an error.

Note: Analog devices do not support connections to IP Media devices.

Add the following code example (after the existing example):
Example D (Analog Front-end and CNF AudioConferening Party)

```
#include <srllib.h>
#include <dxxxlib.h>
#include <cnflib.h>
#include <devmgmt.h>
void main()

ag_handle = dx_Open("dxxxB1C1", NULL, EV_SYNC );
if( ag_handle == -1 )
{

      printf( "dx_Open() failed.\n" );
      exit( 1 );
}

int cnf_brdhandle = cnf_OpenEx("cnfB1", NULL, NULL ,EV_SYNC);
if( cnf_brdhandle == -1 )
{
      printf( "cnf_openEx() failed.\n" );
      exit( 1 );
}

cnf_Confhandle = cnf_OpenConference(cnf_brdhandle, NULL, NULL, NULL);
if( cnf_Confhandle == -1 )
{
      printf( "cnf_OpenConference () failed.\n" );
      exit( 1 );
}

int index = 0;
cnf_Partyhandle = cnf_OpenParty(cnf_brdhandle, NULL, NULL, &index);
if( cnf_Partyhandle == -1 )
{

      printf( "cnf_OpenParty () failed.\n" );
      exit( 1 );
}
if( sr_enbhdlr( ag_handle, EV_ANYEVT, DxEventHandler ) == -1 )
{
      printf( "sr_enbhdlr() failed.\n" );
      exit( 1 );
```

```
}
if( sr_enbhdlr( cnf_handle, EV_ANYEVT, CnfEventHandler ) == -1 )
{
      printf( "sr_enbhdlr() failed.\n" );
      exit( 1 );
}
if( dev_Connect( ag_handle, cnf_Partyhandle, DM_FULLDUP|DM_ANALOG_INTF,
                                EV_ASYNC ) == -1 )
{
        printf( "dev_Connect() failed.\n" );
         exit( 1 );
}
// Wait for Connection and Multimedia Play to
complete sr_waitevt(-1);

if( dev_Disconnect( ag_handle, EV_ASYNC ) == -1 )
{
       printf( "dev_Disconnect() failed.\n" );
       exit( 1 );
}
if( dev_Disconnect(cnf_Partyhandle, EV_ASYNC ) == -1 )
{
       printf( "dev_Disconnect() failed.\n" );
       exit( 1 );
}
```

3.4.8 Dialogic® Digital Network Interface Software Reference

There are currently no updates to this document.

3.4.9 Dialogic® Fax Software Reference

Update to the **fx_initstat( )** function in support of a new fax gateway.

Add Dialogic® HMP 3.0WIN Software to the function syntax Platform.
Add the following note to the Description:
For the DF_T38GW mode, the application *must*:

- connect the IP device to the Fax Channel using the gc_Extension model for connecting the IP data stream to the fax device (or the IPML functions directly)
- connect the PSTN device to the Fax Channel using the appropriate CT Bus API calls

before calling **fx_initstat( )**. The application also must enable the event handler to process the fax gateway completion event: TFX_T38GW. This will be the indication to the application that the fax gateway processing has completed. Add the following State value to the Parameter table:

DF_T38GW set fax channel to gateway mode

Add the following new Cautions:

Once **fx_initstate( )** is called in gateway mode, the application can not stop the fax gateway operation directly. If either the PSTN or IP calls are disconnected prior to the fax gateway session completing, the fax channel will generate the TFX_T38GW event after fax protocol timers time out.

Upon receipt of the TFX_T38GW event, the application can take what ever action it deems appropriate: complete the tear down of either call (PSTN or IP) change IP media codecs. The fax channel itself, is reset by calling **fx_initstat( )** with the desired parameter prior to the application's next fax operation.

Add the following code example (after the existing example):

```
#include <stdio.h>
#include <srllib.h>
#include <gclib.h>
#include <gcip.h>
#include <gcip_defs.h>

#include <faxlib.h>

int iptdev; /* PSTN device handles. */
int dtidev; /* PSTN device handles. */
int faxdev; /* Fax channel device handle. */

int main(int argc, char* argv[])

{
 int rc;

 .
 .
 .


/*
 * Open the channel using fx_open( ) and obtain the
 * FAX channel device handle in dev. Use dev for all
 * Fax API calls.
 */

 if( faxdev = fx_open( "dxxxB23C1", NULL )) == -1 )
  {
   /* Error opening device. */
   /* Perform system error handling */
   exit( 1 );
  }

/*
 * Establish an SRL handler for the Gateway completion event
 */

 if( sr_enbhdlr( faxdev, TFX_T38GW, GatewayComplete ) == -1 )
   {
    /* Error enabling handler */
    /* Perform system error handling */
    exit( 1 );
   }


/* Open up Global Call devices
 * - PSTN - the DTI device
 * - IP   - the IPT device
 *
 * Be sure to check for failed de
 */
```

```
 if( ( rc = gc_OpenEx( &dtidev, ":N_dtiB2T1:P_isdn", EV_SYNC, NULL ) ) != GC_SUCCESS )
   {
    /* Error opening device. */
    /* Perform system error handling */
    exit( 1 );
   }

 if( ( rc = gc_OpenEx( &iptdev, ":N_iptB1T1:M_ipmB1C1", EV_SYNC, NULL ) ) != GC_SUCCESS )
   {
    /* Error opening device. */
    /* Perform system error handling */
    exit( 1 );
  }


/* !!!!!
 *
 * Please consult the document "Global Call IP Technology
 * Guide" on  how to set up and process inbound/outbound T38 calls. Specify the fax
 * device opened using the GC API model or the IPML devconnect APIs to associate the
 * fax device w/the inbound or outbound IP call.
 *
 * Also, utilize the PSTN device to place or answer the PSTN call and use the
 * appropriate xx_Listen commands to connect this same fax device to the PSTN portion
 * of the "gateway'd" call
 *
 * Once both sides of the call have been connected, issue the fx_initstat command
*
 * !!!!!
 */

/*
 * Set the FAX state to be in V17/T28 Gateway mode
 */

if( fx_initstat( faxdev, DF_T38GW ) == -1)
  {
   /* Error on device. */
   /* Perform system error handling */
   exit( 1 );
  }

.
.
.
}


/*
 * Fax Gateway Event handler
 */

long GatewayComplete ( long hEvent )
{
 .
 .
 .

 if( sr_getevttype( hEvent ) == TFX_T38GW )
   {
    printf( "Phase E status of gateway operation: %ld\n",
    ATFX_ESTAT( sr_getevtdev( hEvent ) );
   }

 .
 .
```

.

}

Update the **ATFX_ESTAT( )** function by adding the following Gateway Error values

#define EFX_SIGNALTIMEOUT 214 - Signal timeout - no data or events received during GW session

#define EFX_DCNTIMEOUT 215 - DCN timeout - GW session almost complete but no DCN received

#define EFX_BADIPADDRESS 216 - Bad IP address - T38 subsystem did not get remote IP address - check R4 application

#define EFX_CTBUSERROR 217 - CT Bus error w/TDM portion of GW session - check R4 application

3.4.10 Dialogic® Global Call API Programming Guide

There are currently no updates to this document.

3.4.11 Dialogic® Global Call API Library Reference

Update to Chapter 2, “Function Information”

The functions **gc_SipPrack( )** and **gc_SipPrackResponse( )** should be added for SIP PRACK Handling. See Section 1.125, “SIP PRACK Handling Support (RFC 3262)”, on page 228 for more information.

Global update to the Global Call API Library Reference

Code examples

The purpose of the code examples in this document is to illustrate how Global Call functions are used. The examples are not necessarily for Dialogic® HMP-specific applications.

Functions supported in async mode only

Many of the Global Call functions are supported in asynchronous mode only when used with HMP. The async only limitation is for IP, not for calls over the PSTN. Refer to the "IP-Specific Function Information" chapter in the Global Call IP Technology Guide for any limitations or differences in function support when used with IP technology.

References to different technologies

Global Call functions can be used with many different technologies. The technologies supported in this release of Dialogic® HMP software are IP, E1/T1, ISDN, and SS7. Throughout this document, Dialogic® HMP users should ignore any references to Analog and technologies for Springware boards.

Update to **gc_util_insert_parm_val( )** (IPY00043078)

In the description for **gc_util_insert_parm_val( )**, a note should be added stating that **gc_Start( )** must be called before **gc_util_insert_parm_val( )**. Also, the code example should be replaced with the following:

```
#include <stdio.h>
include <srllib.h>
#include <gclib.h>
#include <gcerr.h>

void main( )
{
    GC_PARM_BLKP my_blkp = NULL;
    GC_PARM_DATAP my_parmp;
    GC_INFO gc_error_info; /* GlobalCall error information data
    */ int type = 1;

    /* Issue a gc_Start() call to initialize the library */
    if ( gc_Start(NULL) != GC_SUCCESS )
    {
        /* process error return as shown */
        gc_ErrorInfo( &gc_error_info );
        printf ("Error: gc_Start(), GC ErrorValue: 0x%hx - %s, CCLibID: %i - %s,
                CC ErrorValue: 0x%lx - %s\n", gc_error_info.gcValue,
                gc_error_info.gcMsg, gc_error_info.ccLibId, gc_error_info.ccLibName,
                gc_error_info.ccValue, gc_error_info.ccMsg);
        return (gc_error_info.gcValue);
    }

    /* insert parm by reference */
    if ( gc_util_insert_parm_ref( &my_blkp, GC_SET_SERVREQ, PARM_REQTYPE,
                                  sizeof( int ), &type ) != GC_SUCCESS )
    {
        /* Process error */
    }
    /* insert parm by value */
    if ( gc_util_insert_parm_val( &my_blkp, GC_SET_SERVREQ, PARM_ACK,
                                  sizeof( short ), GC_ACK ) != GC_SUCCESS )
    {
        /* Process error */
    }

    /* Now we should have a GC_PARM_BLK with 2 parameters */

   /* Following use of gc_util_next_parm retrieves the first parameter in a
    * GC_PARM_BLK, which in this case is PARM_REQTYPE
    */ my_parmp = gc_util_next_parm( my_blkp, NULL );

    /* Retrieve the next parameter after getting the first one
    */ my_parmp = gc_util_next_parm( my_blkp, my_parmp );

    /* This function finds and returns specified parameter, NULL if not found
    */ my_parmp = gc_util_find_parm( my_blkp, GC_SET_SERVREQ, PARM_ACK );

    /* After GC_PARM_BLK is no longer needed, delete the block
    */ gc_util_delete_parm_blk( my_blkp );

    /* Set my_blkp to NULL now that the block has been deleted
    */ my_blkp = NULL;

    /* Issue gc_Stop() Next */
    if (gc_Stop() != GC_SUCCESS )
    {
        /* process error return as shown */
        gc_ErrorInfo( &gc_error_info );
        printf ("Error: gc_Stop(), GC ErrorValue: 0x%hx - %s, CCLibID: %i - %s,
                CC ErrorValue: 0x%lx - %s\n",
                gc_error_info.gcValue, gc_error_info.gcMsg,
                gc_error_info.ccLibId, gc_error_info.ccLibName,
                gc_error_info.ccValue, gc_error_info.ccMsg);
    }
}
```

3.4.12 Dialogic® Global Call Analog Technology Guide

This document is added to the bookshelf to support the following analog devices:

- Dialogic® D/80PCIE-LS
- Dialogic® D/4PCIUFEQ
- Dialogic® D/4PCIU4SEQ

See Section 1.60, “Support for Dialogic® D/4PCIU Boards”, on page 143 and Section 1.67, “Support for the Dialogic® D/80PCIE-LS Media Board”, on page 157 for more information.

3.4.13 Dialogic® Global Call E1/T1 CAS/R2 Technology Guide

A new version of this document is now available on the documentation bookshelf.

3.4.14 Dialogic® Global Call IP Technology Guide

With Service Update 372, update to Section 8.2, “IP-Specific Dialogic® Global Call API Functions” (IPY00118472)

The following should be updated in “First Party Call Control (1PCC) Mode” under “gc_AcceptModifyCall( )”.

Replace the following:
To accept the changes to the dialog and media session exactly as proposed, the application calls **gc_AcceptModifyCall( )** with a NULL pointer as parmblkp.

With the following:
While the application has the choice of invoking **gc_AcceptModifyCall( )** with a NULL pointer as **parmblkp**, this does not guarantee accepting the media session exactly as proposed. See section “Accepting a SIP re-INVITE Request” for more details.

Replace the following:
An application can also formulate a specific SDP answer by inserting appropriate media session parameter elements (GCSET_CHAN_CAPABILITY / IPPARM_LOCAL_CAPABILITY) into the GC_PARM_BLK parameter block that it references in the **gc_AcceptModifyCall( )** function call.

With the following:
An application should formulate a specific SDP answer by inserting appropriate media session parameter elements (GCSET_CHAN_CAPABILITY / IPPARM_LOCAL_CAPABILITY) into the GC_PARM_BLK parameter block that it references in the **gc_AcceptModifyCall( )** function call.

With Service Update 372, update to Section 4.29, “Using SIP Transport Layer Security (TLS)”

The following should be updated in Section 4.29.1, “Overview of TLS” and Section 4.29.2, “Configuring and Enabling TLS”.

Replace the following:
http://www.openssl.org/docs/apps/ciphers.html

With the following:
http://www.openssl.org/docs/man1.0.2/apps/ciphers.html

With Service Update 328, 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, parmblkp,
GC_SINGLECALL); gc_util_delete_parm_blk(parmblkp); {
```

3.4.15 Dialogic® Global Call ISDN Technology Guide

A new version of this document is now available on the documentation bookshelf.

3.4.16 Dialogic® Global Call SS7 Technology Guide

A new version of this document is now available on the documentation bookshelf.

3.4.17 Dialogic® IP Media Library API Programming Guide and Library Reference

Update to **ipm_getLocalMediaInfo( )** function

The following note should be added:

Note: **ipm_getLocalMediaInfo( )** must be called at least once after opening an IPM device and prior to calling **ipm_StartMedia( )** to initialize media information; otherwise a failure can occur when starting a new media session.

Update to **ipm_StartMedia( )** function

The following note should be added:

Note: **ipm_getLocalMediaInfo( )** must be called at least once after opening an IPM device and prior to calling **ipm_StartMedia( )** to initialize media information; otherwise a failure can occur when starting a new media session.

3.4.18 Dialogic® Learn Mode and Tone Set File API Software Reference

This document is added to the bookshelf to support the following analog devices:

- Dialogic® D/80PCIE-LS
- Dialogic® D/4PCIUFEQ
- Dialogic® D/4PCIU4SEQ

See Section 1.60, “Support for Dialogic® D/4PCIU Boards”, on page 143 and Section 1.67, “Support for the Dialogic® D/80PCIE-LS Media Board”, on page 157 for more information.

3.4.19 Dialogic® Multimedia API Programming Guide and Library Reference

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.

3.4.20 Dialogic® SRL API Programming Guide

There are currently no updates to this document.

3.4.21 Dialogic® SRL API Library Reference

Update to **ATDV_ERRMSGP( )** function

This function does not support multimedia (mm) devices. Use **mm_ErrorInfo( )** to get error information for a multimedia device.

Update to **ATDV_LASTERR( )** function

This function does not support multimedia (mm) devices. Use **mm_ErrorInfo( )** to get error information for a multimedia device.

3.4.22 Dialogic® Station Side Interface API Library Reference

There are currently no updates to this document.

3.4.23 Dialogic® Voice API Library Reference

In Service Update 307, update to the DX_IOTT data structure reference page. (IPY00093803)

Add the following note to the io_bufp field description:

Note: In asynchronous mode, the io_bufp (base memory address) must remain in scope for the duration of the function.

3.4.24 Dialogic® Voice API Programming Guide

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.

3.5 Supported Applications Documentation Updates

This section contains updates to the following documents:

- Dialogic® MSML Media Server Software User's Guide

3.5.1 Dialogic® MSML Media Server Software User's Guide

Removed document from the bookshelf in Service Update 328 since MSML Media Server Software support is no longer available. MSML support is now available under Dialogic® PowerMedia™ Extended Media Server (XMS).

3.6 Demonstration Software Documentation Updates

This section contains updates to the following documents:

- Dialogic® Audio Conferencing API Demo Guide
- Dialogic® CSP API Demo Guide
- Dialogic® Global Call API Demo Guide
- Dialogic® IP Media Server Demo Guide
- Dialogic® IP Gateway (Global Call) Demo Guide
- Dialogic® Multimedia Demo Guide
- Dialogic® Ansrmt Voice Demo Online Help
- Dialogic® Xaansr Voice Demo Online Help
- Dialogic® VoiceDemo Online Help

3.6.1 Dialogic® Audio Conferencing API Demo Guide

There are currently no updates to this document.

3.6.2 Dialogic® CSP API Demo Guide

There are currently no updates to this document.

3.6.3 Dialogic® Global Call API Demo Guide

There are currently no updates to this document.

3.6.4 Dialogic® IP Media Server Demo Guide

There are currently no updates to this document.

3.6.5 Dialogic® IP Gateway (Global Call) Demo Guide

There are currently no updates to this document.

3.6.6 Dialogic® Multimedia Demo Guide

There are currently no updates to this document.

3.6.7 Dialogic® Ansrmt Voice Demo Online Help

There currently are no changes to this document.

3.6.8 Dialogic® Xaansr Voice Demo Online Help

The following procedure replaces the existing procedure in the **"Running the Demo"** topic (IPY00010908 = PTR 36541):

1. Select Start from the Action menu. The voice channels configured in the Options window are displayed.
2. Dial an extension number configured for one of the voice channels, if available.
3. As the voice prompt plays, write down the 4-digit access code that is displayed next to the voice channel in use. The 4-digit access code takes the form n234, where n is the last digit of the voice channel number.
4. After the voice prompt ends, record a brief message and hang up the telephone.

Note: You can stop the recording at any time by pressing any digit.

5. Redial the same extension number, if applicable, as in step 2.
6. During the voice prompt, enter the 4-digit access code using the telephone keypad. The system plays the previously recorded message.
7. Hang up the telephone.

3.6.9 Dialogic® VoiceDemo Online Help

There currently are no changes to this document.