Dialogic®

Dialogic® Host Media Processing Software Release 1.3WIN

Release Update

February 19, 2008

05-2421-028

Publication Date: February 19, 2008

Document Number: 05-2421-028

About This Publication

This section contains information about the following topics:

Purpose

This Release Update addresses issues associated with Dialogic® Host Media Processing 1.3WIN Software Release. In addition to summarizing issues that were known as of this release, the Release Update will continue to be updated to serve as the primary mechanism for communicating new issues that arise after the release date.

Intended Audience

This Release Update is intended for all users of Dialogic® Host Media Processing 1.3WIN Software Release.

How to Use This Publication

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

- Document Revision History: This section summarizes the ongoing changes and additions that are made to this Release Update after its original release. This section is organized by document revision and document section.
- Post Release Developments: This section describes significant changes to the release subsequent to the general availability release date. For example, the new features provided in the Service Update are described in this section.
- Release Issues: This section lists issues that may affect the Dialogic® HMP 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 Dialogic® HMP Software release documentation set that could not be made to the documents prior to the release. The updates are organized by documentation category and by individual document.

Related Information

See the following for additional information:

- For information about the products and features supported in this release, see the *Dialogic® Host Media Processing Software Release 1.3WIN Release Guide*, which is included as part of the documentation bookshelf for the release.
- For further information on issues that have an associated defect number, you may use the Defect Tracking tool at *http://membersresource.dialogic.com/defects/*. When you select this link, you will be asked to either LOGIN or JOIN.
- *http://www.dialogic.com/support* (for Dialogic technical support)
- *http://www.dialogic.com/* (for Dialogic product information)

Document Revision History

This revision history summarizes the changes made in each published version of the Release Update for Dialogic® Host Media Processing Software Release 1.3WIN, which is a document that is subject to updates during the lifetime of the release.

Document Rev 28, published February 14, 2008

Update for Service Update 121.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00041296, IPY00041583, IPY00041789, IPY00041876.

Document Rev 27, published January 10, 2008

Update for Service Update 120.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00039677, IPY00039707, IPY00039823, IPY00039847, IPY00039965, IPY00040440, IPY00040743, IPY00041048, IPY00041118, IPY00041300.

Document Rev 25, published September 10, 2007

Update for Service Update 118.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00038945, IPY00038992, IPY00039036, IPY00039225, IPY00039315, IPY00039333, IPY00039401, IPY00039409, IPY00039451, IPY00039505, IPY00039564, IPY00039639.

Document Rev 24, published July 3, 2007

Update for Service Update 113.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00038339, IPY00038708, IPY00038732, IPY00038827, IPY00038848, IPY00038868.

Document Rev 23, published June 5, 2007

Update for Service Update 111.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00038218, IPY00038342.

Document Rev 22, published May 22, 2007

Update for Service Update 109.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00038060, IPY00038150, IPY00038240, IPY00038343, IPY00038365.

In the Documentation Updates chapter:

- Added updates to Dialogic® Host Media Processing Software Release 1.3WIN Software Installation Guide.

Document Rev 21, published May 16, 2007

Update for Service Update 108.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00036779, IPY00037359, IPY00037603, IPY00037699.

Document Rev 20, published April 10, 2007

Update for Service Update 107.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00037704, IPY00037737, IPY00037778, IPY00037825.

Document Rev 19, published April 2, 2007

Update for Service Update 106.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00036658, IPY00037541, IPY00037613, IPY00037708.

Document Rev 18, published March 7, 2007

Update for Service Update 102.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00037172, IPY00037252, IPY00037333, IPY00037360.

Document Rev 17, published February 22, 2007

Update for Service Update 101.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00036292, IPY00036930. Also, added IPY00035875 (resolved in Service Update 96).

Document Rev 16, published February 15, 2007

Update for Service Update 100.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00035642, IPY00036250. Also, added IPY00034499 (resolved in Service Update 89).

Document Rev 15, published January 19, 2007

Update for Service Update 96.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00032865, IPY00036052, IPY00036265, IPY00036266.

In the Documentation Updates chapter:

- Added updates to Native Configuration Manager API for Windows Operating Systems Library Reference.

Document Rev 14, published December 12, 2006

Update for Service Update 91.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00035684, IPY00035822.
- Added the following Known Defect to the Issues Table: IPY00035875.

Document Rev 13, published November 17, 2006

Update for Service Update 89.

In the Post Release Developments chapter:

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

In the Release Issues chapter:

- Changed the following Resolved Defects to SU 86 in the Issues Table: IPY00032375, IPY00033575, IPY00033733, IPY00033738, IPY00033770.
- Added the following Resolved Defects to the Issues Table: IPY00034526, IPY00034686, IPY00034777, IPY00035350, IPY00035473, IPY00035517, IPY00035613.

Document Rev 12, published October 24, 2006

Update for Service Update 86.

In the Post Release Developments chapter:

- Added information about support for Support for Advanced Network Services.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00032375, IPY00033371, IPY00033575, IPY00033597, IPY00033733, IPY00033738, IPY00033770, IPY00034210, IPY00034215, IPY00034313.

In the Documentation Updates chapter:

- Added an update to Section 9.4, “Stopping HMP”, in the *Dialogic Host Media Processing Software Release 1.3WIN Administration Guide.*
- Updated the descriptions of the **NCM_StopDlgSrv( )** and **NCM_StopSystem( )** functions in the *Native Configuration Manager API for Windows Operating Systems Library Reference*.

Document Rev 11, published July 11, 2006

Update for Service Update 78.

In the Post Release Developments chapter:

- Added information about the `its_sysinfo` utility.

In the Release Issues chapter:

- Added the following Resolved Defects to the Issues Table: IPY00032378, IPY00032459, IPY00033102, IPY00033322.

In the Documentation Updates chapter:

- Changed the description of the MSG_ACTID parameter in the *Audio Conferencing API for Linux and Windows Operating Systems Library Reference* and the *Audio Conferencing API for Linux and Windows Operating Systems Programming Guide*.
- Changed the "Background Music" chapter in the *Audio Conferencing API for Linux and Windows Operating Systems Programming Guide*.
- Added a new Caution to the description of the **dx_listenEx()** and **dx_unlistenEx()** functions in the *Voice API for Host Media Processing Library Reference*.

Document Rev 10, published April 24, 2006

Update for Service Update 76.

In the Release Issues chapter:

- Added the following Resolved PTRs to the Issues Table: 36787, 36842, 36860, 36867, 36869.

In the Documentation Updates chapter:

- In the *Global Call for Host Media Processing Technology Guide,* Section 7.3.16, "gc_MakeCall() Variances for IP", the last paragraph before Section 7.3.16.1 (page 312) was updated.
- Added information regarding the *Global Call for Host Media Processing Technology Guide* about changes to the three data fields for the IP_H221NONSTANDARD data structure.

Document Rev 09, published March 17, 2006

Update for Service Update 73.

In the Release Issues chapter:

- Added the following Resolved PTR to the Issues Table: 36793

In the Documentation Updates chapter:

- Added information regarding the *Global Call for Host Media Processing Technology Guide* about changes to the three data fields for the IP_H221NONSTANDARD data structure.

Document Rev 08, published February 24, 2006

Update for Service Update 72.

In the Post Release Developments chapter:

- Added information about the SIP Re-INVITE feature.
- Added information about preserving user configuration data.

In the Release Issues chapter:

- Added the following Resolved PTRs to the Issues Table: 36080, 36317, 36349, 36372, 36437, 36643, 36647, 36669, 36677

Document Rev 07, published December 15, 2005

Update for Service Update 67.

In the Post Release Developments chapter:

- Added information about the Distinguishing Between 180 and 183 Responses feature.

In the Release Issues chapter:

- Added the following Resolved PTRs to the Issues Table: 36523
- Removed the “APIC Timer and HMP” information from the Compatibility Notes section.

In the Documentation Updates chapter:

- Added information about SIP Provisional (1xx) Responses to the Global Call IP for Host Media Processing Technology Guide.
- Added a note to the Gatekeeper Registration Failure (H.323) section in the Global Call IP for Host Media Processing Technology Guide.

Document Rev 06, published November 23, 2005

Update for Service Update 65.

In the Release Issues chapter:

- Added the following Resolved PTRs to the Issues Table: 35660, 35692, 35938, 36027, 36082, 36280, 36419

Document Rev 05, published October 31, 2005

Additional update for Service Update 61.

In the Post Release Developments chapter:

- Added information about the Fast Start Coder feature.

In the Documentation Updates chapter:

- Added information about the Fast Start Coder and the IP_VIRTBOARD data structure to the Global Call IP for Host Media Processing Technology Guide.

Document Rev 04, published October 11, 2005

Updated for Service Update 61.

In the Release Issues chapter:

- Added the following Resolved PTRs to the Issues Table: 34936, 35596, 35693, 35872, 35928, 35979
- Moved PTR 34952 from the Known to the Resolved section of the Issues Table.
- Moved PTR 32740 from the Known to the Known (permanent) section of the Issues Table.

In the Documentation Updates chapter:

- Added documentation corrections for the IP Media Library API for Host Media Processing Library Reference
- Added documentation corrections for the IP Media Library API for Host Media Processing Programming Guide

Document Rev 03, published August 12, 2005

Updated for Service Update 54.

In the Release Issues chapter:

- Added the following Resolved PTRs: 35434, 35520, 35545, 35585
- Moved the information from the Restrictions and Limitations section into the Issues Table as Known (permanent) issue types.
- Added Issue Type and SU No. columns to Issues Table.

Document Rev 02, published June 30, 2005

Updated for Service Update 52.

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

- Support for 120 T.38 Fax channels
- Special Information Tone (SIT) frequency detection enhancements
- User configurable RTCP reporting interval
- User configurable UDP Port range
- Support for G.726 IP Coder
- Added ATEC_TERMMSK() function for CSP
- Comfort noise generation in conferencing

The Documentation Updates section contains the following changes:

- In the Release Documentation section, for the Release Guide, added information regarding support for 120 fax channels.

- In the Installation Documentation section, for the Installation Guide, added information about the Installation Guide being revised to include new HMP software installation procedures.
- In the Development Software Documentation section, for the IP Media Library API for Host Media Processing Library Reference, added information about support for RTCP Reporting Interval, UDP Port Range, and the G.726 Coder.
- In the Development Software Documentation section, for the Voice API for Host Media Processing Library Reference, added information about the ATDX_CRTNID() function to support the SIT Tones.
- In the Development Software Documentation section, for the Voice API for Host Media Processing Programming Guide, revised information about the SIT Frequency Detection.
- In the Demonstration Software Documentation section, for the Audio Conferencing API for Host Media Processing Demo Guide, changed the information in Step 3 of the "Starting the Demo" procedure.
- In the Demonstration Software Documentation section, for the Continuous Speech Processing API for Host Media Processing on Windows Demo Guide, changed the information about the directory in which the demo is located.

Document Rev 01, published May 2005

Initial Version of document.

Post Release Developments 1

This section describes significant changes to the Dialogic® Host Media Processing Software Release 1.3WIN subsequent to the general availability release.

1.1 Service Update for Dialogic® HMP Software Release 1.3WIN

This Service Update for Dialogic® Host Media Processing Software Release 1.3WIN is now available. Service Updates provide fixes to known problems, and may also introduce new functionality. New versions of the Service Update will be released periodically.

For information about installing this Service Update, refer to the *Dialogic® Host Media Processing Release 1.3WIN Installation Guide*. A new version of the Installation Guide has been added to the documentation bookshelf since the initial release of Dialogic® Host Media Processing Software Release 1.3WIN to describe the new installation procedures.

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

With the Service Update, 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.

1.2.1 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.

1.2.2 New Parameter IDs

An existing parameter set ID (IPSET_CALLINFO) and new parameter IDs support the CPN information as shown in the table below:

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 char   CPN_NPI;   /* Numbering plan identification */
typedef unsigned char   CPN_SI;     /* Screening Indicator      */
typedef unsigned char   CPN_PI;     /* Presentation Indicator */
```

1.2.3 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 the 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;
```

1.2.3.1 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.

1.2.4 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() */
}
```

1.2.5 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 *Global Call IP for Host Media Processing 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.

1.2.6 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);
   }
}
```

1.2.7 Documentation

The online bookshelf provided with Dialogic® HMP 1.3WIN contains information about all release features including features for application development, configuration, administration, and diagnostics.

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

- *Global Call IP for Host Media Processing Technology Guide*
- *Global Call API for Linux and Windows Operating Systems Library Reference*

For more information about the IP Media API, see the following documents:

- *IP Media Library API for Host Media Processing Programming Guide*
- *IP Media Library API for Host Media Processing Library Reference*

1.3 Support for Advanced Network Services

This Service Update verifies that Dialogic® HMP 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 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 please see:

http://www.dialogic.com/support/network/sb/CS-009747.htm

1.3.1 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

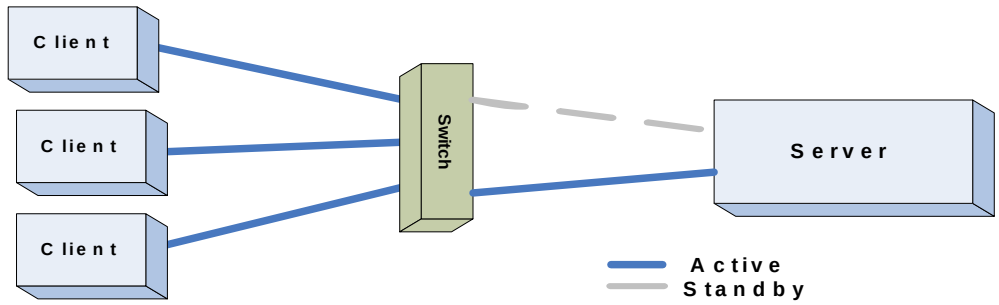
1.3.2 Team Roles

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.

1.3.3 Fault Tolerance

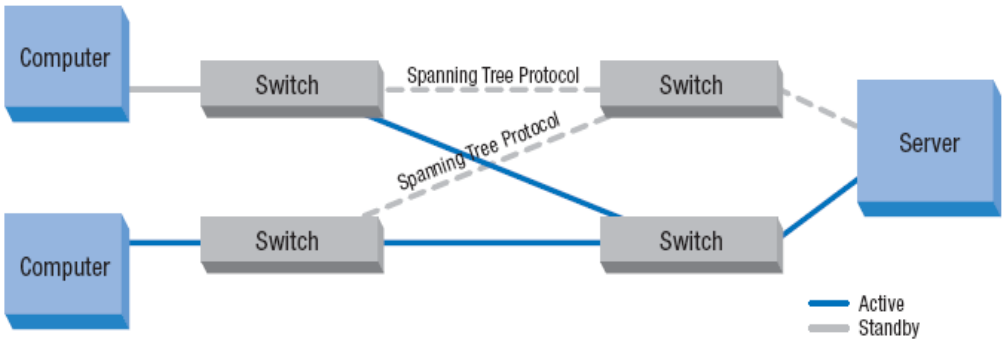
Failure protection is afforded through two types of fault tolerance, *Adapter Fault Tolerance* (AFT) and *Switch Fault Tolerance* (SFT). AFT is the default mode when a team is created. When AFT is operating, the secondary adapter automatically carries traffic when there is a connection failure of any 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.

1.3.4 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 in Host Media Processing Technology Guide*.

1.4 Command Line Interface to the its_sysinfo Utility

This feature provides a command line interface for invoking `its_sysinfo`, a standalone utility that gathers pertinent MCPD and non-MCPD 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*.

1.4.1 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*.

1.4.2 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 administration guide for the Dialogic® HMP Software 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 administration guide for the Dialogic® HMP Software 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
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.*

Table 1. Log Files Archived by its_sysinfo (Continued)

Log Files	Windows	Linux
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

1.5 Receiving and Handling SIP re-INVITE Requests

RFC 3261 specifies that User Agents must be able to send and respond to additional INVITE requests after a dialog has been established to allow modification of the dialog or the media session. Additional INVITE requests in an existing dialog are known as re-INVITE requests to distinguish them from an initial INVITE request that would initiate a new dialog. Re-INVITE requests contain the same Call-ID, To, and From headers as the original INVITE request that established the dialog. Either party in a dialog can issue a re-INVITE, and only one re-INVITE can be pending at any given time.

The re-INVITE method is a general purpose mechanism that can be used to modify or update most of the properties of a dialog (the most notable exceptions being the header fields that are used to identify the message as a subsequent INVITE rather than a new INVITE) or the associated media session. The Global Call implementation of re-INVITE in the Dialogic® Host Media Processing Service Update code line supports the following capabilities which are described in the following subsections:

- specifying, changing, or refreshing header field values or parameters for the existing dialog; for example, refreshing Contact information that is expiring
- changing the direction of the streaming; for example, changing from half-duplex to full-duplex streaming
- changing the RTP port of the remote endpoint

  Note: Global Call does not provide a mechanism for initiating an RTP port change, but Global Call applications can receive and act on remote port change requests received from non-Global Call applications.

- changing between audio and T.38 fax modes

  Note: The previously implemented automatic and manual modes for audio/T.38 switching (as documented in the "T.38 Fax Server" section of the *Global Call IP for Host Media Processing Technology Guide*) have used re-INVITE "under the hood" when using the SIP protocol. The new capability for applications to have access to re-INVITE requests also obligates the application to handle switching between audio and T.38 fax modes manually and explicitly.

The ability to sending and receiving SIP re-INVITE requests is provided by three IP-specific Global Call APIs, which are documented in detail elsewhere in this document, and six IP-specific event types. The new APIs are:

- **gc_AcceptModifyCall( )**
- **gc_RejectModifyCall( )**
- **gc_ReqModifyCall( )**

1.5.1 Enabling Application Access to re-INVITE Requests

In order to have access to SIP re-INVITE requests, applications must set a specific parameter value using the Global Call **gc_SetConfigData( )** function. To enable the three **gc_xxxModifyMedia( )** APIs and the GCEV_REQ_MODIFY_CALL event type that is used to notify applications of re-INVITE requests, the application must include the following parameter element in the GC_PARM_BLK that it passes to the **gc_SetConfigData( )** function:

IPSET_CONFIG
  IPSPARM_OPERATING_MODE
    - value = IP_T38_MANUAL_MODIFY_MODE

Unless this parameter value is set, any attempt to call one of the **gc_xxxModifyMedia( )** functions will fail with an IPERR_BAD_PARM error code.

1.5.2 Receiving a SIP re-INVITE Request

REF3261 specifies that either party in a SIP dialog can initiate a re-INVITE transaction, so Global Call applications must be able to receive and handle incoming re-INVITE requests.

When the IP Call Control Library receives a re-INVITE request, the library first examines the request to determine whether it specifies media properties that are acceptable by the local endpoint. If the received re-INVITE request specifies media capabilities that are not supported by the local system, the call control library automatically sends a 488 Not Acceptable Here response to the requesting party and generates a GCEV_REQ_MODIFY_UNSUPPORTED event to the application. This unsolicited event contains a CCLIB cause code of IPEC_SIPReasonStatus488NotAcceptableHere. This event is sent for informational purposes only; the library has already sent the appropriate response to the remote UA, so the local application does not need to take any action upon receiving this informational event.

If the received re-INVITE request contains an SDP offer that specifies media capabilities that *are* supported by the local media device (or if it does not contain an SDP offer), the call control library automatically sends a 100 Trying response to the requester and generates an unsolicited GCEV_REQ_MODIFY_CALL event to notify the application.

1.5.2.1 Retrieving Dialog and Media Session Properties

The METAEVENT associated with the GCEV_REQ_MODIFY_CALL event contains a pointer to a GC_PARM_BLK structure that the library has populated with the following information from the re-INVITE request:

- parameter elements for any SIP header fields that the application has registered to receive (as described in the "Registering SIP Header Fields to be Retrieved" section of the *Global Call IP Technology Guide*)
- one or more parameter elements that contain media session properties that were proposed in the SDP offer
- a parameter element that contains the remote RTP transport address from the SDP offer

The parameter elements associated with the Call-ID, To, and From headers will contain the same values that were used in the original INVITE request that established the dialog. All other header fields and parameters have potentially been changed, and it is the application's responsibility to parse and compare the values if appropriate. The header fields that the application has registered to receive are reported in parameter elements of the following type:

IPSET_SIP_MSGINFO
    IPPARM_SIP_HDR
        - value = complete header string, including name, value, and any parameters

If the re-INVITE request contains an SDP offer, the media capabilities proposed in the offer may or may not match the properties of the current media session. It is the application's responsibility to analyze the media properties proposed in the SDP offer, to determine whether the properties are different from the current session properties, and to decide whether the proposed change (if any) is acceptable.

The GC_PARM_BLOCK that is associated with the GCEV_REQ_MODIFY_CALL event may contain any number of parameter elements which identify the supported media properties that were proposed in the request. Each proposed media capability is handled as a parameter element of the following type:

GCSET_CHAN_CAPABILITY
    IPPARM_LOCAL_CAPABILITY
        - value = IP_CAPABILITY data structure

The number of these parameter elements depends on the specifics of what change the re-INVITE is requesting:

- If the SDP offer in the re-INVITE is proposing a full-duplex media session, there will be a pair of GCSET_CHAN_CAPABILITY/IPPARM_LOCAL_CAPABILITY parameter elements for each proposed media capability that is supported on the local platform,

one element for each direction. Within each parameter pair, all fields of the of the IP_CAPABILITY structure will be the same except for the direction fields, one of which will be IP_CAP_DIR_LCLRECEIVE and the other IP_CAP_DIR_LCLTRANSMIT.

- If the SDP offer in the re-INVITE is proposing a half-duplex media session, there may be only a single GCSET_CHAN_CAPABILITY/ IPPARM_LOCAL_CAPABILITY element in the parameter block, although multiple elements are possible if multiple coders are being proposed. Within each parameter element, the IP_CAPABILITY.direction field will be either IP_CAP_DIR_LCLRECVONLY or IP_CAP_DIR_LCLSENDONLY.
- If the SDP offer in the re-INVITE is seeking to place the call on hold, there may be only a single GCSET_CHAN_CAPABILITY/ IPPARM_LOCAL_CAPABILITY element in the parameter block, although multiple elements are possible. When the re-INVITE is requesting to place the call on hold, the IP_CAPABILITY.direction field will be set to either IP_CAP_DIR_LCLRTPINACTIVE or IP_CAP_DIR_LCLRTPRTCPINACTIVE.
- If the SDP offer in the re-INVITE is seeking to change the session from audio mode to T.38 fax mode, there will usually be only a single GCSET_CHAN_CAPABILITY/ IPPARM_LOCAL_CAPABILITY element in the parameter block.

Finally, the GC_PARM_BLK will include a parameter element that contains the remote RTP transport address, which may be the same as the existing address or may be different. It is the application's responsibility to compare the address to determine whether it is different and whether the proposed change is acceptable.

The RTP transport address is handled as a parameter element of the following type:

IPSET_RTP_ADDRESS
IPPARM_REMOTE
- value = RTP_ADDR data structure

There will always be at least one of these parameter elements if the re-INVITE request contains an SDP offer (which is the typical case for re-INVITE requests); multiple elements will exist when the SDP offer contains more than one RTP address in multiple "m=" lines.

Note: SDP does not explicitly communicate RTCP port addresses, but these can be inferred from RTP addresses by means of the "plus one" offset convention.

1.5.2.2 Determining Acceptability of a re-INVITE Request

When an application retrieves and analyzes the dialog and media session properties that were contained in a re-INVITE request, it must take into account the media platform's abilities to change the properties of existing sessions. Changes in dialog properties (e.g., new or updated header fields in the re-INVITE request) have no platform dependency and can always be accepted at the application's discretion.

In the Dialogic® Host Media Processing Software Release 1.3WIN Service Update, an application can accept at its discretion a re-INVITE request that is proposing the following types of change in the media session:

- Changing the RTP address of the remote endpoint

- Changing the direction property of the media session, for example from half-duplex to full duplex
- Switching between audio and T.38 fax modes

An application must reject any re-INVITE request that is proposing any of the following changes in the media session:

- Any change in the coder properties. An application can only accept a re-INVITE request if the IP_CAPABILITY structures retrieved from the GCEV_REQ_MODIFY_CALL event contain coder properties that exactly match the coder properties of the existing session.

1.5.3 Responding to a re-INVITE Request

After an application has received an unsolicited GCEV_REQ_MODIFY_CALL event signaling a re-INVITE request, and has retrieved and analyzed the parameter elements from the GC_PARM_BLK associated with the METAEVENT, it is able to accept or reject the proposed change by calling the appropriate Global Call API.

1.5.3.1 Rejecting a re-INVITE Request

If an application determines that the received re-INVITE request is attempting to change the coder properties of the current session, or if the application decides that it does not want to accept the proposed change for some application-specific reason, it simply calls **gc_RejectModifyCall( )** to send a final response message with the specified 3xx–6xx reason code. The specific reason code to send is specified using the appropriate IPEC_SIPReasonStatus… defines as defined in *gcip_defs.h* and documented in Chapter 10 of the *Global Call IP for Host Media Processing Technology Guide*.

When the remote UA acknowledges the rejection response, the library generates a GCEV_REJECT_MODIFY_CALL completion event and the session continues unchanged, just as if a re-INVITE request had never been issued.

If the transmission of the rejection message fails for some reason, the library generates a GCEV_REJECT_MODIFY_CALL_FAIL event. In the case of such a failure, the re-INVITE transaction is still in progress, and the application should make another attempt to respond via **gc_RejectModifyCall( )** or **gc_AcceptModifyCall( )**.

1.5.3.2 Accepting a SIP re-INVITE Request

When an application determines that the changes to the existing dialog or media session that were proposed in a received re-INVITE request are acceptable, it calls the **gc_AcceptModifyCall( )** function to send a 200 OK response. But because the SDP offer contained in a re-INVITE request may contain more than one session proposal, the application has the opportunity to specify which proposal it wishes to accept.

If the application calls **gc_AcceptModifyCall( )** with a NULL pointer as the **parmblkp** parameter, the library uses the codec preferences that were used in the original INVITE dialog to formulate the SDP response. In this case, if the SDP offer in the re-INVITE

proposed a codec that the application did not indicate as acceptable in the original INVITE dialog, the library treats the situation as a rejection of the call modification request. In this case, a 488 Not Acceptable Here response is sent to the remote party to terminate the re-INVITE dialog, and a GCEV_REJECT_MODIFY_CALL event is sent to the application.

To formulate a specific SDP answer, an application inserts appropriate media capability parameter elements into the GC_PARM_BLK parameter block that it passes to **gc_AcceptModifyCall( )**. Each parameter element is of the following format:

GCSET_CHAN_CAPABILITY
    IPPARM_LOCAL_CAPABILITY
    - value = IP_CAPABILITY data structure

A full-duplex connection requires two such parameter elements, one for each direction. A half-duplex connection requires one parameter element with the direction field of the IP_CAPABILITY structure set appropriately. To accept a request to transition to T.38 mode, the application must insert a parameter element with an IP_CAPABILITY structure that contains the appropriate codec and direction values. If the application does not specify a capability that matches a proposed capability in the re-INVITE's SDP offer, the library treats the situation as a rejection of the modification request, sends a 488 Not Acceptable Here response to the remote party to terminate the re-INVITE dialog, and generates a GCEV_REJECT_MODIFY_CALL to the application.

When the remote UA acknowledges the 200 OK response, the library generates a GCEV_ACCEPT_MODIFY_CALL event to notify the application that the re-invite transaction has completed successfully. If the transmission of the 200 OK message fails for some reason, the library generates a GCEV_ACCEPT_MODIFY_CALL_FAIL event. In the case of such a failure, the re-INVITE transaction is still in progress, and the application should make another attempt to respond to the re-INVITE request.

1.5.4 Sending a SIP re-INVITE Request

To send a SIP re-INVITE request, an application first constructs a GC_PARM_BLK that contains parameter elements for the dialog and/or media session properties that it wishes to change. Then the application passes that parameter block in a call to the **gc_ReqModifyCall( )** function. Note that there can be only a single re-INVITE transaction pending at any given time; if there is a re-INVITE already pending (initiated by either endpoint), calling **gc_ReqModifyCall( )** produces an error result.

If a re-INVITE request sent by a Global Call application times out, the library generates a GCEV_MODIFY_CALL_FAIL event to the application with a cause value of IPEC_SIPReasonStatus408RequestTimeout. In compliance with RFC 3261 the 408 time-out condition causes the library to send BYE to terminate the dialog, and it notifies the application of this termination with a GCEV_DISCONNECTED event.

The GC_PARM_BLK that the application constructs may contain two types of parameter elements. There may be one or more elements to specify SIP header fields to change or update the properties of the dialog, such as the Contact or Via information, and there may be one or more elements to specify media capabilities to be included in the SDP offer within the re-INVITE request.

1.5.4.1 Inserting SIP Header Fields in a re-INVITE Request

SIP header fields to be sent in a re-INVITE are specified using the standard technique described in the *Global Call IP for Host Media Processing Technology Guide*. The application simply inserts parameter elements of the following type into the GC_PARM_BLK it passes to **gc_ReqModifyCall( )**:

IPSET_SIP_MSGINFO
    IPPARM_SIP_HDR
        - value = complete header string, including header field name

The header fields are inserted in the SIP message in the same order in which they are inserted into the GC_PARM_BLK. See Section 4.6.5, "Setting SIP Header Fields for Outbound Messages" in the *Global Call IP for Host Media Processing Technology Guide* for more details on sending SIP headers.

When setting header fields in SIP re-INVITE requests, there are some restrictions to note:

- Request-URI and Call-ID cannot be set because they are used to identify the request as a subsequent INVITE request (re-INVITE).
- CSeq cannot be set.
- In the From and To headers, the URI and Tag cannot be changed because they are used to identify the request as a re-INVITE. In both cases, the Display and some of the URI parameters *can* be changed, but the application must ensure that the URI and Tag substrings that it includes when specifying the header string are identical to those in the original INVITE.
- Max-Forwards can be set by the application, but if the application does not set it the library automatically sets it to 70.
- Contact and Via can be set by the application, but if the application does not provide them the library automatically inserts the corresponding header field from the last INVITE or 2xx response that the application sent in the current dialog.

All other header fields, including proprietary headers, can be set without restriction.

1.5.4.2 Specifying Media Session Properties in a SIP re-INVITE

If an application wishes to change any media session properties via a re-INVITE request, it must insert appropriate media capability parameter elements into the GC_PARM_BLK that it passes to **gc_ReqModifyCall( )**. If there is no need to change media session properties (for example, when using re-INVITE simply to refresh the dialog Contact information), the application can opt to not include media session property parameter elements in the GC_PARM_BLK, in which case the library will use the last SDP answer when it constructs the re-INVITE.

These parameter elements for media capabilities are of the form:

GCSET_CHAN_CAPABILITY
    IPPARM_LOCAL_CAPABILITY
        - value = IP_CAPABILITY structure

For a full-duplex media session, the application must insert these capability parameter elements in pairs, one for transmit (IP_CAPABILITY.direction = IP_CAP_DIR_LCLTRANSMIT) and one for receive (IP_CAPABILITY.direction = IP_CAP_DIR_LCLRECEIVE). Because the Dialogic® HMP 1.3WIN Service Update does not support changing coder properties for an active session, there can only be a single pair of capability parameter elements, and they must specify the existing audio coder properties for the session.

For a half-duplex media session, the application inserts a single parameter element with the IP_CAPABILITY.direction field set to either IP_CAP_DIR_LCLTXONLY or IP_CAP_DIR_LCLRXONLY. The coder properties must match the established audio coder properties for the session.

When requesting the remote endpoint to switch to T.38 mode, the application inserts only a single parameter element with the following fields of the IP_CAPABILITY structure set to the indicated values:

- capability = GCCAP_DATA_t38UDPFax
- type = GCCAPTYPE_RDATA
- direction = IP_CAP_DIR_LCLRXTX

In each case, the IP_CAPABILITY structure must be fully specified. If only one property is being changed (for example, only changing the direction), the remaining fields of the structure must contain the current values for each of the other capability properties.

1.5.5 Canceling a Pending re-INVITE Request

If an application wishes to cancel a pending re-INVITE request, it first inserts a special parameter element into a GC_PARM BLK, then passes that parameter block to **gc_ReqModifyCall( )**.

The parameter element used to cancel a pending re-INVITE is:

IPSET_MSG_SIP
    IPPARM_SIP_METHOD
    - value = IP_MSGTYPE_SIP_CANCEL

No other parameter elements can be present in the GC_PARM_BLK when canceling a re-INVITE request.

1.5.6 New Defines to Support re-INVITE

To support re-INVITE functionality, several new values have been defined in the IP Call Control Library.

1.5.6.1 New IP_CAPABILITY.direction Defines

The supported values for the direction field of the IP_CAPABILITY data structure are described as follows:

direction

Identifies the direction and state of the stream that the media attributes in this structure apply to. Possible values are:

- IP_CAP_DIR_LCLRECEIVE – Capabilities specified in the structure refer to receive direction of a full duplex media session.
- IP_CAP_DIR_LCLRECVONLY – Capabilities refer to a half-duplex, receive-only media session.
- IP_CAP_DIR_LCLSENDONLY – Capabilities refer to a half-duplex, send-only media session.
- IP_CAP_DIR_LCLTRANSMIT – Capabilities specified in the structure refer to transmit direction of a full duplex media session.
- IP_CAP_DIR_LCLTXRX – Capabilities specified in the structure refer to both transmit and receive directions of a symmetrical full duplex media session. Supported for T.38 only.
- IP_CAP_DIR_LCLRTPINACTIVE – Capabilities refer to a media session that has been put on hold but with RTCP still active. RTP streaming is temporarily disabled until direction value is changed again. This value is only valid when using SIP, and only when sending or responding to a re-INVITE request.
- IP_CAP_DIR_LCLRTPRTCPINACTIVE – Capabilities refer to a media session that has been put on hold with RTCP as well as RTP inactive. Both RTP and RTCP streaming are disabled until direction value is changed again. Both RTP and RTCP ports may be new once streaming is re-enabled. This value is only valid when using SIP, and only when sending or responding to a re-INVITE request.

1.5.6.2 New Value for IPSET_CONFIG / IPPARM_OPERATING_MODE

A third value has been defined for the IPSET_CONFIG / IPPARM_OPERATING_MODE parameter to enable application access to SIP re-INVITE requests. The updated descriptions of the defined values are as follows:

IP_T38_AUTOMATIC_MODE

Default mode. A request to transition to or from T.38 fax gateway mode is handled automatically without application involvement. This mode cannot be used when the application requires access to SIP re-INVITE requests.

IP_T38_MANUAL_MODE

A request to transition to or from T.38 fax server mode is reported as a GCEV_EXTENSION event with an IPSET_SWITCH_CODEC parameter. This mode cannot be used when the application requires access to SIP re-INVITE requests.

IP_T38_MODIFY_MODE

Enables application access to SIP re_INVITE request. In SIP, a re-INVITE requesting transition to or from T.38 fax mode is reported as a GCEV_REQ_MODIFY_CALL event. In H.323, a request to transition to or from T.38 fax server mode is reported as a GCEV_EXTENSION event with an IPSET_SWITCH_CODEC parameter.

gc_AcceptModifyCall( )

Name: int gc_AcceptModifyCall (crn, parmblkp, mode)

Inputs:

CRN crn	• call reference number of call targeted for modification
GC_PARM_BLK *parmblkp	• pointer to GC_PARM_BLK which contains attributes of call which are being accepted (optional)
unsigned long mode	• completion mode (EV_ASYNC)

Returns: 0 if successful
<0 if unsuccessful

Includes: gclib.h

Category: Call Modification

Mode: Asynchronous only

Description

The **gc_AcceptModifyCall( )** function is used to accept a request from the network or remote party to change one or more attributes of the current SIP dialog (call).

This function initiates a 200 OK response code to an incoming re-INVITE request (an INVITE request on an established call), which has been indicated to the application as an unsolicited GCEV_REQ_MODIFY_CALL event on the respective call object. The metadata associated with this event references a GC_PARM_BLK which contains parameter elements that communicate the contents of the re-INVITE request to the application. The 200 OK response sent by this function indicates acceptance of the change(s) proposed in the re_INVITE request, which may be changes to one or more signaling attributes (message headers) and/or properties of the media session.

Parameter	Description
crn	call reference number of call targeted for modification
parmblkp	pointer to GC_PARM_BLK which contains call attributes that are being accepted (optional)
mode	completion mode; must be EV_ASYNC

The function returns either <0 (to indicate failure) or 0 depending only upon the validity of the parameters. The function return does not indicate any status as to the success or failure of the sending of the response message. The final result of the attempt to send the response is provided in termination events.

Note: This function is only supported when the value IP_T38_MANUAL_MODIFY_MODE has been set for the IPSET_CONFIG / IPPARM_OPERATING_MODE parameter using the **gc_SetConfigData( )** function. If this parameter value has not been set, the function call will fail with an error value of IPERR_BAD_PARM.

Receiving SDP Offer in Re-INVITE:

In cases where one or more media sessions are present in an SDP offer within the re-INVITE, those session proposals that are supported by the given media platform are indicated as Global Call parameter elements within the parameter block associated with the GCEV_REQ_MODIFY_CALL event. Each proposed media type is indicated by a separate parameter within the parameter block using the following:

GCSET_CHAN_CAPABILITY
IPPARM_LOCAL_CAPABILITY
- value = IP_CAPABILITY structure

For a symmetrical, full-duplex media proposal, at least two such parameters—one for transmit and one for receive—are forwarded in the parameter block. For a half-duplex proposal, an on-hold request, or a request to transition to T.38 fax mode, there may be only a single parameter element with the given set of media session attributes.

In addition to being informed of the media session proposals, the application is also informed of the remote RTP transport addresses. Each RTP port that is proposed in an SDP offer received within a re-INVITE (one per “m=” line) is indicated as a separate parameter within the parameter block associated with the GCEV_REQ_MODIFY_CALL event. These remote RTP address parameters are identified as follows:

IPSET_RTP_ADDRESS
IPPARM_REMOTE
- value = RTP_ADDR structure

Because SDP does not communicate RTCP ports, only RTP ports are exchanged; the RTCP port will have the typical “plus one” offset from the RTP port.

The application must retrieve and analyze the proposed capabilities to determine whether they are acceptable before calling **gc_AcceptModifyCall( )**.

Sending SDP Answer in 200OK:

To accept the changes to the dialog and media session exactly as proposed, or to negotiate the codec using the preferences in the original INVITE dialog, the application calls **gc_AcceptModifyCall( )** with a NULL pointer as **parmblkp**.

An application can also formulate a specific SDP answer by inserting 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. A full-duplex connection requires two such parameter elements, one for each direction. A half-duplex connection requires one parameter element with the direction field of the IP_CAPABILITY structure set appropriately. Accepting an on-hold request requires a single parameter with the proposed codec capability and one of the direction values that specifies inactive streaming.

If the capabilities to be used in the SDP answer—whether specified by the application or derived from the initial INVITE—do not match the capabilities that were contained in the

SDP offer (both codec capability and direction), the library treats the situation as a rejection of the call modification request. A 488 Not Acceptable Here response is sent to the remote party to terminate the re-INVITE dialog, and a GCEV_REJECT_MODIFY_CALL event is generated to the application.

Note: When accepting a codec change, the local endpoint's properties are updated immediately when the application calls this function; all outgoing packets use the new codec, and only incoming packets that use the new codec will be accepted. This may produce a perceptible artifact (for example, a click or a brief silence) until the remote endpoint receives the 200 OK response and changes its codec.

■ Termination Events

GCEV_ ACCEPT_MODIFY_CALL
: Successful termination event. Indicates that the 200OK response was successfully sent, and an ACK reply has been received. It also indicates that the requested call attribute change(s) has been performed locally.

GCEV_ACCEPT_MODIFY_CALL_FAIL
: Unsuccessful termination event. Indicates that the signaling of the modification acceptance response has failed. This could be caused by a failure in the message transport, a failure in the attempt to change the call attribute, or the expiration of a response timer for the request. The re-INVITE transaction is still in progress and the application may make another attempt to respond via a new call to **gc_AcceptModifyCall( )** or **gc_RejectModifyCall( )**. No modifications to the existing dialog or media session are performed and the current state remains as it was prior to the incoming modification request.

GCEV_REJECT_MODIFY_CALL
: Unsuccessful termination event. Indicates that the capabilities the application intended to signal in the SDP answer do not match any of the capabilities that were received in the SDP offer. This event implies that a 488 Not Acceptable Here final response was sent to the remote UA and that an ACK has been received, meaning that the re-INVITE dialog is terminated. No modifications to the existing media session are performed and the current state remains as it was prior to the incoming modification request.

■ Cautions

- Only one modification transaction can be pending in a call at any given time. Until the pending re-INVITE has been accepted, rejected, or canceled, no additional re-INVITE can be sent by either party.
- Only one attempt to send a response to a re-INVITE request can be pending at a time. A response must fail (as indicated by a failure termination event) before a new response is attempted, otherwise the function call will fail.
- The GCEV_REQ_MODIFY_CALL event will only arrive when a call is connected. But if the call is dropped—either locally via **gc_DropCall( )** or remotely as indicated by a GCEV_DISCONNECTED event—before a response is initiated via **gc_AcceptModifyCall( )**, the request is invalid and the response can no longer be sent.

- The potential for glare situations exist with a CANCEL being received from the remote party as the local application intends to send 200OK. If the library receives the CANCEL before the **gc_AcceptModifyCall( )**, the function call fails because the re-INVITE dialog is terminated and the application receives an informational GCEV_MODIFY_CALL_CANCEL event.

Errors

- The function returns GC_ERROR if any of the parameters is invalid, if the call is not in the connected state, if there is no re-INVITE request pending, or if the value of the configuration parameter IPSET_CONFIG / IPPARM_OPERATING_MODE has not been set to IP_T38_MANUAL_MODIFY_MODE. Use the **gc_ErrorInfo( )** function to retrieve further information.
- Upon receiving a GCEV_ACCEPT_MODIFY_FAIL event, use the **gc_ResultInfo( )** function to retrieve information about the failure event. See the “Error Handling” section in the *Global Call API Programming Guide*. All Global Call error codes are defined in the *gcerr.h* file while IP-specific error codes are specified in *gcip_defs.h*. On failure, no modifications to the existing dialog or media session are performed and the current state remains as it was prior to the attempting the modification request.

Example

```
.
.
.
/* Dialogic Header Files */
#include <gcip.h>
#include <gclib.h>
.
.
.

   /* SRL event handler: */
   for (;;)
   {
      if (-1 != sr_waitevt(500))   process_event();
   }

void process_event(void)
{
   METAEVENT    metaevent;
   GC_INFO      t_info;

   /* Populate the metaEvent structure */
   if(GC_SUCCESS != gc_GetMetaEvent(&metaevent))   return;

   /* process GlobalCall events */
   if ((metaevent.flags & GCME_GC_EVENT) == 0)   return;

   switch (metaevent.evttype)
   {
   .
   .
   .
      case GCEV_REQ_MODIFY_CALL:  /* request to modify call attribute */
      {
         EXTENSIONEVTBLK *extblkp = metaevent.extevtdatap;
         GC_PARM_BLKP parm_blkp = &extblkp->parmblk;
```

```
   GC_PARM_BLKP replyParmblkp = NULL;
   GC_PARM_DATAP curParm = NULL;
   IP_CAPABILITY cap;
   RTP_ADDR rtp;
   unsigned char proposal_accepted = FALSE;

   while ((curParm = gc_util_next_parm(parm_blkp, curParm)) != NULL)
   {
      if ((curParm->set_ID == GCSET_CHAN_CAPABILITY) &&
          (curParm->parm_ID == IPPARM_LOCAL_CAPABILITY))
      {
         memcpy(&cap, curParm->value_buf, curParm->value_size);
         /* determine if capability is acceptable (logic not shown) */
         if (isCapabilityAcceptable(cap) == TRUE)
         {
            /* insert parameter with accepted capability in parameter block reply */
            /* (logic not shown) */
            insertCapIntoReply(cap,replyParmblkp);
            proposal_accepted = TRUE;
         }
      }
      else if ((curParm->set_ID == IPSET_SIP_MSGINFO) &&
               (curParm->parm_ID == IPPARM_SIP_HDR))
      {
         /* parse SIP header and make appropriate updates (logic not shown) */
         proposal_accepted = TRUE;
      }
      else if ((curParm->set_ID == IPSET_RTP_ADDRESS) &&
               (curParm->parm_ID == IPPARM_REMOTE))
      {
         memcpy(&rtp, curParm->value_buf, curParm->value_size);
         if (isMediaReRouteAcceptable(rtp) == TRUE)
         {
            /* update RTP transport addresses in GUI (logic not shown) */
            updateRTPGUI(&rtp);
            proposal_accepted = TRUE;
         }
      }
   }

   /* if proposal is acceptable accept the request                          */
   /* format accepted attributes (i.e. media types) in a parmblk (optional, */
   /* NULL if none) */
   if (proposal_accepted)
   {
      if (gc_AcceptModifyCall(crn, replyParmblkp, EV_ASYNC) < 0)
        /* failure logic here*/
   }
   else /* not acceptable so respond with SIP Client Error    */
        /* final response of 488 Not Acceptable Here          */
      if (gc_RejectModifyCall(crn,
                              IPEC_SIPReasonStatus488NotAcceptableHere,
                              EV_ASYNC) < 0)
        /* failure logic here */

   break;
}

case GCEV_ACCEPT_MODIFY_CALL:
   .
   .
   .
   /* notify user of changed attribute */
   .
   .
   .
   break;
```

```
      case GCEV_ACCEPT_MODIFY_CALL_FAIL:
         /* process failure to change attribute */
         if (gc_ResultInfo(&metaevent, &t_info) < 0)
         {
            /* failure logic here */
         }
         /* process information contained in t_info */
         /* can optionally call gc_RejectModifyCall( ) to retry */
         .
         .
         .
         break;

      case GCEV_REJECT_MODIFY_CALL:
         .
         .
         .
         /* notify user of rejected attribute */
         .
         .
         .
         break;

      case GCEV_REJECT_MODIFY_CALL_FAIL:
         /* process failure to reject request */
         if (gc_ResultInfo(&metaevent, &t_info) < 0)
         {
            /* failure logic here */
         }
         /* process information contained in t_info */
         /* can optionally call gc_RejectModifyCall( ) to retry */
         .
         .
         .
         break;
         .
         .
         .
   } /* endof switch */
} /* endof process_event function */
```

See Also

- **gc_RejectModifyCall( )**
- **gc_ReqModifyCall( )**

gc_RejectModifyCall( )

Name: int gc_RejectModifyCall (crn, reason, mode)

Inputs:

CRN crn	• call reference number of call targeted for modification
unsigned long reason	• reason for rejecting request to change call attribute
unsigned long mode	• completion mode (EV_ASYNC)

Returns: 0 if successful
<0 if unsuccessful

Includes: gclib.h

Category: Call Modification

Mode: Asynchronous only

Description

The **gc_RejectModifyCall( )** function is used to reject a request from the network or remote party to change an attribute of the current call.

This function initiates a 3xx thorough 6xx response code to an incoming re-INVITE request, as indicated by an incoming GCEV_REQ_MODIFY_CALL event on the respective call object. The actual response code that is sent is specified by the **reason** parameter.

Parameter	Description
crn	call reference number of the call targeted for modification; must match the CRN contained in the GCEV_REQ_MODIFY_CALL event
reason	the reason for rejecting the request to modify call attributes, specified using the IPEC_SIPReasonStatusXXX… symbolic defines for SIP reason codes from 300 through 699. See *gcip_defs.h* or Chapter 10 of the *Global Call IP Technology Guide* for a complete list of these defines.
mode	must be EV_ASYNC

The function returns either <0 (to indicate failure) or 0, depending only upon the validity of the parameters. The function return does not indicate any status as to the success or failure of the sending of the rejection response message. The final result of sending the response is provided to the application in termination events.

Note: This function is only supported when the value IP_T38_MANUAL_MODIFY_MODE has been set for the IPSET_CONFIG / IPPARM_OPERATING_MODE parameter using the **gc_SetConfigData( )** function. If this parameter value has not been set, the function call will fail with an error value of IPERR_BAD_PARM.

Termination Events

GCEV_ REJECT_MODIFY_CALL

Successful termination event. Indicates that rejection of the received re-INVITE request has completed successfully. This event implies that the specified 3xx through 6xx response was sent and that an ACK was received from the remote party. The requested call attribute modifications are not performed and the call state remains as it was prior to receiving the incoming re-INVITE request.

GCEV_REJECT_MODIFY_CALL_FAIL

Unsuccessful termination event. Indicates that the signaling of the rejection response failed. The re-INVITE transaction is still in progress and the application may make another attempt to respond via a new call to **gc_AcceptModifyCall( )** or **gc_RejectModifyCall( )**. No modifications to the existing media session are performed and the current state remains as it was prior to receiving the incoming re-INVITE request.

Cautions

- Only one modification transaction can be pending in a call at any given time. Until the pending re-INVITE has been accepted, rejected, or canceled no additional re-INVITE can be sent by either party.
- A GCEV_REQ_MODIFY_CALL event can only arrive when a call is connected. But if the call is dropped—either locally via **gc_DropCall( )** or remotely as indicated by a GCEV_DISCONNECTED event—before a response is initiated via **gc_RejectModifyCall( )**, the request is invalid and the response can no longer be sent.
- Only one attempt to respond to a re-INVITE request can be pending at a time. A response attempt must fail (as indicated by a failure termination event) before a new response is attempted, otherwise the function call will fail.

Errors

- The function returns GC_ERROR if any of the parameters is invalid, if the call is not in the connected state, if there is no pending re-INVITE request, or if the value of the configuration parameter IPSET_CONFIG / IPPARM_OPERATING_MODE has not been set to the value IP_T38_MANUAL_MODIFY_MODE. Use the **gc_ErrorInfo( )** function to retrieve further information.
- Upon receiving a GCEV_REJECT_MODIFY_CALL_FAIL event, use the **gc_ResultInfo( )** function to retrieve information about the event. See the “Error Handling” section in the *Global Call API Programming Guide*. All Global Call error codes are defined in the *gcerr.h* file while IP-specific error codes are specified in *gcip_defs.h*. On failure, no modifications to the existing dialog or media session are performed and the current state remains as it was prior to the incoming modification request.

Example

```
.
.
.
/* Dialogic Header Files */
#include <gcip.h>
#include <gclib.h>
.
.
.

   /* SRL event handler: */
   for (;;)
   {
      if (-1 != sr_waitevt(500))  process_event();
   }

void process_event(void)
{
   METAEVENT    metaevent;
   GC_INFO      t_info;

   /* Populate the metaEvent structure */
   if(GC_SUCCESS != gc_GetMetaEvent(&metaevent))  return;

   /* process GlobalCall events */
   if ((metaevent.flags & GCME_GC_EVENT) == 0)  return;
   switch (metaevent.evttype)
   {
   .
   .
   .
      case GCEV_REQ_MODIFY_CALL:  /* request to modify call attribute */
      {
         EXTENSIONEVTBLK *extblkp = metaevent.extevtdatap;
         GC_PARM_BLKP parm_blkp = &extblkp->parmblk;
         GC_PARM_BLKP replyParmblkp = NULL;
         GC_PARM_DATAP curParm = NULL;
         IP_CAPABILITY cap;
         RTP_ADDR rtp;
         unsigned char proposal_accepted = FALSE;

         while ((curParm = gc_util_next_parm(parm_blkp, curParm)) != NULL)
         {
            if ((curParm->set_ID == GCSET_CHAN_CAPABILITY) &&
                (curParm->parm_ID == IPPARM_LOCAL_CAPABILITY))
            {
               memcpy(&cap, curParm->value_buf, curParm->value_size);
               /* determine if capability is acceptable (logic not shown) */
               if (isCapabilityAcceptable(cap) == TRUE)
               {
                  /* insert parameter with accepted capability in parameter block reply */
                  /* (logic not shown) */
                  insertCapIntoReply(cap,replyParmblkp);
                  proposal_accepted = TRUE;
               }
            else if ((curParm->set_ID == IPSET_SIP_MSGINFO) &&
                     (curParm->parm_ID == IPPARM_SIP_HDR))
            {
               /* parse SIP header and make appropriate updates (logic not shown) */
               proposal_accepted = TRUE;
            }
            else if ((curParm->set_ID == IPSET_RTP_ADDRESS) &&
                     (curParm->parm_ID == IPPARM_REMOTE))
            {
               memcpy(&rtp, curParm->value_buf, curParm->value_size);
```

```
         if (isMediaReRouteAcceptable(rtp) == TRUE)
         {
            /* update RTP transport addresses in application (logic not shown) */
            updateRTPGUI(&rtp);
            proposal_accepted = TRUE;
         }
      }
   }
   /* if proposal is acceptable accept the request */
   /* format accepted attributes (i.e. media types) in a parmblk (optional, */
   /* NULL if none) */
   if (proposal_accepted)
   {
      if (gc_AcceptModifyCall(crn, replyParmblkp, EV_ASYNC) < 0)
        /* failure logic here */
   }
   else /* not acceptable so respond with SIP Client Error */
        /* final response of 488 Not Acceptable Here          */
      if (gc_RejectModifyCall(crn,
                              IPEC_SIPReasonStatus488NotAcceptableHere,
                              EV_ASYNC) < 0)
        /* failure logic here */
   break;
}

case GCEV_ACCEPT_MODIFY_CALL:
  .
  .
  .
  /* notify user of changed attribute */
  .
  .
  .
  break;

case GCEV_ACCEPT_MODIFY_CALL_FAIL:
   /* process failure to change attribute */
   if (gc_ResultInfo(&metaevent, &t_info) < 0)
      /* failure logic here */

   /* process information contained in t_info */
   /* can optionally call gc_RejectModifyCall( ) to retry */
   .
   .
   .
   break;

case GCEV_REJECT_MODIFY_CALL:
   .
   .
   .
   /* notify user of rejected attribute */
   .
   .
   .
   break;

case GCEV_REJECT_MODIFY_FAIL:
   /* process failure to reject request */
   if (gc_ResultInfo(&metaevent, &t_info) < 0)
      /* failure logic here */
```

```
            /* process information contained in t_info */
            /* can optionally call gc_RejectModifyCall( ) to retry */
            .
            .
            .
            break;

            .
            .
            .
      } /* endof switch */
   } /* endof process_event function */
```

See Also

- **gc_AcceptModifyCall( )**
- **gcReqModifyCall( )**

gc_ReqModifyCall( )

Name: int gc_ReqModifyCall (crn, parmblkp, mode)

Inputs:

CRN crn	• call reference number of call targeted for modification
GC_PARM_BLK *parmblkp	• pointer to GC_PARM_BLK which contains attributes of call requested for modifying
unsigned long mode	• completion mode (EV_ASYNC)

Returns: 0 if successful
<0 if unsuccessful

Includes: gclib.h

Category: Call Modification

Mode: Asynchronous

Description

The **gc_ReqModifyCall( )** function is used to initiate a request to the network or remote party to change an attribute of the current call.

This function initiates a subsequent INVITE (also known as a re-INVITE) request in the context of a current dialog (connected call). The re-INVITE can be used to change signaling headers or one or more attributes of the media session. This function is also used to cancel a pending re-INVITE that the application previously initiated.

Parameter	Description
crn	call reference number of call targeted for modification
parmblkp	pointer to GC_PARM_BLK which contains attributes of call requested for modifying. This parameter block may contain a combination of SIP header fields and Global Call channel capabilities (which must be identical to the current session capabilities except for the direction property) that will be inserted into the SDP offer formulated by the library.
mode	must be EV_ASYNC

The function returns either <0 (to indicate failure) or 0, depending only upon the validity of the parameters. The function return does not indicate any status as to the success or failure of the sending of the re-INVITE request message. The final result of the attempt to send the request is provided in termination events.

Note: This function is only supported when the value IP_T38_MANUAL_MODIFY_MODE has been set for the IPSET_CONFIG / IPPARM_OPERATING_MODE parameter using the **gc_SetConfigData( )** function. If this parameter value has not been set, the function call will fail with an error value of IPERR_BAD_PARM.

The parameters elements contained in the GC_PARM_BLK that is passed to this function determine the contents of the re-INVITE request message. A special parameter element is also defined to cancel a pending re-INVITE request.

To set one or more message header fields in the re-INVITE request, the application inserts into the GC_PARM_BLK a parameter of the following form for each header field to be set:

IPSET_SIP_MSGINFO
    IPPARM_SIP_HDR
    - value = string representing the complete header field, including field name

Most SIP header fields that are valid in an INVITE request can be modified in a re-INVITE request without restriction. The most notable exceptions to this generalization are the Call-ID header and the URI and Tag in the To and From headers, which RFC 3261 specifies must match the headers in the original INVITE request. The following table specifies the header fields that are subject to restrictions or that are automatically populated by the SIP stack.

Header Field	Modifiable Parameters	Restricted Parameters	Automatically Populated Information
Call-ID	None	All	All
Contact	All	None	If not specified, copied from last INVITE or 2xx response transmitted in current dialog
CSeq	None	All	All
From	Display, URI parameters except: user, ttl, method, maddr	URI, Tag	URI, Tag
Max-Forwards	All	None	If not specified, 70
To	Display, URI parameters except: user, ttl, method, maddr	URI, Tag	URI, Tag
Via	All	None	If not specified, copied from last INVITE or 2xx response transmitted in current dialog

To request a change in the attributes of a media session, the application uses the same parameter mechanism that is used when offering a session invitation via **gc_MakeCall( )**. The application inserts into the GC_PARM_BLK one or more parameter of the following form:

GCSET_CHAN_CAPABILITY
    IPPARM_LOCAL_CAPABILITY
    - value = IP_CAPABILITY structure containing the details of the proposed media session, including capability (transcoder type) and direction

To modify the media attributes for a full-duplex connection, the application must insert at least two of these parameters, one for each direction, with the appropriate value set in the direction field of each IP_CAPABILITY structure. All fields in each IP_CAPABILITY structure must be fully specified even if only one characteristic is actually being changed (for example, if only the direction field is being modified to place a call on hold). If no media capability parameters are inserted into the GC_PARM_BLK, the library

automatically includes the last SDP answer from the dialog when it constructs the re-INVITE request.

To cancel a pending re-INVITE request, the application inserts into the GC_PARM_BLK the following parameter:

IPSET_MSG_SIP

IPPARM_SIP_METHOD

- value = IP_MSGTYPE_SIP_CANCEL, size = sizeof(int)

Note: When using this parameter value, this must be the only parameter element inserted into the GC_PARM_BLK.

Termination Events

GCEV_ MODIFY_CALL_ACK

Successful termination event for call modification request. Indicates that the network or remote party accepted and acknowledged the request with a 200OK, and that the library has acknowledged the 200OK. This event also indicates that any media changes that were proposed and accepted have been completed.

GCEV_MODIFY_CALL_REJ

Unsuccessful termination event for call modification request, indicating that the request was rejected. The network or remote party declined and rejected the request by sending a 3xx, 4xx, 5xx, or 6xx response code in reply to the re-INVITE, and the library automatically sent an ACK. The specific response code can be retrieved from the Global Call METAEVENT by calling **gc_ResultInfo( )**. If the response code from the remote party was a 408 Request Time-out or 481 Dialog/Transaction Does Not Exist, the call that was being modified is disconnected automatically, and a GCEV_DISCONNECTED event is generated to the application. For all other response codes, no modifications to the existing dialog or media session are performed and the current state remains as it was prior to the attempting the modification request.

GCEV_MODIFY_CALL_FAIL

Unsuccessful termination event for call modification request, indicating that the signaling of the request failed. Some possible reasons include a failure in the message transport, a time-out awaiting the response from the network or remote party, attempting to modify a call which is not currently connected, or attempting to initiate a request to modify a call while another modify request transaction is still pending. More specific information can be retrieved from the Global Call METAEVENT by calling **gc_ResultInfo( )**. On failure, no modifications to the existing dialog or media session are performed and the current state remains as it was prior to the attempting the modification request.

GCEV_CANCEL_MODIFY_CALL

Successful termination event for a request to cancel a pending call modification request. Indicates that the remote UA accepted the CANCEL method and sent a 200OK, and the library automatically sent an ensuing ACK. The previously sent re-INVITE dialog is terminated and no attribute changes are performed. In this case the application will not receive a termination event for the original **gc_ReqModifyCall( )** call (the one which initiated the re-INVITE dialog).

GCEV_CANCEL_MODIFY_CALL_FAIL

Unsuccessful termination event for a request to cancel a pending call modification request. Indicates that the signaling of the CANCEL method failed, likely due to invalid state, such as having received a final 2xx-6xx response to the subject re-INVITE. In this case, the application *will* receive a termination event for the prior **gc_ReqModifyCall( )** call (either before or after this event) to indicate the successful or failed outcome of original re-INVITE transaction.

Cautions

- Only asynchronous mode is supported. Calling the function in synchronous mode will fail and return an error value of GC_ERROR while setting CCLIB error to IPERR_BAD_PARAM.
- This function can only be called in the connected call state. If the CRN is not valid, the function fails and returns GC_ERROR while setting CCLIB error to IPERR_BAD_PARAM.
- Only one re-INVITE transaction can be pending in a call at any given time. Any re-INVITE transaction previously issued on the call must terminate (as indicated by a termination event) before a new one is initiated, otherwise the function will fail.

Errors

- The function returns GC_ERROR (with CCLIB error set to IPERR_BAD_PARM) if the CRN is not valid, if the mode is not set to EV_ASYNC, or if the value of the configuration parameter IPSET_CONFIG / IPPARM_OPERATING_MODE has not been set to IP_T38_MANUAL_MODIFY_MODE.
- Upon receiving a termination event that indicates a failure, use the **gc_ResultInfo( )** function to retrieve information about the event. See the “Error Handling” section in the *Global Call API Programming Guide*. All Global Call error codes are defined in the *gcerr.h* file while IP-specific error codes are specified in *gcip_defs.h*.

Example

The following example illustrates the use of **gc_ReqModifyCall( )** to refresh the Contact header:

```
...
/* Dialogic Header Files */
#include <gcip.h>
#include <gclib.h>
...

/* Request Contact refresh:                                          */
/* Assumes: 1) caller has verified call to be in connected state     */
/*          2) caller has enabled event handler for GCEV_MODIFY_CALL_ACK, */
/*             GCEV_MODIFY_CALL_REJ, and GCEV_MODIFY_CALL_FAIL.      */

int refreshToHomeLocation (CRN crn)
{

   char *pContactHeader = "Contact: Rich <r.intelligent@myhomeISP.com>";
```

```
    gc_util_insert_parm_ref(&parmblkp,
                            IPSET_SIP_MSGINFO,
                            IPPARM_SIP_HDR,
                            (unsigned char)strlen(pContactIdHeader) + 1,
                            pContactHeader);

    if (NULL == parmblkp)  return FAILURE;

    if (gc_ReqModifyCall(crn, parmblkp, EV_ASYNC) < 0)  return FAILURE;

    gc_util_delete_parm_blk(parmblkp);

} /* End of function. */
```

See Also

- **gc_AcceptModifyCall( )**
- **gc_RejectModifyCall( )**

1.6 User Configuration Files

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

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

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

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

Example

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

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

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

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

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

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

1.7 Distinguishing Between 180 and 183 Responses

This feature provides Global Call with the ability to distinguish between 180 and 183 responses when the application receives a GCEV_ALERTING event. For additional information, see Section 3.4.7, "Global Call IP for Host Media Processing Technology Guide", on page 83.

1.8 Fast Start Coder

This feature allows applications to examine offered fast start coders as soon as a new incoming call is offered to the application. The IP call control library now provides support for the application to examine this coder information by placing it in the extra data (in the form of IP_CAPABILITY structures) associated with the GCEV_OFFERED event. Applications can now be programmed to choose whether or not to accept the fast start coder information reported by GCEFV_OFFERED events.

For additional information about the fast start coder, see Retrieving Coder Information from Fast Start Call Offers in Section 3.4.7, "Global Call IP for Host Media Processing Technology Guide", on page 83.

1.9 One Step Installation Process

It is not necessary to uninstall the previous version of Release 1.3 software before upgrading to a new Service Update. During installation, the Service Update will determine if there is a previous version of the Dialogic® HMP 1.3WIN Software installed, or that no Dialogic® HMP 1.3WIN Software is currently installed, and correctly install the appropriate Dialogic® HMP 1.3WIN Software. See the Dialogic® Host Media Processing Software

Release 1.3WIN *Software Installation Guide* (05-2418-002) for complete installation information.

Note: If a Dialogic® HMP 1.1WIN Software Release software version was previously installed, it must be uninstalled before installing the Dialogic® HMP 1.3WIN software. For additional information about uninstalling Dialogic® HMP 1.1WIN software, see the Dialogic® Host Media Processing Software Release 1.3WIN *Software Installation Guide*.

1.10 Support for 120 T.38 Fax Channels

The number of T.38 fax channels supported has been increased from 32 to 120. Refer to Section 3.1.1, “Dialogic® Host Media Processing Software Release 1.3WIN Release Guide”, on page 75 for updated information about media resources and resource configurations tested.

1.11 Enhancements to SIT Frequency Detection

This release provides the following enhancements to Special Information Tone (SIT) frequency detection:

- Broader default SIT sequence definitions to allow greater coverage for SIT sequences detected in the field.
- Three new SIT sequence definitions in the SIT tone set. The new SIT sequences are: InterLATA no circuit (TID_SIT_NC_INTERLATA), InterLATA reorder tone (TID_SIT_RO_INTERLATA), and ineffective other (TID_SIT_IO).
- A new catch all SIT sequence definition (TID_SIT_ANY) to cover SIT sequences that fall outside the range of the defined SIT sequences.
- Support for the **ATDX_CRTNID( )** function to allow retrieval of the SIT ID.

For additional information about SIT frequency detection, see Section 3.4.14, “Voice API for Host Media Processing Library Reference”, on page 98 and Section 3.4.15, “Voice API for Host Media Processing Programming Guide”, on page 102.

1.12 Configurable RTCP Reporting Interval

Previously, Real Time Control Protocol (RTCP) was calculated about every 5 seconds. The actual interval at which RTCP is calculated is somewhat randomized to distribute CPU consumption. Users will now be able to specify the reporting interval within a specific range of 1 to 15 seconds. See RTCP Reporting Interval in Section 3.4.10, “IP Media Library API for Host Media Processing Library Reference”, on page 93 for additional information.

1.13 Configurable UDP Port Range

Previously, both the RTP and RTCP ports for an IP device were statically assigned and could not be changed at runtime. The user will now be able to select the RTP Base port number at runtime. See UDP Port Range in Section 3.4.10, "IP Media Library API for Host Media Processing Library Reference", on page 93 for additional information.

1.14 Support for G.726 IP Coder

The G.726 Codec is now supported for IP (RTP) encoding/decoding (16, 24, 32, and 40 kbps). See G.726 Coder in Section 3.4.10, "IP Media Library API for Host Media Processing Library Reference", on page 93 for additional information.

1.15 New CSP Function

Added the **ATEC_TERMMSK()** function to the Continuous Speech Processing (CSP) API library. This function returns the reason for the last CSP I/O function termination. For additional information about this function, see the *Continuous Speech Processing API for Linux and Windows Operating Systems Library Reference*.

1.16 Comfort Noise Generation in Conferencing

Comfort noise can now be generated on the outputs of a conference whenever the conference output drops below the no-talker level on an across-the-board basis. The conferencing comfort noise generation (CNG) can be enabled or disabled for all conferences by pre runtime user configuration. This is accomplished by editing the .config file as described in the following paragraphs.

1.16.1 Enabling Comfort Noise

To enable comfort noise, edit the 0x3b38 parameter in the [0x3b] section of the .config file by changing **`SetParm=0x3b38,0`** to **`SetParm=0x3b38,1`** as shown in bold below:

```
[0x3b]
SetParm=0x3b38,1 !  Disable Comfort Noise in Conferencing (1=Enable, 0=Disable)
```

It is recommended that you accept the default values for setting the silence threshold, noise amplitude, and hang time parameters. If it is necessary, however, to modify these parameters, you will need to manually add the following parameters to the .config file following the 0x3b38 parameter:

```
SetParm=0x3b39,100 !  Silence threshold, default is 100 (-43dbm), 1630 (-18.5dbm) is maximum

SetParm=0x3b3a,200 !  Noise amplitude, default is 200 (-43dbm), 3270 (-18.5dbm) is maximum

SetParm=0x3b3b,25 !  Hang time, default is 25 (25 gives 12msx25=300ms, but actually results in
620 ms due to exponential averaging)
```

1.16.2 Editing the Config File

The first step in generating a modified FCD file is to edit the corresponding CONFIG file. Once the CONFIG file parameters are modified, the *fcdgen* utility is used to convert the CONFIG file to an FCD file.

Note: If you want to preserve the default parameter values contained in the CONFIG file, make a backup copy of the file prior to editing it.

To edit the CONFIG file:

1. From the command prompt, go to the *\Program Files\Dialogic\Data* directory and locate the CONFIG file.
2. Using a text editor (for example, WordPad), open the CONFIG file that corresponds to the FCD file you want to modify. By default, the CONFIG file will have the same file name as the FCD file, but with a *.config* extension.
3. Edit the CONFIG file as necessary.
4. Save and close the CONFIG file.

Proceed with Section 1.16.3, “Generating the FCD File”, on page 53.

1.16.3 Generating the FCD File

In order to generate an FCD file, the corresponding CONFIG file must be converted to an FCD file. *fcdgen* converts the CONFIG file into a format that can be read directly by the downloader.

1. From the command prompt, go to the *\Program Files\Dialogic\Data* directory.
2. Execute *fcdgen* as follows:

```
..\bin\fcdgen -f <input file>.config -o <output file>.fcd
```

For example:

```
..\bin\fcdgen -f qs_t1.config -o qs_t1.fcd
```

The resulting FCD file is created in the *\Program Files\Dialogic\Data* directory. If the -o option is omitted from the command, the default output FCD file will have the same filename as the user-modified input configuration file, but with a *.fcd* extension.

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® Host Media Processing 1.3WIN Software Release. 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® Host Media Processing 1.3WIN Software Release. 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. Additional information on defects may be available via the Defect Query tool at *http://membersresource.dialogic.com/search/ptrs/ptrsearch.asp* (Note that when you select this link, you will be asked to either LOGIN or JOIN.).

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, Dialogic® HMP 1.3WIN
  - a software product; for example, the Global Call 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.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00041583	--	121	Dialogic® HMP Software	Dialogic® HMP Software sends the bye message, IPEC_SIPReasonStatusBYE = 5800, upon making a call after the remote Nortel CS1000 sends an additional media type.
Resolved	IPY00041789	--	121	Global Call IP (SIP)	Update is not checked in the “Allow” header of Ingress INVITE/ 200 OK.
Resolved	IPY00041296	--	121	H.323	In the LD state MediaDrop, the EVENT_CHANNELS_OPENED event is handled only when the call is in the Connected state. This event should be handled in the Alerting and Proceeding states as well.
Resolved	IPY00041876	--	121	IP Media	H.323 transfer fails with the RTF log ERR1 message, EVENT_CONNECT_SIGNALING failed.
Resolved	IPY00041048	--	120	DTMF (H245)	Digits are missing when using User Input Indication (UII) messages to receive DTMF.
Resolved	IPY00039677	--	120	Global Call	The **gc_ReleaseCall( )** function does not receive a termination event after the application attempts to handle a GCEV_TASKFAIL.
Resolved	IPY00041300	--	120	Global Call IP (SIP)	SIP calls are rejected with a "486 Busy Here" message.
Resolved	IPY00041118	--	120	Global Call IP (SIP)	The application is unable to make a SIP call using the **gc_MakeCall( )** function on the same channels previously used to make an H323 call.
Resolved	IPY00040440	--	120	Global Call IP (SIP)	A fall back occurs when DNS resolution fails while making a SIP HMP call.
Resolved	IPY00039823	--	120	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	IPY00039707	--	120	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.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00039965	--	120	GlobalCall (IP)	An outbound IP call fails with "IPEC_SIPReasonStatus503ServiceUnavailable" when the hostname is passed for the destination address in the dialstring.
Resolved	IPY00040743	--	120	IP	There is excessive logging generated by SIP INFO messages.
Resolved	IPY00039847	--	120	Signaling (H.323)	A RequestMode is not triggered upon CED detection in AUTO mode//When we attempt to make a TDM fax call across two IPT machines joined by H323 across an IP network - the sending of the fax is unsuccessful.
Resolved	IPY00038945	--	118	Conferencing (DCB)	When running a conferencing activity under high load on dual core systems, there is a potential for the **dcb_delconf( )** function to hang and never return.
Resolved	IPY00039505	--	118	Fax	A request to switch to T.38 fax is rejected, but a gcev_taskfail event is not received.
Resolved	IPY00039639	--	118	Global Call	The **gc_SetAlarmParm( )** function causes an assert in *msvcrt.dll*.
Resolved	IPY00039451	--	118	Global Call	The **gc_AcceptModifyCall( )** function fails to return a termination event.
Resolved	IPY00039564	--	118	Global Call IP (SIP)	A GCEV_EXTENSION event (IPSET_SWITCH_CODEC, IPPARM_READY) is not received for T.38 call.
Resolved	IPY00039401	--	118	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	--	118	Global Call IP (SIP)	The SIP INVITE fast-start codec G.729A should default to annex B.
Resolved	IPY00039315	--	118	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	--	118	Global Call IP (SIP)	There is an incorrect wrong codec in the SIP 200OK SDP. It should be G729a because the INVITE contained G729a, rather than G729ab.
Resolved	IPY00039036	--	118	Global Call IP (SIP)	The application returns a "486 Busy Here" response when closing and reopening a device in 3PCC mode.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00038992	--	118	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.
Resolved	IPY00039409	--	118	IPML	An incorrect eQoSType, QOSTYPE_ROUNDTRIPLATENCY, is specified for QoSRTCPTimeout and QoSRTPTimeout.
Resolved	IPY00038339	--	113	Dialogic® HMP Software	Blue Screen of Death (BSOD) occurs.
Resolved	IPY00038848	--	113	Global Call	When **gc_DropCall( )** returns a cause code of 5801 from the application, an invalid response code appears in the RTF log.
Resolved	IPY00038868	--	113	Global Call IP (SIP)	A GCEV_CONNECTED event is not returned from outgoing calls using SIP.
Resolved	IPY00038827	--	113	Global Call IP (SIP)	The **gc_AnswerCall( )** function does not receive a response when a SIP CANCEL is received after a SIP INVITE.
Resolved	IPY00038732	--	113	Global Call IP (SIP)	Dialogic® HMP Software does not choose the bit rate specified by the application if the remote codec fails to specify the bit rate.
Resolved	IPY00038708	--	113	SRL	An access violation occurs during **sr_freedata( )** causing the application to stop.
Resolved	IPY00038218	--	111	Device Management	An exception occurs in the Device Management library when **dev_SetError( )** keeps data in local thread storage.
Resolved	IPY00038342	--	111	Global Call IP (SIP)	Simultaneous disconnects cause an error at RTL after a number of calls.
Resolved	IPY00038150	--	109	Global Call IP	A simultaneous re-INVITE from both sides of call causes a TASKFAIL for the **gc_ReqModifyCall( )** function.
Resolved	IPY00038365	--	109	Global Call IP (SIP)	Egress SIP calls work briefly, but then omit the SDP in an egress INVITE message.
Resolved	IPY00038343	--	109	Global Call IP (SIP)	SIP registration fails after SIP7, despite the INIT_IP_VIRTBOARD structure specifying SIP_registrar_registrations up to 23.
Resolved	IPY00038240	--	109	Global Call IP (SIP)	An assert occurs on an inbound re-INVITE.
Resolved	IPY00038060	--	109	Global Call IP (SIP)	A rvSdpStringGetData crash occurs when there are no media attributes containing “rtpmap” in a SIP INVITE.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00037359	--	108	Global Call IP	The receipt of RFC2833 digits after the G.711 Comfort Noise packet causes Dialogic® HMP Software to double detect the first RFC2833 digit incorrectly.
Resolved	IPY00036779	--	108	Global Call IP (SIP)	Assertion faults are received when “DON’T CARE” is used with the G.726 coder.
Resolved	IPY00037699	--	108	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	IPY00037603	--	108	Installation	The SNMP component gets installed despite not choosing it as an option during System Release installation.
Resolved	IPY00037825	--	107	Global Call IP (SIP)	The **gc_InvokeXfer( )** function fails with the AVAYA IP PBX.
Resolved	IPY00037778	--	107	Global Call IP (SIP)	The Session Description Protocol (SDP) is missing from SIP re-INVITE after calling the **gc_ReqModifyCall( )** function.
Resolved	IPY00037737	--	107	Global Call IP (SIP)	The **gc_makecall( )** function returns a GCEV_TASKFAIL event when frames per packet are set to 20.
Resolved	IPY00037704	--	107	Media	The **dev_connect( )** function does not return immediately when running in asynchronous mode.
Resolved	IPY00037708	--	106	Diagnostics	The `its_sysinfo` tool, a standalone utility used to collect system data, is not collecting a full memory dump.
Resolved	IPY00037613	--	106	Global Call IP (SIP)	The **gc_InvokeXfer( )** function does not return a termination event.
Resolved	IPY00037541	--	106	SIP Signal	Upon receipt of the GCEV_REQ_MODIFY_CALL event, the RTP port-address info in the event data stays unchanged until the **gc_AcceptModifyCall( )** function is called and the GCEV_ACCEPT_MODIFY_CALL event is received.
Resolved	IPY00036658	--	106	SRL	The **gc_GetMetaEvent( )** function is failing with an Internal (SRL) failure.
Resolved	IPY00037252	--	102	Fax	The Fax resource hangs indefinitely when calling the **dev_Disconnect( )** function.
Resolved	IPY00037172	--	102	Global Call IP	The application returns a taskfail with IPERR_INVALID_PHONE_NUMBER as the reason.
Resolved	IPY00037333	--	102	SIP	There is no audio when calling from a digital phone through Alcatel PBX to Dialogic® HMP Software Release 1.3WIN.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00037360	--	102	SNMP	The SNMP service crashes periodically.
Resolved	IPY00036292	--	101	Host Interface	Issuing **dx_stopch( )** with the mode parameter set to EV_NOSTOP does not return a TDX_NOSTOP event when called after **dx_getdig( )**.
Resolved	IPY00036930	--	101	IPML Veneer	There is an excess byte (trailing '\0') put in the NonStandardData field of the Q931 Facility message sent by the **gc_Extension( )** function. Previous builds (SU81) did not have this extra byte.
Resolved	IPY00036250	--	100	SIP Call Control	Dialogic® HMP Software Release 1.3WIN SU42 works well with 3rd party gateway; however, when customer upgrades to Dialogic HMP Software Release 1.3WIN SU78, the call cannot be connected with 3rd party gateway anymore.
Resolved	IPY00035642	--	100	Voice	Dialogic® HMP Software crashes if the CONFIG file contains Silence Threshold setparm (0x70b) during startup.
Resolved	IPY00036266	--	96	Demo	Fix Dialogic® HMP Fax firmware to properly handle T.30 RTN ("Retrain Negative" response).
Resolved	IPY00036265	--	96	Fax	Fix Dialogic® HMP Fax f/w to properly ignore T.30 CRP ("Cmd Repeat" response).
Resolved	IPY00036052	--	96	Fax	Blue Screen of Death (BSOD) when using fax.
Resolved	IPY00035875	--	96	Global Call IP	Global Call IP applications that were recompiled using Service Updates 86, 89, or 91 need to be recompiled again with Service Update 96 or later.
Resolved	IPY00032865	--	96	Voice	If a user attempts to play forever or until EOF is reached (specifying io_length = -1) with UIO plays on Dialogic® 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	IPY00035822	--	91	Global Call IP	Dialogic® HMP 2.0 and 1.3 do not respond to SIP 407 Proxy Authentication Required messages.
Resolved	IPY00035684	--	91	OA&M	Memory leak in **NCM_GetFamilyDeviceByAUID( )**.
Resolved	IPY00034499	--	89	Conferencing	When running conferencing activity under high load on dual core systems, there is a potential for **dcb_delconf( )** to hang and never return.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00035517	--	89	Fax	The EOLs at the end of the page were incorrect for MR encoding.
Resolved	IPY00035613	--	89	Global Call IP	Dialogic® HMP 1.3 fails to send a BYE message after dropping call on Avaya IP PBX.
Resolved	IPY00035350	--	89	H323 Call Control	Data provided after a 0x00 byte will not be sent in the "Nonstandard Control Information".
Resolved	IPY00035473	--	89	IP	IPM channel hangs after receiving disconnect.
Resolved	IPY00034686	--	89	SIP Call Transfer	Problems with call transfer. GCEV_REQ_XFER event not received in application.
Resolved	IPY00034526	--	89	SNMP	dlgagent.log keeps growing.
Resolved	IPY00034777	--	89	Voice	Problems playing a file from remote source, if the source is disconnected and exception is not handled correctly
Resolved	IPY00033733	--	86	APIC driver	System freezes when running Dialogic® HMP 1.3. Advanced Programmable Interrupt Controller (APIC) driver was improved.
Resolved	IPY00033371	--	86	APIC driver	APIC timer calibration fix.
Resolved	IPY00033738	--	86	Dialogic® HMP	Issuing **dx_stopch( )** with the mode parameter set to EV_NOSTOP on idle channel does not return TDX_NOSTOP event.
Resolved	IPY00032375	--	86	Dialogic® HMP	Orbacus services fail on startup.
Resolved	IPY00034313	--	86	Fax	Time display in fax header of AM or PM is wrong for certain time period.
Resolved	IPY00034215	--	86	Fax	Dialogic® HMP1.3 after EOM fax protocol message is sent to receiving side running Hylafax Application (software-based fax receiver).
Resolved	IPY00034210	--	86	Fax	Synchronization object timeout errors occurring.
Resolved	IPY00033575	--	86	FAX	Dialogic® HMP needs to handle FAX if Nortel Gateway converts to T.38 before ever reaching HMP.
Resolved	IPY00033770	--	86	IP Host	Empty Terminal Control String (TCS) is seen after invoking transfer using H.323. Call routing failure.
Resolved	IPY00033597	--	86	IP Media Session Control (RTP)	Blue Screen of Death (BSOD) fix .

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved	IPY00033322	--	78	Dialogic® HMP	Dialogic® HMP server fails to start one out of three attempts on IBMx205 and Dell 1600 server.
Resolved	IPY00033102	--	78	IP Host	Supervised Transfer fails on Dialogic® HMP 1.3 SU67 --Party B getting GCEV_XFER_FAIL with “reason 5800”
Resolved	IPY00032459	--	78	IP Host	Contact field changes during call.
Resolved	IPY00032378	--	78	IP Host	One of my customers sends REGISTER to proxy, proxy sends back OK but Dialogic® HMP reacts REJECT.
Resolved		36867	76	FAX	Fax disconnect occurs under poor network and dropped packet conditions.
Resolved		36860	76	FAX	Blue Screen of Death (BSOD) occurs after a fax disconnect.
Resolved		36869	76	Media	Issuing **dx_play( )** in ASYNC mode to play a zero length wave file will not return a completion event.
Resolved		36842	76	Media	When playing multiple wav files from a DX_IOTT data structure, the streaming fails and a TDX_ERROR event is generated.
Resolved		36787	76	Media	No completion event (TDX_PLAY or TDX_ERROR) is received when **dx_playiottdata( )** is executed in conjunction with UIO callback functions, and the Read() indicates a file length of zero.
Resolved		36793	73	IP Host	When using SIP protocol, an error occurs at the RTL level when invoking the **gc_InvokeXfer( )** function.
Resolved		36643	72	DCM	Three dm3nk warning messages are received when starting DCM service.
Resolved		36080	72	Demo	Compiling IP Media Server demo results in the following error message: “IPMediaServer.mak not found”
Resolved		36677	72	Fax	Inbound fax call fails. This happen when previous call on the same device is dropped and media devices are disconnected using **gc_SetUserInfo( )**.
Resolved		36372	72	Fax	**fx_sendfax( )** flag with the resolution does not overwrite the resolution set with DF_IOTT.
Resolved		36669	72	HMP	Starting DCM service after disconnecting LAN connection causes a blue screen or system reboot to occur.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved		36647	72	IP Host	When calling an invalid IP address or a valid one that is not answering SIP calls, you have to wait 64 seconds before you can call **gc_DropCall( ) function**.
Resolved		36349	72	IP Host	The PARM_TX_ADJVOL_ and PARM_RX_ADJVOL_ parameters in the ipmlib.h header file are defined via #defines, but have semi-colons after them which makes it difficult to compile the application.
Resolved		36317	72	IP Host	Failure in libgch3r.dll when registering SIP Header Fields in multiple iterations using **gc_util_insert_parm_ref( )** and **gc_SetConfigData( )** on Dialogic® HMP 1.3WIN SU 54.
Resolved		36437	72	OA&M	Toggling DSS start up state between Manual and Semi-Automatic mode also toggles the board download state provided by **NCM_GetSystemState( )** despite services not actually starting or stopping.
Resolved		36523	67	IP	When attempting to route an IP Media device with a Fax device, calling the **dev_Connect( )** API function will fail with -1. At that time, calling **ATDV_LASTERR( )** and **ATDV_ERRMSGP( )** on the IPM device handle will return "no error" and the fax handle will return "unknown error (1)".
Resolved		36280	65	DCM	DCM Help->About menu does not display the current Dialogic® HMP Service Update value.
Resolved		35660	65	DCM	Incorrect instructions for activating a license.
Resolved		36027	65	Documentation	If the user enters a non-default directory when installing the Dialogic® HMP Software, the suffix \Dialogic\HMP is appended to that entry. This needs to be documented.
Resolved		35692	65	Documentation	When an application attempts to set the QoS Packet Loss Alarm level, the system always rounds down to the next lower 10% increment. For example, if you try to set the level to 15%, the system will actually use 10%. If you try to set the level to9%, the system will use 0%. For Jitter, the system rounds down to next lower 5% increment. For example, 8% becomes 5%. This limitation needs to be clearly explained in the documentation.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Resolved		36082	65	Fax	After a receiving a bad checksum on a V.21 HDLC frame, the Fax error handling does not restart a protocol timer. This results in a channel hanging if the sender subsequently disconnects without retrying the frame.
Resolved		36419	65	IP	If **ipm_GetSessionInfo( )** or **ipm_SetParm( )** is called when performing bulk IPcalls, the application may fail under heavy calls (e.g., 240 channels) with a "Synchronization Object Timeout" error.
Resolved		35938	65	Voice	When a small Wave file (e.g., 46 Bytes) is used, no TDX_PLAY event is received until the call is dropped, due to “m_Offset too large” error.
Resolved		35979	61	Global Call IP	H.323 "DONT_CARE" Codec results in "call control library specific failure".
Resolved		35872	61	Global Call IP	Call does not get connected if incoming H.323 setup contains H.245 address information.
Resolved		35693	61	Global Call IP	Gatekeeper registration after un-registration fails.
Resolved		35596	61	Global Call IP	Application cannot access full content of Bearer Capability IE in SETUP message.
Resolved		34952	61	Global Call IP	An application may receive an unexpected GCEV_DISCONNECTED event with a IPEC_Q931TransportError error when making calls.
Resolved		35928	61	Voice	Sound quality is poor when playing OKI ADPCM encoded file.
Resolved		34936	61	Voice	Call progress analysis incorrectly deciphers certain answering machine messages as positive voice detection (PVD).
Resolved		35585	54	Global Call IP	Secondary Call Reference Number (CRN) is not provided on party C in a case of attended SIP transfer.
Resolved		35520	54	Global Call IP	When using H.323, calling **gc_Extension( )** with IPPARM_PHONELIST may cause an application failure.
Resolved		35434	54	HMP	On machines without an APIC timer, DCM may not be able to restart Dialogic® HMP after a successful start/stop operation without a system reboot.
Resolved		35545	54	IPML	Function **ipm_StartMedia( )** does not return the IPMEV_STARTMEDIA event when using Windows XP.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR 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		35116		DCM	Pop-up error messages occur when "Country" tab is selected in DCM. ***Workaround:*** Do not select Country tab. It is not supported or required.
Known (permanent)		--		Demo	Demos are intended to be run using a maximum of 4 channels.
Known		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. ***Workaround:*** 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 (permanent)		--		Fax	For fax applications, the header file *srllib.h* must appear before the header file *faxlib.h* in the #include directive.
Known		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. ***Workaround:*** None is required. Ignore this message.
Known		33789		Fax	If using Debug View when making T.38 fax calls, there may be an occasional "FC3_ReadFrame call FAILED" error message. ***Workaround:*** None required as there is no error. Ignore the message.
Known		30626		Fax	The TO: field in the fax header for polled faxes is empty. ***Workaround:*** None
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.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
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	Global Call applications using T.38 should call **gc_SetUserInfo(IPPARM_T38_CONNECT)** after GCEV_OFFERED, but before the next GC function call, for T.38 only calls. Otherwise, the call will fail and the application will get a GCEV_TASKFAIL event.
Known (permanent)		--		Global Call IP	The **ipm_SendRFC2833SignalToIP( )** function is not supported on Dialogic® HMP. The application cannot use this API to generate RFC2833 digits on a call. Use the voice device to dial digits and set up RFC2833 transfer mode in a call.
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. • If the application is using IPML for call control, the application should stop the IPML devices using **ipm_Stop( )**. Any open devices should then be closed using **ipm_Close( )**. • 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).

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
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.
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.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Known		34894		Global Call IP	An application may fail in libgc.dll if gc board line device is closed after issuing **gc_SetConfigData( )** for streaming status extension events. ***Workaround:*** Do not close the board device until the application is ready to exit.
Known		34740		Global Call IP	False "RTP Timeout" alarms may occur while running 32 channels. ***Workaround:*** None
Known		34249		Global Call IP	Event viewer may display dm3nk warning messages during startup. ***Workaround:*** Ignore the warning message as there is no impact to HMP.
Known (permanent)		32740		HMP	After several cycles of starting and stopping the System Service, or a single device (HMP), the System Service may hang if any Debug View[1] client (DBMON or Debug View) is active. 1. Copyright © 1998-2005 Mark Russinovich
Known (permanent)		--		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)		--		HMP	HMP may not run properly on a computer if Advanced Configuration and Power Interface (ACPI) is installed. See the *Apic Timer and HMP* information in the *Compatibility Notes* section.
Known (permanent)		--		HMP	The HMP system does not operate with Dialogic® board level products, or with Dialogic System Release software installed on the same machine.
Known (permanent)		--		HMP	When using HMP, advanced power management features, such as hibernation, will not be supported.
Known		35039 (35009, 35004, 35003, 35002, 34958)		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. ***Workaround:*** Reboot the system after any post-HMP installation IP address change.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Known		35034		HMP	If the ssp.mlm file is inadvertently deleted or moved by the user or another application, the event viewer error message will contain the wrong directory path (\Dialogic\data instead of \Intel\HMP\data). ***Workaround:*** Interpret "\dialogic" as "\intel\hmp" in the message.
Known		35033		HMP	Occasionally, during repetitive downloads of 240 channel configurations, HMP download will fail. ***Workaround:*** Repeat download.
Known		34191		HMP	When stopping HMP services, the following event viewer error message may be displayed: dwCheckPoint =<n>, where n is a positive integer ***Workaround:*** None is required. Ignore this message.
Known		30386		HMP	Voice/Fax devices and IPML devices return different Device Family and Bus Mode values when used with the **xxx_getctinfo( )** function call on an HMP 1.1 platform. ***Workaround:*** None
Known		35060		Installation	The installation does not exit gracefully if the process is cancelled during the file copy stage. ***Workaround:*** Do not cancel the installation once the process has started.
Known		34690 (34255)		Installation	A warning message from the operating system about HMP drivers not being digitally signed may be displayed during installation. ***Workaround:*** Ignore the warning and continue installation. For information on disabling the warning message, please refer to Section 4.5, "Disabling the Windows Driver Signing Check" in the HMP Software Installation Guide.
Known		34992		Licensing	When a license is activated on a system that previously had activated licenses with more capability, selecting the Restore Defaults option from the DCM Device menu shows all the associated PCD files. If the older (higher capability) PCD file is selected, download will fail. ***Workaround:*** Select the PCD file associated with the license correctly activated.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
Known		34976		Licensing	The Licensing CLI does not recognize an error when incorrect syntax is used for the -D-F options and reactivates the current license. This typically occurs with long directory or file names. ***Workaround:*** Use short names for directories and files.
Known		34819		Licensing	If a PCD file is selected that exceeds the capabilities authorized by the activated license, no error message is provided when the download fails. ***Workaround:*** Only select PCD files of equal or less capability than the activated license.
Known		30738		Licensing	For the Command Line License Manager, download fails if relative paths are used for the -D option when the license is activated. ***Workaround:*** Use the absolute path name or place license files in the Program Files\Intel\HMP\data directory and do not use the -D option.
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 will not terminate after the time specified by the DX_MAXTIME termination parameter has elapsed.

Table 1. Issues Sorted by Type, Dialogic® HMP Software Release 1.3WIN

Issue Type	Defect No.	PTR No.	SU No.	Dialogic® Product or Component	PTR Description
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.
Known		34242		Voice	Under heavy load, the TDX_UNDERRUN event may not occur when stream data is played out. ***Workaround:*** None is required. The application should recover.
Known		32636		Voice	A very low percentage of DTMF tone leakage was observed on low bit rate coders when the **ipm_ReceiveDigits( )** was used with the DTMF transfer mode set to Out-of-Band. ***Workaround:*** Most call control applications will not be affected because, during Out-of-Band transfer mode, H.245 UII messages are used to relay DTMF event information. These H.245 UII messages have higher priority.

2.2 Compatibility Notes

Adjusting the TimedWait State

Running ONLY call control on 10 or more timeslots 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 2000 *http://support.microsoft.com/default.aspx?scid=kb;en-us;120642*
- 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*

Echo Cancellation With Continuous Speech Processing

For HMP Continuous Speech Processing (CSP), the Echo Cancellation parameter default value is set to OFF (echo cancellation disabled). The application should keep this parameter value set to OFF.

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

Echo Cancellation With Conferencing

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

For additional information about Conferencing, refer to the *Audio Conferencing API for Linux and Windows Operating Systems Library Reference* and *Audio Conferencing API for Windows Operating Systems Programming Guide*.

Configuring UDP/RTP Port Range

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\Intel\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\Intel\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\Intel\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\Intel\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\Intel\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\Intel\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.

H.323 T.38 HMP Interoperability with theDialogic® PBX-IP Media Gateway

HMP has not been tested with the current version of the Dialogic PBX-IP Media Gateway (Version 4.0 Service Update 2) when using H.323 T.38 Fax.

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.

Documentation Updates 3

The following sections, which correspond to the top level categories used in the online documentation navigation page, include updates to the documentation.

3.1 Release Documentation

This section contains updates to the following documents:

- Dialogic® Host Media Processing Software Release 1.3WIN Release Guide

3.1.1 Dialogic® Host Media Processing Software Release 1.3WIN Release Guide

The following information needs to be added to the Dialogic® Host Media Processing Software Release 1.3WIN Release Guide:

- In **Section 2.2, "Basic Software Requirements"**, add the Windows 2000 Advanced Server, SP4 to the list of operating systems.
- In **Section 2.3.4 "Configurations Tested"**, Table 3 and Table 5 have been changed to reflect the increase of supported fax channels from 32 to 120.

Table 3. Resource Configurations Tested

Configuration	RTP	Enhanced RTP	Voice	Conferencing (DCB)	Fax	Speech
IVR	240	0	240	0	0	0
IVR w/Speech	200	0	200	0	0	200
IVR/Conferencing	60	12	60	0	6	60
Unified Messaging	120	120	120	0	0	0
Unified Messaging	120	0	120	0	0	0
Fax	120	40	120	0	32	0
Fax	96	64	96	24	8	96
Fax	120	0	120	0	120	0

Configuration	RTP	Enhanced RTP	Voice	Conferencing (DCB)	Fax	Speech
Conferencing	240	0	100	240	0	0
Conferencing	96	64	48	96	0	0
Gateway	23	11	23	0	4	23

Table 5. Media Resources

Media Resource	Resource Code	Number of Resources Available	Resource Description
RTP G.711	r	0-240	Streaming digitized voice over RTP using G.711 coder
Voice	v	0-240	Basic voice ports that allow you to control volume, record with Automatic Gain Control (AGC), DTMF, and user-defined tone detection.
Enhanced RTP	e	0-120	Enhances the resource by supporting digitized voice over RTP using G.723.1, G.729A, and G.729AB coders.
Conferencing	c	0-240	Conferences parties using advanced features such as coach/pupil mode, tone clamping, and active talker notification.
Speech Integration (Continuous Speech Processing)	s	0-240	Speech integration capabilities that allow you to integrate Dialogic HMP Software with speech engines for Automatic Speech Recognition (ASR) and Text-to-Speech (TTS) support using the Continuous Speech Processing (CSP) APIs.
T.38 Fax Termination	f	0-120	Provides support for T.38 fax origination and termination sessions.

- In **Section 2.4, "Ordering the Product"**, the number of fax resources available, as listed in Table 5, should be 0-120.
- In **Section 4.7, "Audio Conferencing (DCB) API Library"**, the second sentence in the ninth bullet in the **Features** section is deleted. The feature description should read:
  - Active talker indication to determine which conferees in any given conference are currently talking.
- In **Section 4.9, "API Functions Not Supported"**, the only Audio Conferencing API function not supported is **dcb_GetAtiBitsEx( )**.

3.2 Installation Documentation

This section contains updates to the following documents:

- Dialogic® Host Media Processing Software Release 1.3WIN Software Installation Guide

3.2.1 Dialogic® Host Media Processing Software Release 1.3WIN Software Installation Guide

Update to section 2.1, "Prerequisites for Software Installation," page 15

The following requirement should be added to the list, after Windows operating system:

- The Dynamic Host Configuration Protocol (DHCP) Windows Client Service which is part of the Windows operating system must be enabled in order to obtain the IP address before the Dialogic HMP system starts. It is also recommended that the DHCP Client startup type be set to "Automatic." If it is set to manual mode, the HMP system will not start until DHCP service is started manually.

Update to section 2.3, "Performing a Full Install of the Software," page 25

In the bullet about checking the IP address, the following information should be added:

- The IP address used by HMP is obtained by the DHCP Client Windows Service before the HMP system starts. Be sure DHCP Windows Service is enabled on your system.

Update to section 2.4, "Performing an Update Install," page 26

In the bullet about checking the IP address, the following information should be added:

- The IP address used by HMP is obtained by the DHCP Client Windows Service before the HMP system starts. Be sure DHCP Windows Service is enabled on your system.

Update to section 2.5, "Performing a Silent Install," page 29

In the bullet about checking the IP address, the following information should be added:

- The IP address used by HMP is obtained by the DHCP Client Windows Service before the HMP system starts. Be sure DHCP Windows Service is enabled on your system.

The Installation Guide has been replaced by a new version in the electronic bookshelf. The new version of the Installation Guide includes updated information about:

- Upgrading from Dialogic HMP Software Release 1.1WIN
- Upgrading from an earlier version of Dialogic HMP Software Release 1.3WIN
- Installing Dialogic HMP Software Release 1.3WIN for the first time

In Section 2.3, "Performing a Full Install of the Software," add the following Note to Step 5:

Note: The \Dialogic\HMP path sequence will be appended to the destination folder you select.

3.3 Operating Software Documentation

This section contains updates to the following documents:

- Dialogic Host Media Processing Software Release 1.3WIN Administration Guide
- SNMP Agent Software for Dialogic Host Media Processing Software for Windows Administration Guide
- Native Configuration Manager API for Windows Operating Systems Library Reference
- Native Configuration Manager API for Windows Operating Systems Programming Guide

3.3.1 Dialogic Host Media Processing Software Release 1.3WIN Administration Guide

Replace the existing CLI procedure in Chapter 7, “Switching to a Different License” with the following (Section 7.3, “Using the CLI Version of the License Manager”):

Using the CLI version of the License Manager, the process for switching to a different license is as follows:

1. Open a DOS command prompt window.

2. First, you must locate the license file you want to switch to and look at the configuration details to make sure this license file is the one you want to start using. Type one of the following commands (the first one is the short form of the command and the second is the long form of the command):

   LicenseManager -s -D <directory> -F <file>

   LicenseManager --showdetails --directory <directory> -- file <file>

   where <directory> is the directory in which the license file you want to switch to is located and <file> is the name of the license file you want to switch to.

Note: If you use the command LicenseManager -s (or --showdetails) without any parameters, it will show the details of the active license by default. However, since you are trying to switch from the current active license to a different license, you must use the directory and file parameters to specify the license you want to switch to.

The details of the license you are going to switch to will display. After you examine the details and confirm that this is the license you want to activate, go to the next step.

3. To activate the license you just verified, type one of the following commands (the first one is the short form of the command and the second is the long form of the command):

   LicenseManager -a -D <directory> -F <file>

   LicenseManager --activate --directory <directory> --file <file>

   where <directory> is the directory in which the license file was put and <file> is the license file's name.

   The license you specified is now active.

   Note: There can be multiple license files on disk, but only one of them can be active at a particular time. The activate command lets you activate a particular license file.

4. Invoke the DCM GUI. From the Start menu, select **Configuration Manager- DCM** from the HMP program folder. If you have not started the system yet, start the HMP system. If HMP is already running, go to the next step.

5. On the DCM Main Window click the **Stop Service** option from the **Service** pull-down menu or click the **Stop Service** icon.

6. From the DCM Action pull-down menu, click **Restore Defaults**. DCM will detect HMP Software and automatically use the new license.

Update to Section 9.4, "Stopping HMP". Add the following information after item 2:

When HMP is stopped via DCM, the Dialogic System Service (DSS) and all other Dialogic services will be stopped. When HMP is started again via DCM, the DSS and all other Dialogic services will be started automatically.

Note: The Boardserver service (BoardServer.exe service for the SNMP Agent Software functionality) is not a part of the DCM start-up sequence. If the Boardserver service was running prior to stopping HMP, 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.

Note: The **NCM_StopSystem()** function cannot be used to stop Dialogic services. This function is intended to stop the HMP virtual device.

3.3.2 SNMP Agent Software for Dialogic Host Media Processing Software for Windows Administration Guide

There currently are no changes to this document.

3.3.3 Dialogic Host Media Processing Diagnostic Guide

- Added a description of the `its_sysinfo` utility.

3.3.4 Native Configuration Manager API for Windows Operating Systems Library Reference

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_StartSystem( )** function

The note in the Description section is invalid and should be removed. The function description should read as follows:

The **NCM_StartSystem( )** function starts the Dialogic system service and starts all boards.

If your system is running in manual mode, the **NCM_StartSystem( )** function starts the system service and starts all boards in the system. If your system is running in semi-automatic mode, the Dialogic system service will run uninterrupted and a call to the **NCM_StartSystem( )** function will start all boards. Use the **NCM_GetSystemState( )** function to determine whether or not the system service is running.

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 start-up 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.

3.3.5 Native Configuration Manager API for Windows Operating Systems Programming Guide

There currently are no changes to this document.

3.4 Development Software Documentation

This section contains updates to the following documents:

- Audio Conferencing API for Linux and Windows Operating Systems Library Reference
- Audio Conferencing API for Linux and Windows Operating Systems Programming Guide
- Continuous Speech Processing API for Linux and Windows Library Reference
- Continuous Speech Processing API for Linux and Windows Programming Guide
- Global Call API for Host Media Processing Library Reference
- Global Call API for Host Media Processing on Windows Programming Guide
- Global Call IP for Host Media Processing Technology Guide
- Device Management API For Windows and Linux Operating Systems Library Reference
- Fax Software Reference for Windows
- IP Media Library API for Host Media Processing Library Reference
- IP Media Library API for Host Media Processing Programming Guide
- Standard Runtime Library API for Linux and Windows Operating Systems Library Reference
- Standard Runtime Library API for Windows Operating Systems Programming Guide
- Voice API for Host Media Processing Library Reference
- Voice API for Host Media Processing Programming Guide

3.4.1 Audio Conferencing API for Linux and Windows Operating Systems Library Reference

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

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

MSG_ACTID (Active Talker Identification)

Enables or disables Active Talker Identification (or Notification). Possible values are ACTID_ON or ACTID_OFF. ACTID_ON is the default. This parameter does not enable or disable the active talker *feature*, which is always enabled. It only disables the *notification* to the application program. The active talker *feature* sums the 3 most active talkers in a

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

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

3.4.2 Audio Conferencing API for Linux and Windows Operating Systems Programming Guide

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

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

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

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

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

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

Do the following to implement background music in a conference:

- Create a three-party conference, where one party is the music resource.
- When you add music to the conference, make the following parameter settings. Set its party attributes so that it uses transmit-only mode. That is, for the conference party that transmits music, enable the MSPA_MODEXMITONLY attribute in the MS_CDT data structure **chan_attr** field.

- The Conferencing AGC Noise Level Lower Threshold may need to be adjusted. This parameter is set at -40 dBm by default and filters out any signals below this level. If the background music levels are low, the music may not be summed into the conference by the active talker feature, or it may come in and out. In this case, the AGC Noise Level Lower Threshold should be reduced by adding or changing the CSUMS_AGC_low_threshold (0x3B1F) parameter in the configuration file for the board and re-starting the board. For more information on this parameter, see the applicable configuration guide; e.g., *Dialogic® on DM3 Architecture Configuration Guide*, or the configuration guide for Dialogic® Host Media Processing (HMP) software.

3.4.3 Continuous Speech Processing API for Linux and Windows Library Reference

There currently are no changes to this document.

3.4.4 Continuous Speech Processing API for Linux and Windows Programming Guide

There currently are no changes to this document.

3.4.5 Global Call API for Host Media Processing Library Reference

There currently are no changes to this document.

3.4.6 Global Call API for Host Media Processing on Windows Programming Guide

There currently are no changes to this document.

3.4.7 Global Call IP for Host Media Processing Technology Guide

The following information needs to be added to the Global Call IP for Host Media Processing Technology Guide:

Corrections to Code Examples

The code example in Section 4.10.1, "Sending an INFO Message" (page 163), and both code examples in Section 4.10.4, "Responding to an INFO Message" (page 166), have incorrect parameters in the gc_Extension() function call. In all three cases, the first two parameters are incorrect and should identify a CRN rather than a device handle. The correct line of code would read:

```
retval = gc_Extension(GCTGT_GCLIB_CRN, crn, IPEXTID_SENDMSG, parmblkp,
&retblkp, EV_ASYNC);
```

In Section 7.2.17, “gc_OpenEx() Variancesfor IP,” the following note should be included:

Note: Applications should avoid closing and re-opening devices multiple times. Board devices and channel devices should be opened during initialization and should remain open for the duration of the application.

Fast Start Coder

The following information is added to the Global Call IP for Host Media Processing Technology Guide:

Retrieving Coder Information from Fast Start Call Offers

Any call offer that is received can potentially contain “fast start” coder information, in the form of an SDP offer in an INVITE request when using SIP or a fastStart element in a Setup message when using H.323. The IP call control library handles any such fast start coder information internally to begin the coder negotiation process, but it may be useful to the application to access the offered coder information, as well. The call control library can be configured at start-up to provide application access to fast start coder information for SIP or H.323 or both. When this access is enabled and the library accepts a fast start offer, the extra data associated with the GCEV_OFFERED event that is sent to the application will contain one or more additional parameter elements to convey the coder information that was contained in the offer.

Enabling Access to Fast Start Coder Information

Application access to fast start coder information is a feature that can be disabled or enabled independently for the SIP and H.323 protocols at the time the **gc_Start( )** function is called.

The **INIT_IPCCLIB_START_DATA( )** and **INIT_IP_VIRTBOARD( )** functions, which must be called before the **gc_Start( )** function, populate the IPCCLIB_START_DATA and IP_VIRTBOARD structures, respectively, with default values. The default values of the sip_msginfo_mask and h323_msginfo_mask fields in the IP_VIRTBOARD structure disable all optional message information access features, including access to fast start coder information. The default values of these data structure fields must be overridden with appropriate values for each IPT board device on which access needs to be enabled. For each of the two message information mask fields, the value that the application sets will typically be an OR of two or more defined mask values as described in the IP_VIRTBOARD data structure below.

The defined mask values that are used to enable access to fast start coder information are:

IP_SIP_FASTSTART_CODERS_IN_OFFERED
: enables application access to coder information contained in SDP offers in SIP INVITE requests

IP_H323_FASTSTART_CODERS_IN_OFFERED
: enables application access to coder information contained in fastStart elements in H.323 Setup messages

Note that it is not possible to toggle the fast start coder information access between enabled and disabled states without stopping and restarting the system via **gc_Stop( )** and **gc_Start( )**.

The following code snippet shows how an application might initialize two virtual boards to enable basic message information access and access to fast start coder information for both SIP and H.323 protocols.

```
.
.
.
INIT_IPCCLIB_START_DATA(&ipcclibstart, 2, ip_virtboard);
INIT_IP_VIRTBOARD(&ip_virtboard[0]);
INIT_IP_VIRTBOARD(&ip_virtboard[1]);
ip_virtboard[0].sip_msginfo_mask =
         IP_SIP_MSGINFO_ENABLE | IP_SIP_FASTSTART_CODERS_IN_OFFERED;
         /* override SIP default to enable access to message info and faststart coder info*/
ip_virtboard[1].sip_msginfo_mask =
         IP_SIP_MSGINFO_ENABLE | IP_SIP_FASTSTART_CODERS_IN_OFFERED;
         /* override SIP default to enable access to message info and faststart coder info*/
ip_virtboard[0].h323_msginfo_mask =
         IP_H323_MSGINFO_ENABLE | IP_H323_FASTSTART_CODERS_IN_OFFERED;
         /* override H.323 default to enable access to message info and faststart coder info*/
ip_virtboard[1].h323_msginfo_mask =
         IP_H323_MSGINFO_ENABLE | IP_H323_FASTSTART_CODERS_IN_OFFERED;
         /* override H.323 default to enable access to message info and faststart coder info*/
.
.
.
```

Accessing Fast Start Coder Information

The Global Call IP call control library includes coder information in the extra data associated with a GCEV_OFFERED event when all of the following conditions are true:

- The library was started with the fast start coder information option enabled for the appropriate protocol.
- The fast start mode is enabled.
- The call offer is a fast start offer; that is, it includes an SDP offer (SIP) or fastStart element (H.323).
- The SDP offer or fastStart element specifies at least one coder that the library supports.

When all of these conditions are true, the extra data associated with the GCEV_OFFERED event will be a GC_PARM_BLK that contains one or more parameter elements of the following type:

IPSET_CALLINFO
    IPPARM_OFFERED_FASTSTART_CODER
        - value = IP_CAPABILITY data structure

Each such parameter element reflects a coder specification that was contained in the call offer. If the offer contains multiple coder specifications, the order of the parameter elements in the parameter block reflects the order of the specifications in the offer message. This order reflects the remote endpoint's coder preference, with the first specification being the most preferred and the last specification being the least preferred. If any coder properties were left unspecified by the remote end, the matching fields in the corresponding IP_CAPABILITY structure will be filled in with the value GCCAP_dontCare.

If any of the four conditions described above are not true, there will be no IPSET_CALLINFO / IPPARM_OFFERED_FASTSTART_CODER parameter element in the parameter block associated with the GCEV_OFFERED.

When the IP_CAPABILITY data structure is used to convey fast start coder information, the direction field of the structure uses the following special value defines:

IP_CAP_DIR_RMTRECEIVE
: Remote coder was specified to be Receive-only.

IP_CAP_DIR_RMTRTPINACTIVE
: Remote coder was specified with RTP port 0, which is used in SIP to inactivate RTP streaming. Only supported when using SIP.

IP_CAP_DIR_RMTRTPRTCPINACTIVE
: Remote coder was specified with RTP address 0.0.0.0, which is used in SIP to inactivate both RTP and RTCP. Only supported when using SIP.

IP_CAP_DIR_RMTTRANSMIT
: Remote coder was specified to be Transmit-only.

IP_CAP_DIR_RMTTXRX
: Remote coder was specified to be capable of both Transmit and Receive.

The following data structure has been added to the Global Call IP for Host Media Processing Technology Guide:

IP_VIRTBOARD

```
typedef struct
{
    unsigned short            version;
    unsigned int              total_max_calls;
    unsigned int              h323_max_calls;
    unsigned int              sip_max_calls;
    IP_ADDR                   localIP;
    unsigned short            h323_signaling_port;
    unsigned short            sip_signaling_port;
    void                      *reserved;
    unsigned short            size;
    unsigned int              sip_msginfo_mask;
    unsigned int              sup_serv_mask;
    unsigned int              h323_msginfo_mask;
    MIME_MEM                  sip_mime_mem
    unsigned short            terminal_type
    IP_ADDR                   outbound_proxy_IP
    unsigned short            outbound_proxy_port;
    char *                    outbound_proxy_hostname;
    EnumSIP_Enabled           E_SIP_tcpenabled;
    EnumSIP_TransportProtocol E_SIP_OutboundProxyTransport;
    EnumSIP_Persistence       E_SIP_Persistence;
    unsigned short            SIP_maxUDPmsgLen;
    EnumSIP_TransportProtocol E_SIP_DefaultTransport;
    EnumSIP_RequestRetry      E_SIP_RequestRetry;
    EnumSIP_Enabled           E_SIP_OPTIONS_Access;
    unsigned int              sip_registrar_registrations;
}IP_VIRTBOARD;
```

Description

The IP_VIRTBOARD data structure is used to store configuration and capability information about an IPT board device that is used when the device is started. An array of IP_VIRTBOARD structures (one for each virtual board in the system) is referenced by the IPCCLIB_START_DATA structure, which is passed to the **gc_Start( )** function. The IP_VIRTBOARD structure must be initialized to default values by the **INIT_IP_VIRTBOARD()** initialization function; those default values can be overridden by the application before calling **gc_Start( )**.

Field Descriptions

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

version
: The version of the structure. The correct version number is populated by the **INIT_IP_VIRTBOARD( )** function and should not be overridden by the application.

total_max_calls
: The maximum total number of IPT devices that can be open concurrently using either the H.323 or SIP protocol. Values range from 1 to 2016 (IP_CFG_MAX_AVAILABLE_CALLS). The default value is 120.

h323_max_calls
: The maximum number of IPT devices that can be used for H.323 calls. Possible values are in the range 1 to 2016 (IP_CFG_MAX_AVAILABLE_CALLS). The default value is 120.

sip_max_calls
: The maximum number of IPT devices that can be used for SIP calls. Possible values are in the range 1 to 2016 (IP_CFG_MAX_AVAILABLE_CALLS). The default value is 120.

localIP
: The local IP address of type IP_ADDR. See the reference page for IP_ADDR on page 456.

h323_signaling_port
: The H.323 call signaling port. Possible values are a valid port number or IP_CFG_DEFAULT. The default H.323 signaling port is 1720.

sip_signaling_port
: The SIP call signaling port. Possible values are a valid port number or IP_CFG_DEFAULT. The default SIP signaling port is 5060.

reserved
: For library use only

size
: For library use only

sip_msginfo_mask (structure version ≥ 0x101 only)
: Enables and disables access to SIP message information. Access is disabled by default. The following mask values, which may be OR'ed together, are defined to enable these features:
  - IP_SIP_MSGINFO_ENABLE – enable access to supported SIP message information fields
  - IP_SIP_MIME_ENABLE – enable sending and receiving of SIP messages that contain MIME information

- IP_SIP_FASTSTART_CODERS_IN_OFFERED – enable receiving coder information from a SIP "FastStart" call offer via the GCEV_OFFERED event

sup_serv_mask (structure version ≥ 0x102 only)

Enables and disables the call transfer supplementary service. The service is disabled by default. Use the following value to enable the feature:

- IP_SUP_SERV_CALL_XFER – enable call transfer service

h323_msginfo_mask (structure version ≥ 0x103 only)

Enables and disables reception of H.323 message information. Access is disabled by default. The following mask values, which may be OR'ed together, are defined to enable the features:

- IP_H323_ANNEXMMSG_ENABLE – Enable reception of H.323 Annex M tunneled signaling messages in H.225 messages
- IP_H323_MSGINFO_ENABLE – enable access to H.323 message information fields
- IP_H323_FASTSTART_CODERS_IN_OFFERED – enable receiving coder information from an H.323 FastStart call offer via the GCEV_OFFERED event

sip_mime_mem (structure version ≥ 0x104 only)

Sets the number and size of buffers that will be allocated for the MIME memory pool when the SIP MIME feature is enabled (no buffers are allocated if the feature is not enabled). The default values indicated below are set by the **INIT_MIME_MEM( )** macro, which is called by the **INIT_IP_VIRTBOARD( )** initialization function. The MIME_MEM data structure is defined as follows:

```
typedef struct
{
     unsigned short   version;     /* Version set by INIT_MIME_MEM   */
     unsigned int     size;        /* Default = 1500                 */
     unsigned int     number;      /* Default = (sip_max_calls * 5)  */
}MIME_MEM;
```

terminal_type (structure version ≥ 0x104 only)

Sets the Terminal Type for the virtual board which will be used during RAS registration (H.323 terminal type) and during Master Slave determination (H.245 terminal type). The value may only be changed from the default that is set by the **INIT_IP_VIRTBOARD( )** initialization function before calling **gc_Start( )**. Unsigned shorts from 0 to 255 are valid values, but the specific values 0 and 255 are reserved and will result in the terminal type being set to the default. Values larger than 255 are truncated to 8 bits. The following symbolic values are defined:

- IP_TT_GATEWAY (Default) – Value = 60, for operation as terminal type Gateway
- IP_TT_TERMINAL – value = 50, for operation as terminal type Terminal

outbound_proxy_IP (structure version ≥ 0x105 only)

Sets the IP address of the SIP outbound proxy, which is used instead of the original Request URI for outbound SIP requests. The default value is 0, which disables outbound proxy unless the outbound_proxy_hostname field is set to a non-NULL name.

outbound_proxy_port (structure version ≥ 0x105 only)

Sets the port number of the SIP outbound proxy specified by outbound_proxy_IP. The default value is 5060, which is the same as the default SIP signaling port number.

outbound_proxy_hostname (structure version ≥ 0x105 only)

Sets the specified hostname as the SIP outbound proxy instead of a hard-coded IP address. If outbound_proxy_IP is set to 0, this hostname is resolved as the outbound proxy address. If outbound_proxy_IP is set to an IP address, this field is ignored and outbound_proxy_IP and outbound_proxy_port are used instead. The default value is NULL.

E_SIP_tcpenabled (structure version ≥ 0x106 only)

Enables the handling of incoming SIP messages that use TCP (received on the port number specified in sip_signaling_port), and the ability to specify TCP transport for SIP requests. The following symbolic values are defined:

- ENUM_Disabled (default) – disable TCP transport support (use default UDP transport)
- ENUM_Enabled – enable TCP transport support for incoming and outgoing messages

E_SIP_OutboundProxyTransport (structure version ≥ 0x106 only)

Selects the default transport protocol for SIP requests when an outbound proxy has been set up via the outbound_proxy_IP or outbound_proxy_hostname field (assuming that TCP is enabled via E_SIP_tcpenabled). The following symbolic values are defined:

- ENUM_TCP – use TCP protocol for the outbound proxy; if this value is set when TCP is not enabled or when TCP is enabled but no SIP proxy is configured, **gc_Start( )** returns an IPERR_BAD_PARM error
- ENUM_UDP (default) – use UDP protocol for the outbound proxy

E_SIP_Persistence (structure version ≥ 0x106 only)

Sets the persistence of TCP connections (assuming that TCP has been enabled via E_SIP_tcpenabled). This field has no effect on whether TCP is used for requests; it only affects the connections that are made when TCP is actually used. The following symbolic values are defined:

- ENUM_PERSISTENCE_NONE – no persistence; TCP connection is closed after each request
- ENUM_PERSISTENCE_TRANSACT – transaction persistence; TCP connection is closed after each transaction
- ENUM_PERSISTENCE_TRANSACT_USER (default) – user persistence; TCP connection is maintained for the lifetime of the "user" of the transaction (the CallLeg, for example)

SIP_maxUDPmsgLen (structure version ≥ 0x106 only)

Sets the maximum size for UDP SIP requests; above this threshold, the TCP transport protocol is automatically used instead of UDP (assuming that TCP is enabled via E_SIP_tcpenabled). The default value is 1300 (as recommended by RFC3261). Value may be set to 0 or VIRTBOARD_SIP_NOUDPMSGSIZECHECK to disable the size checking and reduce the message processing overhead.

E_SIP_DefaultTransport (structure version ≥ 0x106 only)

Sets the default transport protocol that is used when there is no proxy set (assuming that TCP is enabled by E_SIP_tcpenabled). The application can override the default for a particular request by explicitly specifying the transport protocol with a "transport= " header parameter. The following symbolic values are defined:

- ENUM_TCP – use TCP unless ";transport=udp" is set by application; if this value is set when TCP is not enabled, **gc_Start( )** returns an IPERR_BAD_PARM error
- ENUM_UDP (default) – use UDP unless ";transport=tcp" is set by application

E_SIP_RequestRetry (structure version ≥ 0x107 only)

Sets the behavior that the SIP stack follows when a particular address-transport combination has failed for a SIP request; this may be a UDP failure after multiple retries or a TCP failure. The following symbolic values are defined:

- ENUM_REQUEST_RETRY_ALL (default) – there will be a retry if the DNS server has provided a list of IP addresses with transports, and there will also be a retry on the last (or only) address if the transport was TCP and the failure reason qualifies for retry

- ENUM_REQUEST_RETRY_DNS – there will be a retry if the DNS server has provided a list of IP addresses with transports
- ENUM_REQUEST_RETRY_FORCEDTCP – there will be a retry if the DNS server has provided a list of IP addresses with transports, and there will also be a retry on the last (or only) address if the transport was forced to be TCP because of message length and the failure reason qualifies for retry
- ENUM_REQUEST_RETRY_NONE – there will be no retry on request failure

E_SIP_OPTIONS_Access (structure version ≥ 0x108 only)

Enables application access to incoming OPTIONS, and the ability to send OPTIONS requests. The following symbolic values are defined:

- ENUM_Disabled (default) – disable application access to OPTIONS messages
- ENUM_Enabled – enable application access to OPTIONS messages

sip_registrar_registrations (structure version ≥ 0x109 only)

Specifies the number of unique SIP registrations that can be created. A unique registration is defined as a unique Address Of Record/Registrar pair, so registering the same AOR on a different Registrar is counted as a second unique registration. The range for this field is 1 to 10000. The default value is sip_max_calls (120).

Distinguishing Between 180 and 183 Responses in GCEV_ALERTING

The following information is added to the Global Call IP for Host Media Processing Technology Guide:

SIP Provisional (1xx) Responses

RFC 3261 defines five provisional messages (also called informational messages) that may be sent to the calling party when the server at the called party is performing some further action and does not yet have a definitive response. One of these provisional messages, the 100 Trying message, is uniquely reported to the calling application via the maskable GCEV_PROCEEDING event type. The other four provisional messages, which have response codes in the 18x range, are all reported to the calling application via the same Global Call event type, GCEV_ALERTING. This section describes the mechanisms that Global Call provides to allow applications to differentiate among the 18x provisional responses, which include:

- 180 (Ringing)
- 181 (Call Is Being Forwarded)
- 182 (Queued)
- 183 (Session Progress)

Note: RFC 3261 indicates that the server for the called party may issue more than one 182 Queued response to update the caller about the status of the queued call, but the call control library only generates a GCEV_ALERTING event for the **first** 182 Queued response for a given call.

For all provisional messages, the primary content is the Status-Code in the response's Status-Line, and the technique for retrieving this information is described in the Retrieving Status-Code for 18x Provisional Responses section below.

RFC 3261 specifies that 182 and 183 responses may optionally contain additional information about the call status in the Reason-Phrase of the message's Status-Line. The technique for retrieving this information is described in the Retrieving Reason-Phrase from 182 and 183 Provisional Responses section below.

RFC 3261 also specifies that 183 responses can optionally contain more details about the call progress in message header fields or the message body. Applications can retrieve this information using generic access mechanisms.

Retrieving Status-Code for 18x Provisional Responses

When using SIP, each GCEV_ALERTING event will have an associated GC_PARM_BLK that contains the specific status code for the 18x provisional response message in a parameter element of the following type:

IPSET_SIP_RESPOSNSE_CODE
    IPPARM_RECEIVED_RESPONSE_STATUS_CODE
    - value = 3-digit integer retrieved as Status-Code from Status-Line of the received provisional message

Retrieving Reason-Phrase from 182 and 183 Provisional Responses

The mechanism provided for retrieving the Reason-Phrase for 182 and 183 provisional response messages is an extension of the generic mechanism for accessing SIP header fields, even though the Reason-Phrase is not technically a header field.

Applications must first register to receive the Reason-Phrase. This registration only needs to be performed once for a board device, and may be performed at any time during the life of an application.

To register to receive the Reason-Phrase, the application first constructs a GC_PARM_BLK that contains the following element:

IPSET_CONFIG
    IPPARM_REGISTER_SIP_HEADER
    - value = "Reason-Phrase"

The application then calls **gc_SetConfigData( )** with this GC_PARM_BLK to register for reception of all the header fields that are identified in the parameter block.

When the Global Call library receives a 182 or 183 provisional response, it generates a a GCEV_ALERTING event that has an associated GC_PARM_BLK to contain extra data about the event. If the application has previously registered to receive the Reason-Phrase, this GC_PARM_BLK will contain a parameter element as follows:

IPSET_MSG_INFO
IPPARM_SIP_HDR
- value = NULL-terminated string which begins with the string “Reason-Phrase:”

Note: Depending on the list of header fields that the application has registered to receive, the GC_PARM_BLK associated with the GCEV_ALERTING event may contain multiple parameter elements that use the IPSET_SIP_MSG_INFO / IPPARM_SIP_HDR ID pair. It is the application’s responsibility to parse the value strings of these parameter elements to identify the one that begins with the “Reason-Phrase:” string.

Gatekeeper Registration Failure (H.323)

The following note is added to the Global Call IP for Host Media Processing Technology Guide at the end of the Gatekeeper Registration Failure (H.323) section:

Note: The RAS GCEV_TASKFAIL event automatically repeats at intervals of 30 seconds if the application does not re-register with a Gatekeeper. This is done to remind the application that it must deal with the registration failure before it can successfully make or receive any new calls.

The following changes are necessary for the Global Call IP for Host Media Processing Technology Guide (doc no. 05-2239-005):

On the reference page for the IP_H221NONSTANDARD data structure (page 391), the descriptions of the three data fields are updated as follows (Defect# IPY00029961 = PTR# 36777):

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

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

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

In Section 7.3.16, "gc_MakeCall( ) Variances for IP", the last paragraph before Section 7.3.16.1 (page 312) is updated as follows (Defect# IPY00029956 = PTR# 36646):

When using SIP, if the remote side does not send a final response to an outgoing INVITE (sent by the call control library) within 64 seconds, the gc_MakeCall( ) function times out and the library generates a GCEV_DISCONNECTED event to the application. If the application attempts to drop the call before the 64 second time-out is reached, the library's behavior depends on whether a provisional response was received. When no provisional response was received before the application cancels the call, the library cleans up the call immediately. But if a provisional response was received before the application attempts to cancel the call, the library sends a CANCEL to the remote endpoint and generates a GCEV_DROPCALL to the application after it receives the

200OK response to the CANCEL and a 487RequestTerminated response for the original INVITE, or when an additional 32-second time-out expires.

Update to **section 8.2, "Parameter Set Reference"**

New parameter IDs have been added to IPSET_CALLINFO so that you can send and receive CPN fields via Global Call over an IP network. See Section 1.2, "Global Call API Access to New H.323/Q.931 Message IEs", on page 14 for more details.

3.4.8 Device Management API For Windows and Linux Operating Systems Library Reference

There currently are no changes to this document.

3.4.9 Fax Software Reference for Windows

There currently are no changes to this document.

3.4.10 IP Media Library API for Host Media Processing Library Reference

The following information needs to be added to the IP Media Library API for Host Media Processing Library Reference:

RTCP Reporting Interval

The Dialogic® Host Media Processing for 1.3WIN Service Update implements an enumeration that allows IPML applications to set the real time control protocol (RTCP) calculation frequency within a given range (1-15 seconds). By default, IPML calculates RTCP statistics approximately every 5 seconds.

To support this feature, the following row should be added to Table 2. eIPM_PARM Parameters and Values on the IP_PARM_INFO data structure reference page:

eIP_PARM Define	Description and Values
PARMBD_RTCPAUDIO_INTERVAL	sets the RTCP calculation frequency for audio streams Type: integer Valid Values: 1 to 15 Default: 5

Notes:1. The PARMBD_RTCPAUDIO_INTERVAL is a board-level parameter; the setting is applied to all IP devices on the board.

2. If the value is changed, the new value is immediately applied to the board.

3. This feature applies to audio RTP and RTCP packets only

UDP Port Range

For previous releases, when the **ipm_Open( )** function is called to open the initial IP device (ipmB1C1), the device is assigned an RTP port number of 0xc000 (41952). This is referred to as the RTP Base port number. The RTCP port for the ipmB1C1 device is 0xc001. RTP/RTCP ports for subsequent IP devices are assigned in a linear fashion, after 0xc001. After the **ipm_StartMedia( )** function is called to start the media session, the RTP/RTCP port numbers assigned to the IP devices should not be changed.

The Dialogic® Host Media Processing Software 1.3 for Windows Service Update implements an enumeration that allows IPML applications to change the RTP Base port number after the board has been downloaded. The RTP Base port number should not be changed when any channel is actively streaming.

To support this feature, the following row should be added to Table 2. eIPM_PARM Parameters and Values on the IP_PARM_INFO data structure reference page:

eIP_PARM Define	Description and Values
PARMBD_RTPAUDIO_PORT_BASE	sets the RTP Base port number for audio streams Type: integer Valid Values: 0 to 65535 Default: 49152

Notes:1. The PARMBD_RTPAUDIO_PORT_BASE is a board-level parameter and effects all IP devices on the board.

2. If the RTP Base port number is changed while a board's channels are in an active call (streaming), the new value will not take effect on the active channels until the calls end. However, modifying the RTP Base port number on an IP device with active channels will cause unexpected behavior on the active calls.

3. This feature applies to audio RTP and RTCP packets only.

G.726 Coder

The Dialogic® Host Media Processing Software 1.3 for Windows Service Update adds support for the G.726 coder. RFC2833 can be used in-band and out-of-band with the G.726 coder.

To support this coder, the following information should be added to the IP_CODER_INFO data structure reference page:

Field Descriptions:

The following G.726 coder information should be added to the eCoderType data structure field:

- CODER_TYPE_G726_16K = 2 bits/sec
- CODER_TYPE_G726_24K = 3 bits/sec

- CODER_TYPE_G726_32K = 4 bits/sec
- CODER_TYPE_G726_40K = 5 bits/sec

The following value information should be added to the unCoderPayloadType data structure field:

- static payload: 2
- dynamic payload: 96-127

Note: These payloads apply in the context of G.726 only.

Table 1. Supported Coder Properties for Host Media Processing - the following row should be added:

eCoderType	Frame Size (ms)	Frames per Packet (fpp)	eVadEnable Value
PARMBD_RTPAUDIO_PORT_BASE CODER_TYPE_G726_16K CODER_TYPE_G726_24K CODER_TYPE_G726_32K CODER_TYPE_G726_40K	20	1, 2, or 3	Must be CODER_VAD_DISABLE

IPM_CODER_INFO

The following note is added to the description of the **unCoderPayloadType** field of the **IPM_CODER_INFO** data structure description (page 92)(PTR# 33921):

Note: Applications must set a value that is compatible with the coder type that is specified in the eCoderType field before calling **ipm_StartMedia( )** or **ipm_ModifyMedia( )**. If the application does not set this field, the default value of 0 specifies G.711.

IPM_QOS_THRESHOLD_DATA

The descriptions of several fields in the reference pages for the IPM_QOS_THRESHOLD_DATA data structure (pages 104-105) are updated as follows (PTR# 35692):

unTimeInterval

time interval (in ms) between successive parameter measurements. Value should be set to a multiple of 100; other values are rounded to the nearest hundred.
Note: Value must be greater than unFaultThreshold for the jitter QoS type.

unDebounceOn

time interval for detecting potential alarm fault condition. Must be set to a value that is a multiple of unTimeInterval; other values are rounded down to the next lower multiple of unTimeInterval.
Note: This field is not used for RTCP and RTP Time-out alarms and must be set to 0.

unDebounceOff
time interval for detecting potential alarm non-fault condition. Must be set to a value that is a multiple of unTimeInterval; other values are rounded down to the next lower multiple of unTimeInterval.
Note: This field is not used for RTCP and RTP Time-out alarms and must be set to 0.

unPercentSuccessThreshold
percentage of poll instances in unDebounceOff time interval that the fault threshold must not be exceeded before an "alarm off" event is sent. Allowed values correspond to multiples of the ratio of unDebounceOff to unTimeInterval (i.e., the inverse of the number of poll instances) expressed as an integer percentage; other values are truncated to the next lower percentage multiple.
Note: This parameter is not used for RTCP and RTP Time-out alarms and must be set to 0.

unPercentFailThreshold
percentage of poll instances in unDebounceOn time interval that the fault threshold must be exceeded before an "alarm on" event is sent. Allowed values correspond to multiples of the ratio of unDebounceOn to unTimeInterval (i.e., the inverse of the number of poll instances) expressed as a integer percentage; other values are truncated to the next lower percentage multiple.
Note: This parameter is not used for RTCP and RTP Time-out alarms and must be set to 0.

Additionally, the following paragraphs are added to the end of the description of IPM_QOS_THRESHOLD_DATA (page 105):

QoS debouncing is calculated as an integer number of parameter measurements that must exceed (or fall below) the fault threshold within the debounce interval before an alarm-on (or alarm-off) event is generated. The calculation uses the following formulas:

For QoS alarm-on debouncing:
count = int(int(unDebounceOn/unTimeInterval) * (unPercentFailThreshold/100))

For QoS alarm-off debouncing:
count = int(int(unDebounceOff/unTimeInterval) * (unPercentSuccessThreshold/100))

Clarification of ipm_GetLocalMediaInfo( ) Caution:
On the ipm_GetLocalMediaInfo( ) function reference page beginning on page 25, the Caution text (on page 26) should clarify unCount as a member of the data structure pointed to by pMediaInfo. The caution should read:

To retrieve RTP or T.38 information, set the eMediaType field to MEDIATYPE_LOCAL_RTP_INFO or MEDIATYPE_LOCAL_UDPTL_T38_INFO and set the unCount member of the data structure pointed to by pMediaInfo to 1. See the example for details.

3.4.11 IP Media Library API for Host Media Processing Programming Guide

The following information needs to be added to the IP Media Library API for Host Media Processing Programming Guide:

The fourth paragraph in Section 6.1, "Introduction to DTMF Handling", (page 23) and the note that follows the fourth paragraph should be ignored. The IPM_RFC2833MUTE_AUDIO parameter that the paragraph refers to is not supported; DTMF audio is always muted when in RFC2833 mode. Similarly, Step 5 in the procedure in Section 6.2.3, "Setting RFC 2833 Mode", (page 26) should also be ignored (PTR# 33826).

QOS Alarm Types

The descriptions of the supported QoS alarm types in Section 8.2 (pages 37-38) are updated as follows (PTR#35692):

All QoS alarms operate on a per-channel basis. That is, a QoS alarm indicates the status of a particular channel during a particular session, not the status of an entire IP media resource board.

The following QoS alarm types are supported in the IP media software. These names are used in the IPM_QOS_THRESHOLD_DATA structure when setting parameters for the alarms, and in the IPM_QOS_ALARM_DATA structure that is associated with the IPMEV_QOS_ALARM event that is generated when an alarm state transition occurs.

QOSTYPE_JITTER
: QoS alarm for excessive average jitter

QOSTYPE_LOSTPACKETS
: QoS alarm for excessive percentage of lost packets

QOSTYPE_RTCPTIMEOUT
: QoS alarm for RTCP time-out, indicating that RTCP packets are no longer being received. This alarm can also indicate that the network cable is disconnected.

QOSTYPE_RTPTIMEOUT
: QoS alarm for RTP time-out, indicating that RTP packets are no longer being received. This alarm can also indicate that the network cable is disconnected.

The descriptions of the QoS alarm threshold parameters in Section 8.2 (page 38) are updated as follows:

All QoS alarm types have one or more threshold attributes, such as time interval and fault threshold, which specify how the system determines when to generate a QoS alarm event.

The threshold attributes listed below are specified in IPM_QOS_THRESHOLD_DATA structures that are contained in an IPM_QOS_THRESHOLD_INFO structure that is passed to ipm_SetQoSThreshold( ):

unTimeInterval
: time interval between successive parameter measurements, in multiples of 100 (ms)

unDebounceOn
polling interval for detecting potential alarm fault condition. This interval must be a multiple of unTimeThreshold.

unDebounceOff
polling interval for measuring potential alarm non-fault condition. This interval must be a multiple of unTimeThreshold.

unFaultThreshold
fault threshold value. The meaning and value range of this attribute depend on the alarm type.

unPercentSuccessThreshold
percentage of poll instances in unDebounceOff interval that the fault threshold must not be exceeded before an “alarm off” event is sent. The granularity for this attribute is the ratio of unTimeInterval to unDebounceOff, expressed as a percentage.

unPercentFailThreshold
percentage of poll instances in unDebounceOn interval that the fault threshold must be exceeded before an “alarm on” event is set. The granularity for this attribute is the ratio of unTimeInterval to unDebounceOff, expressed as a percentage.

Note: Not all attributes are supported for all alarm types and products. All attributes that are not supported should be set to 0.

3.4.12 Standard Runtime Library API for Linux and Windows Operating Systems Library Reference

There currently are no changes to this document.

3.4.13 Standard Runtime Library API for Windows Operating Systems Programming Guide

There currently are no changes to this document.

3.4.14 Voice API for Host Media Processing Library Reference

The following function information needs to be added to the Voice API for Host Media Processing Library Reference:

Update to **dx_listenEx( )** (IPY00032425)

In the Cautions section, the following caution is added:

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

Update to **dx_unlistenEx( )** (IPY00032425)

In the Cautions section, the following caution is added:

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

Added **ATDX_CRTNID( )** function

This function should be added to support the special information tone (SIT) frequency detection.

ATDX_CRTNID( )

Name: long ATDX_CRTNID(chdev)

Inputs: int chdev • valid channel device handle

Returns: identifier of the tone that caused the most recent call progress analysis termination, if successful
AT_FAILURE if error

Includes: srllib.h
dxxxlib.h

Category: Extended Attribute

Mode: synchronous

■ Description

The **ATDX_CRTNID( )** function returns the last call progress analysis termination of the tone that caused the most recent call progress analysis termination of the channel device. See the *Voice API Programming Guide* for a description of call progress analysis.

Parameter	Description
chdev	specifies the valid channel device handle obtained when the channel was opened using **dx_open( )**

Possible return values are:

TID_BUSY1
First signal busy

TID_BUSY2
Second signal busy

TID_DIAL_INTL
International dial tone

TID_DIAL_LCL
Local dial tone

TID_DISCONNECT
Disconnect tone (post-connect)

TID_FAX1
First fax or modem tone

TID_FAX2
Second fax or modem tone

TID_RNGBK1
Ringback (detected as single tone)

TID_RNGBK2
Ringback (detected as dual tone)

TID_SIT_ANY
Catch all (returned for a Special Information Tone sequence or SIT sequence that falls outside the range of known default SIT sequences)

TID_SIT_INEFFECTIVE_OTHER or
TID_SIT_IO
Ineffective other SIT sequence

TID_SIT_NO_CIRCUIT or
TID_SIT_NC
No circuit found SIT sequence

TID_SIT_NO_CIRCUIT_INTERLATA or
TID_SIT_NC_INTERLATA
InterLATA no circuit found SIT sequence

TID_SIT_OPERATOR_INTERCEPT or
TID_SIT_IC
Operator intercept SIT sequence

TID_SIT_REORDER_TONE or
TID_SIT_RO
Reorder (system busy) SIT sequence

TID_SIT_REORDER_TONE_INTERLATA or
TID_SIT_RO_INTERLATA
InterLATA reorder (system busy) SIT sequence

TID_SIT_VACANT_CIRCUIT or
TID_SIT_VC
Vacant circuit SIT sequence

Cautions

None.

Errors

This function fails and returns AT_FAILURE if an invalid device handle is specified.

Example

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

main()
{
   DX_CAP  cap_s;
   int     ddd, car;
   char    *chnam, *dialstrg;
   long    tone_id;
   chnam    = "dxxxB1C1";
   dialstrg = "L1234";
   /*
```

```
 *  Open channel
 */
if ((ddd = dx_open( chnam, NULL )) == -1 ) {
   /* handle error */
}

/*
 *  Dial
 */
printf("Dialing %s\n", dialstrg );
car = dx_dial(ddd,dialstrg,(DX_CAP *)&cap_s,DX_CALLP|EV_SYNC);
if (car == -1) {
   /* handle error */
}

switch( car ) {
case CR_NODIALTONE:
   switch( ATDX_DTNFAIL(ddd) ) {
   case 'L':
      printf(" Unable to get Local dial tone\n");
      break;
   case 'I':
      printf(" Unable to get International dial tone\n");
      break;
   case 'X':
      printf(" Unable to get special eXtra dial tone\n");
      break;
   }
   break;

case CR_BUSY:
   printf(" %s engaged - %s detected\n", dialstrg,
         (ATDX_CRTNID(ddd) == TID_BUSY1 ? "Busy 1" : "Busy 2") );
   break;
case CR_CNCT:
   printf(" Successful connection to %s\n", dialstrg );
   break;
case CR_CEPT:
   printf(" Special tone received at %s\n", dialstrg );
   tone_id  = ATDX_CRTNID(ddd);  //ddd is handle that is returned by dx_open()

   switch (tone_id) {

   case TID_SIT_NC:
     printf("No circuit found special information tone received\n");
     break;
   case TID_SIT_IC:
     printf("Operator intercept special information tone received\n");
     break;
   case TID_SIT_VC:
     printf("Vacant circuit special information tone received\n");
     break;
   case TID_SIT_RO:
     printf("Reorder special information tone received\n");
     break;
   case TID_SIT_NC_INTERLATA:
     printf("InterLATA no circuit found special information tone received\n");
     break;
   case TID_SIT_RO_INTERLATA:
     printf("InterLATA reorder special information tone received\n");
     break;
   case TID_SIT_IO:
     printf("Ineffective other special information tone received\n");
     break;
   case TID_SIT_ANY:
     printf("Catch all special information tone received\n");
     break;
```

```
      }
      break;
default:
   break;
}

/*
 *  Set channel on hook
 */
if ((dx_sethook( ddd, DX_ONHOOK, EV_SYNC )) == -1) {
   /* handle error */
}

dx_close( ddd )
```

3.4.15 Voice API for Host Media Processing Programming Guide

The following information needs to be added to the Voice API for Host Media Processing Programming Guide:

In **Section 7.5.6, "SIT Frequency Detection"**, the information following the second paragraph is replaced by the following:

Table 4 provides the default tone definitions for SIT sequences used. The table describes existing SIT sequences that have broader definitions as well as new SIT sequences. Note the following:

- The values in the Freq. column represent minimum and maximum values in Hz.
- Time refers to minimum and maximum on time in 10 msec units; the maximum off time between each tone is 5 (or 50 msec).
- The repeat count is 1 for all SIT segments.
- N/A means not applicable.
- For TID_SIT_ANY, the frequency and time of the first and second segments are open; that is, they are ignored. Only the frequency of the third segment is relevant.
- The tone IDs have aliases:
  - TID_SIT_NO_CIRCUIT (TID_SIT_NC)
  - TID_SIT_OPERATOR_INTERCEPT (TID_SIT_IC)
  - TID_SIT_VACANT_CIRCUIT (TID_SIT_VC)
  - TID_SIT_REORDER_TONE (TID_SIT_RO)
  - TID_SIT_NO_CIRCUIT_INTERLATA (TID_SIT_NC_INTERLATA)
  - TID_SIT_REORDER_TONE_INTERLATA (TID_SIT_RO_INTERLATA)
  - TID_SIT_INEFFECTIVE_OTHER (TID_SIT_IO)

Table 4. Special Information Tone Definitions

SIT		1st Segment		2nd Segment		3rd Segment	
Tone ID	**Description**	**Freq.**	**Time**	**Freq.**	**Time**	**Freq.**	**Time**
TID_SIT_NC	No Circuit Found	950/1020	32/45	1400/1450	32/45	1740/1850	N/A
TID_SIT_IC	Operator Intercept	874/955	15/30	1310/1430	15/30	1740/1850	N/A

Table 4. Special Information Tone Definitions (Continued)

SIT		1st Segment		2nd Segment		3rd Segment	
Tone ID	Description	Freq.	Time	Freq.	Time	Freq.	Time
TID_SIT_VC	Vacant Circuit	950/1020	32/45	1310/1430	15/30	1740/1850	N/A
TID_SIT_RO	Reorder (system busy)	874/955	15/30	1400/1450	32/45	1740/1850	N/A
TID_SIT_NC_ INTERLATA	InterLATA No Circuit Found	874/955	32/45	1310/1430	32/45	1740/1850	N/A
TID_SIT_RO_ INTERLATA	InterLATA Reorder (system busy)	950/1020	15/30	1310/1430	32/45	1740/1850	N/A
TID_SIT_IO	Ineffective Other	874/955	32/45	1400/1450	15/30	1740/1850	N/A
TID_SIT_ANY	Catch all tone definition	Open	Open	Open	Open	1725/1825	N/A

3.4.15.1 ATDX_CRTNID( ) Support

The **ATDX_CRTNID( )** function is now supported and the following new tone IDs can be returned by this function:

Tone ID	Description
TID_SIT_IC TID_SIT_OPERATOR_INTERCEPT	Operator intercept SIT sequence
TID_SIT_IO TID_SIT_INEFFECTIVE_OTHER	Ineffective other SIT sequence
TID_SIT_NC TID_SIT_NO_CIRCUIT	No circuit found SIT sequence
TID_SIT_NC_INTERLATA TID_SIT_NO_CIRCUIT_INTERLATA	InterLATA no circuit found SIT sequence
TID_SIT_RO TID_SIT_REORDER_TONE	Reorder (system busy) SIT sequence
TID_SIT_RO_INTERLATA TID_SIT_REORDER_TONE_INTERLATA	InterLATA reorder (system busy) SIT sequence
TID_SIT_VC TID_SIT_VACANT_CIRCUIT	Vacant circuit SIT sequence
TID_SIT_ANY	Catch all (returned for a SIT sequence that falls outside the range of known default SIT sequences)

For additional information about the **ATDX_CRTNID( )** function, see Section 3.4.14, “Voice API for Host Media Processing Library Reference”, on page 98

3.5 Demonstration Software Documentation

This section contains updates to the following documents:

- Audio Conferencing API for Host Media Processing Demo Guide
- Continuous Speech Processing API for Host Media Processing on Windows Demo Guide
- Global Call API for Host Media Processing Demo Guide
- IP Media Server for Host Media Processing Demo Guide

3.5.1 Audio Conferencing API for Host Media Processing Demo Guide

In **Section 4.1, "Starting the Demo"**, the following changes apply:

- Step 3 of the procedure should read: "Make an IP call into the system using two IP endpoints".

3.5.2 Continuous Speech Processing API for Host Media Processing on Windows Demo Guide

In **Section 4.1, "Starting the Demo"**, the following changes apply:

- For Windows, the demo is located in the following directory:

```
$(INTEL_DIALOGIC_DIR)\demos\SpeechProcessing\CSPDemo\Release\
```

3.5.3 Global Call API for Host Media Processing Demo Guide

There currently are no changes to this document.

3.5.4 IP Media Server for Host Media Processing Demo Guide

There currently are no changes to this document.