Dialogic
Making Innovation Thrive™

Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Release Update

March 13, 2009

05-2587-007

Publication Date: March 13, 2009

Document Number: 05-2587-007

About This Publication

This section contains information about the following topics:

- Purpose
- Intended Audience
- How to Use This Publication
- Related Information

Purpose

This Release Update addresses issues associated with the Dialogic® Multimedia Software for AdvancedTCA Release 1.1. In addition to summarizing issues that were known as of this release, it is intended that the Release Update will continue to be updated to serve as the primary mechanism for communicating any new issues that may arise after the release date.

Intended Audience

This Release Update is intended for all users of the Dialogic® Multimedia Platform for AdvancedTCA and Dialogic® Multimedia Software for AdvancedTCA Release 1.1.

How to Use This Publication

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

- Document Revision History: This section summarizes the ongoing changes and additions that are made to this Release Update after its original release. This section is organized by document revision and document section.
- Post-Release Developments: This section describes significant changes to the release subsequent to the general availability release date. For example, new features provided in the Service Update are described in this section.
- Release Issues: This section lists issues that may affect the system release hardware and software.
- Documentation Updates: This section contains corrections and other changes that apply to the documentation not made prior to the release. These updates are organized by documentation category and by individual document.

Related Information

See the following for additional information:

- For information about the products and features supported in this release, see the *Dialogic® Multimedia Software for AdvancedTCA Release 1.1 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/manuals/* (for Dialogic® product documentation)
- *http://www.dialogic.com/support/* (for Dialogic technical support)
- *http://www.dialogic.com/* (for Dialogic® product information)

Document Revision History

This Revision History summarizes the changes made in this and each previously published version of the Release Update for Dialogic® Multimedia Software for AdvancedTCA Release 1.1, which is a document that is periodically updated throughout the lifetime of the release.

Document Rev 07 - published March 13, 2009

Updated for Service Update 55.

In the Post-Release Developments chapter:

- Added Using Multiple Ethernet Interfaces on RTM or Fabric to Media Gateway.

In the Release Issues chapter:

- Added the following Resolved Issues IPY00035816, IPY00042959, IPY00043816, IPY00044620, IPY00044685, IPY00044750, IPY00044824, IPY00044834, IPY00044978, IPY00045006, IPY00045089, IPY00045260, IPY00045484, IPY00078344, IPY00078369, IPY00078606, IPY00078827, IPY00079123, IPY00079169, IPY00079177, IPY00079254.
- Added the following Known (Permanent) Issue: IPY00038755.
- Added the following Known Issue: IPY00044651.

In the Documentation Updates chapter:

- Added update to Dialogic® Standard Runtime Library API Library Reference for sr_getfdcnt( ) and sr_getfdinfo( ) functions (IPY00045054).
- Added that a new document named Dialogic® Global Call API Demo Guide is now available on the documentation bookshelf.
- Added that a new document named Dialogic® Multimedia Demo Guide is now available on the documentation bookshelf.

Document Rev 06 - published April 29, 2008

Updated for Service Update 46.

In the Post-Release Developments chapter:

- Added Retrieving Vendor Identification Information for 3G Handsets Using 3G-324M API.
- Added Enabling Runtime Tracing in 3G-324M API.

In the Release Issues chapter:

- Added the following Resolved Issues: IPY00042796, IPY00042557, IPY00041623, IPY00040888, IPY00038779.

In the Documentation Updates chapter:

- Added that new version of document (05-2558-004) is available for 3G-324M API Library Reference.

Document Rev 05 - published January 29, 2008

Updated for Service Update 39.

In the Post-Release Developments chapter:

- Added Configuring SIP Stack Parameters with Global Call.

In the Documentation Updates chapter:

- Added documentation update for the Dialogic® Global Call IP Technology Guide to support new feature in the Service Update.

Document Rev 04 - published November 7, 2007

Updated for Service Update 37.

In the Release Issues chapter:

- Added the following Resolved Issues: IPY00040876, IPY00040782, IPY00040542, IPY00040434, IPY00039475, IPY00038848, IPY00038802, IPY00038651, IPY00038058, IPY00036832.

In the Documentation Updates chapter:

- Added that new version of document (05-2257-010) is available in Dialogic® IP Media Library API Library Reference.
- Removed documentation updates as a new version of document (05-2333-004) is available in Dialogic® Voice API Library Reference.
- Removed documentation updates as a new version of document (05-2332-004) is available in Dialogic® Voice API Programming Guide.
- Removed documentation updates as a new version of document (05-2513-001) is available in Dialogic® MSML Media Server Software User’s Guide.

Document Rev 03 - published July 2007

Updated for Service Update 30.

In the Release Issues chapter:

- Added a Known (Permanent) Issue for Multimedia (no defect number).

- Added the following Known Issue: IPY00038605.

In the Documentation Updates chapter:

- Added that new version of document (05-2581-003) is available in Dialogic® Multimedia Software for AdvancedTCA Release 1.1 Release Guide.
- Removed note about Nb UP functionality not being available in Dialogic® IP Media Library API Library Reference. Nb UP functionality is now supported.

Document Rev 02 - published June 2007

Updated for Service Update 29.

In the Release Issues chapter:

- Added SU No. column to the Issues table.
- Added the following Resolved Issues: IPY00037485.
- Removed the following Known (Permanent) Issue: IPY00032616 (not a defect).
- Added the following Known Issues: IPY00038392, IPY00038149, IPY00038148, IPY00038147, IPY00038104, IPY00038090, IPY00038079, IPY00038076, IPY00038058, IPY00037784, IPY00037667.

In the Documentation Updates chapter:

- Added information to the Dialogic® Multimedia Software for AdvancedTCA Release 1.1 Release Guide that a new version is available on the bookshelf.
- Removed documentation updates for the Dialogic® Device Management API Library Reference and Dialogic® IP Media Library API Library Reference since these updates have been incorporated into the latest versions which are now available on the bookshelf.
- Added update to the Dialogic® Multimedia API Library Reference for MM_AUDIO_CODEC data structure.
- Added information to the Dialogic® Multimedia File Conversion Tools User Guide that a new version is available and included link to the document.

Document Rev 01 - published April 2007

Initial version of document.

Post-Release Developments 1

This section describes significant changes to Dialogic® Multimedia Software for AdvancedTCA Release 1.1 subsequent to the general availability release.

1.1 Service Update

A Service Update for Dialogic® Multimedia Software for AdvancedTCA Release 1.1 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. It is intended that this Release Update will document the features in the Service Updates.

1.2 Using Multiple Ethernet Interfaces on RTM or Fabric to Media Gateway

With Service Update 55, using multiple Ethernet interfaces on RTM or fabric to media gateway is supported.

If both connections on RTM or Fabric interfaces can reach all of the same endpoints, a “fall-over” scenario in which, when one link on an RTM or fabric goes down, the traffic will be re-routed to the other RTM or fabric interface. Packets in flight during the “fall-over” may be lost until all of the layer 2 devices from media gateway to endpoint adjust their MAC table entries to reflect the new path through the network. A conservative estimate for the time to adjust MAC table entries is between 300 and 4000ms dependent on the behavior of the network. The length of time is related to the location of the connected media gateway due to propagation delay. There is no explicit support for fail-safe operation in the media gateway, just the natural resilience of Ethernet/IP network connectivity.

Traffic can be directed through separate RTM or fabric interfaces if each interface has a different set of end points. For example, RTM1 is connected to 192.168.10.xxx and RTM2 is connected to 192.168.20.xxx.

There is no special routing or configuration needed to enable the above scenarios. However, both RTM ports will need to be enabled through the CLI interface.

1.3 Retrieving Vendor Identification Information for 3G Handsets Using 3G-324M API

With Service Update 46, retrieving vendor and product identification information of the connected 3G handset is supported in the 3G-324M API library. The **m3g_SetVendorID( )** function, M3G_VENDORID_INFO structure, M3G_REMOTE_VENDORID_EVT_TYP bitmask, and M3GEV_REMOTE_VENDORID_RCVD event are added to the library to support this feature. For details, see the *3G-324M API Library Reference.*

1.4 Enabling Runtime Tracing in 3G-324M API

With Service Update 46, runtime tracing on a per channel or board basis is supported in the 3G-324M API library. This feature allows you to set tracing of the following:

- H.245 messaging (with textual decode)
- raw binary H.223 multiplexed bitstreams
- raw binary audio streams
- raw binary video streams
- call statistics

Logging is directed to a user-specified file.

The **m3g_StartTrace( )** and **m3g_StopTrace( )** functions and M3G_TRACE_INFO structure are added to the library to support this feature. For details, see the *3G-324M API Library Reference*.

A post-processing 3G logfile parser is also provided. The parser utility is called *m3g_parser* and is used as follows from the command line:

```
m3g_parser <inputfile>
```

After executing the parser, a separate logfile (where *n* is the device number) is created for every channel for each of the following categories that may have been enabled in the specified logfile:

h245_n.txt	text file of recorded transmitted and received H.245 messaging
h223tx_n.bin	binary file of recorded raw H.223 transmit bitstream
h223rx_n.bin	binary file of recorded raw H.223 receive bitstream
audiotx_n.bin	binary file of recorded raw audio transmit bitstream
audiorx_n.bin	binary file of recorded raw audio receive bitstream
videotx_n.bin	binary file of recorded raw video transmit bitstream
videorx_n.bin	binary file of recorded raw video receive bitstream
stats_n.txt	text file of statistics

1.5 Configuring SIP Stack Parameters with Global Call

With Service Update 39, selected SIP stack parameters such as timers can now be configured with the Dialogic® Global Call API.

1.5.1 Feature Description

A new data structure, SIP_STACK_CFG, is used to configure SIP stack parameters. Details about the SIP_STACK_CFG data structure fields are given in Section 1.5.2, “SIP_STACK_CFG Data Structure”, on page 10.

To support SIP stack configuration, IP_VIRTBOARD has been updated with a new structure pointer (default is NULL) as follows:

```
typedef struct {
    ...
    ...
    /* The following is added for VIRTBOARD_VERSION_SIP_STACK_CFG support */
        SIP_STACK_CFG    *sip_stack_cfg;
    /* end VIRTBOARD_VERSION_SIP_STACK_CFG additions */
} IP_VIRTBOARD;
```

1.5.2 SIP_STACK_CFG Data Structure

The SIP_STACK_CFG structure definition has been added in the *gcip.h* file. The new data structure is described below.

Note: SIP stack parameters can only be configured once per virtual board (at **gc_Start( )**) and remain in effect throughout the Global Call application (per process).

SIP_STACK_CFG

```
typedef struct {
    unsigned long version;  /* version set by INIT_SIP_STACK_CFG */
    int retransmissionT1;
    int retransmissionT2;
    int retransmissionT4;
    int generalLingerTimer;
    int inviteLingerTimer;
    int provisionalTimer;
    int cancelGeneralNoResponseTimer;
    int cancelInviteNoResponseTimer;
    int generalRequestTimeoutTimer;
} SIP_STACK_CFG;
```

■ Description

The SIP_STACK_CFG data structure is used to configure selected SIP stack parameters such as timers.

The SIP_STACK_CFG data structure is referenced by the IP_VIRTBOARD data structure, which stores configuration and capability information about an IPT (virtual) board device that is populated when the device is started. An array of IP_VIRTBOARD structures (one per virtual board in the system) is referenced by the IPCCLIB_START_DATA structure, which is passed to the **gc_Start( )** function.

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

■ Field Descriptions

The fields of the SIP_STACK_CFG data structure are:

version
: The version number of the data structure. The correct value is set by the **INIT_SIP_STACK_CFG( )** initialization function and should not be overridden.

retransmissionT1
: Determines several timers as defined in RFC 3261. For example, when an unreliable transport protocol is used, a Client Invite transaction retransmits requests at an interval that starts at T1 milliseconds and doubles after every retransmission. A Client General transaction retransmits requests at an interval that starts at T1 and doubles until it reaches T2. The default value is 1000.

retransmissionT2
: Determines the maximum retransmission interval as defined in RFC 3261. For example, when an unreliable transport protocol is used, general requests are retransmitted at an interval that starts at T1 and doubles until it reaches T2. If a provisional response is received, retransmissions continue but at an interval of T2. The parameter value cannot be less than 4000. The default value is 8000.

retransmissionT4

Determines the amount of time the network takes to clear messages between client and server transactions as defined in RFC 3261. For example, when working with an unreliable transport protocol, T4 determines the time that a UAS waits after receiving an ACK message and before terminating the transaction. The default value is 10000.

generalLingerTimer

After a server sends a final response, the server cannot be sure that the client has received the response message. The server should be able to retransmit the response upon receiving retransmissions of the request for generalLingerTimer milliseconds. The default value is 32000.

inviteLingerTimer

After sending an ACK for an INVITE final response, a client cannot be sure that the server has received the ACK message. The client should be able to retransmit the ACK upon receiving retransmissions of the final response for inviteLingerTimer milliseconds. The default value is 32000.

provisionalTimer

The provisionalTimer is set when receiving a provisional response on an Invite transaction. The transaction will stop retransmissions of the Invite request and will wait for a final response until the provisionalTimer expires. If you set the provisionalTimer to 0, no timer is set, and the Invite transaction will wait indefinitely for the final response. The default value is 180000.

cancelGeneralNoResponseTimer

When sending a CANCEL request on a General transaction, the User Agent waits cancelGeneralNoResponseTimer milliseconds before timeout termination if there is no response for the canceled transaction. The default value is 32000.

cancelInviteNoResponseTimer

When sending a CANCEL request on an Invite request, the User Agent waits cancelInviteNoResponseTimer milliseconds before timeout termination if there is no response for the canceled transaction. The default value is 32000.

generalRequestTimeoutTimer

After sending a General request, the User Agent waits for a final response generalRequestTimeoutTimer milliseconds before timeout termination (in this time the User Agent retransmits the request every T1, 2*T1, ... , T2, ... milliseconds). The default value is 32000.

1.5.3 Sample Code

The following example sets the SIP T1 timer to 64 ms.

```
#include "gclib.h"
..
..
#define BOARDS_NUM 1
..
..

/* initialize start parameters */
IPCCLIB_START_DATA cclibStartData;
memset(&cclibStartData,0,sizeof(IPCCLIB_START_DATA));
IP_VIRTBOARD virtBoards[BOARDS_NUM];
memset(virtBoards,0,sizeof(IP_VIRTBOARD)*BOARDS_NUM);

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

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

/* sip stack cfg support */
SIP_STACK_CFG sip_stack_cfg;
INIT_SIP_STACK_CFG(&sip_stack_cfg);
   virtBoard[bid].sip_stack_cfg = &sip_stack_cfg;

   sip_stack_cfg.retransmissionT1 = 64;
```

1.5.4 Documentation

The online bookshelf provided with Dialogic® Multimedia Software for AdvancedTCA Release 1.1 contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Dialogic® Global Call API in general, see the following documents:

- *Dialogic® Global Call API Programming Guide*
- *Dialogic® Global Call API Library Reference*

For features specific to IP technology, see:

- *Dialogic® Global Call IP Technology Guide*

Release Issues 2

The table below lists issues that can affect the hardware and software supported in the Dialogic® Multimedia Software for AdvancedTCA Release 1.1. The following information is provided for each issue:

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

- Known – A minor hardware or software issue. This category includes interoperability issues and compatibility issues. Known issues are still open but may or may not be fixed in the future.
- Known (permanent) – A known hardware or software issue or limitation that is not intended to be fixed in the future.
- Resolved – A hardware or software issue that was resolved (usually either fixed or documented) in this release.

Defect No.
: A unique identification number that is used to track each issue reported via a formal Change Control System. Additional information on defects may be available via the Defect Query tool at *http://membersresource.dialogic.com/defects/*.
Note that when you select this link, you will be asked to either LOGIN or JOIN.

SU No.
: For defects that were resolved in a Service Update, the Service Update number is shown. For defects that were resolved when the base release was generally available (before any Service Updates), a “--” is shown. For non-resolved issues, this information is left blank.

Product or Component
: The product or component to which the problem relates, for example, an API.

Description
: A summary description of the issue. For non-resolved issues, a workaround is included when available.

Issues Sorted by Type, Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00078369	55	3G-324M	An audio problem occurs with a 3G call to a soft phone.
Resolved	IPY00078344	55	3G-324M	Interoperability problems occur with the adaptation layer.

Issues Sorted by Type, Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00045484	55	3G-324M	Only AL3 unidirectional is supported by H324M layer. A change was made in this Service Update to communicate the lack of bi-directional video OLC support to the remote 3G peer endpoint. Now, the application automatically responds with an H.245 OLCReject upon receiving a bi-directional video OLC request,
Resolved	IPY00045089	55	3G-324M	An incompatibility with a 3G-324m service endpoint causes missing DTMFs or no video.
Resolved	IPY00044834	55	3G-324M	The **m3g_StartTrace( )** function appears to cause a firmware crash.
Resolved	IPY00044824	55	3G-324M	The **m3g_OpenLC( )** function returns an M3GEV_OPEN_LC_FAIL event with the error M3G_E_ERR_PROTOCOL when using Motorola handsets with WNSRP enabled.
Resolved	IPY00043816	55	3G-324M	**m3g_StopH245( )** fails when the application isn't able to close the LC before calling the function.
Resolved	IPY00078827	55	CSP	During Dialogic® HMP load testing, the 67th CSP channel fails ec_stream callback.
Resolved	IPY00079254	55	Fax	A sporadic failure occurs with the Dialogic® HMP software.
Resolved	IPY00079177	55	Fax	The fax port does not receive T.38 messages from the remote end.
Resolved	IPY00079123	55	Fax	The application receives a TFX_FAXRECV message, but the TIF file contains a "black" image.
Resolved	IPY00078606	55	Fax	A random issue occurs when using the **fx_stopch( )** function where all the fax channels cannot be used.
Resolved	IPY00045006	55	Fax	An exception in the fax component causes resources to remain stuck in a non-idle state.
Resolved	IPY00044978	55	Fax	The Fax resource remains stuck in the rcvfax state. fax
Resolved	IPY00044750	55	Fax	No TFX_FAXERROR event is received after calling the **fx_stopch( )** function.
Resolved	IPY00044620	55	Fax	The Fax resource remains stuck in the STOP state.
Resolved	IPY00045156	55	Firmware	Incorrect timestamps are displayed in the m3g firmware traces.
Resolved	IPY00035816	55	Firmware	The IPMI and DS1 adaptors display error messages indicating that RTM FRU could not be read even though RTM is not plugged in.
Resolved	IPY00045260	55	IP Media	RFC2833 events are not separated from **ipm_SendDigit( )** DTMFs.
Resolved	IPY00042959	55	IP Media	Delays occur when messages are processed simultaneously across multiple channels.

Issues Sorted by Type, Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Resolved	IPY00044685	55	Multimedia	When hmp3gp (version 2.4.0.44) is generating files created by old multimedia firmware (version 7 video), it results in files with blank video.
Resolved	IPY00042557	46	3G-324M	In a 3G video server application where the digit is meant to stop multimedia play, it takes almost 2 seconds after pressing the digit on 3G handset for the phone to stop receiving multimedia.
Resolved	IPY00041623	46	3G-324M	3G-324M applications built before SU 39 must be recompiled. As of SU 46, INIT inline functions are provided to initialize various data structures. See the 3G-324M API Library Reference for details.
Resolved	IPY00038779	46	Fax	The following fax failure error message is seen with ssp_x86Linux_boot: `[18]PAGE_TxECM: FC3_ReadFrame call FAILED`
Resolved	IPY00040888	46	Firmware	When ssp_x86Linux_boot terminates, there is excessive CPU usage in process from the bmserver.
Resolved	IPY00042796	46	Install	Ethernet interface is incorrectly disabled when configured after software installation.
Resolved	IPY00037485	29	3G-324M	The audio and video synchronization problem seen with recorded 3G calls on the Multimedia Platform.
Known	IPY00038090		Conferencing	There are clicks heard by the user when another party is removed from the conference.
Known (permanent)	IPY00037292		Conferencing	The conferencing party attribute “DTMF clamping” is overwritten by conference level attribute when the party is removed and added back into the same conference.
Known (permanent)	IPY00038755		Firmware	Playback of recorded RFC2833 digits could result in slightly poorer then acceptable performance of 2 in 1000 16 digit strings being improperly detected. Failures typically show multiple digits detected for a single digit sent.
Known	IPY00038392		Firmware	The volume control on IPM devices does not work. The parameter is 0x4002 (PARMCH_RX_ADJVOLUME). Modifying this parameter alone should modify the Rx volume, but it still does not work. ***Workaround:*** Set the parameters, 0x4002 and 0x4f0e, to the same value. It has been verified through the *HMP.Uconfig* file for the entire board. The user has to create the *HMP.Uconfig* file in the **/usr/dialogic/data** directory. Also, it should work when set in code with **ipm_SetParm( )**.
Known	IPY00038076		Firmware	When connecting an MM device to an IPM device, audio quality for mm_play or mm_record is poor for some of the channels when simultaneous channel density is greater than 200.

Issues Sorted by Type, Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Issue Type	Defect No.	SU No.	Product or Component	Description

Known	IPY00037919		Firmware	If the following error is seen in **/var/log/messages**:
				`Apr 7 05:34:13 py1atcatvl9 DIAG: CRITICAL /usr/dialogic/bin/tvl2_diag: could not init AMC/RTM device`
				`Apr 7 05:34:13 py1atcatvl9 Board Manager: From Component: Media Adaptor : Firmware failed`
				The message can be ignored if it is followed by the diagnostics tests passing:
				`Apr 7 05:34:13 py1atcatvl9 DIAG: INFO Ekey register check is disabled`
				`Apr 7 05:34:13 py1atcatvl9 DIAG: INFO Ekey interrupt check is disabled`
				`Apr 7 05:34:13 py1atcatvl9 DIAG: INFO 1 FPGA Registers SUCCESS AMC FPGA`
				`Apr 7 05:34:13 py1atcatvl9 DIAG: INFO 2 FPGA Timer/Inter DISABLED AMC FPGA`
				`Apr 7 05:34:13 py1atcatvl9 DIAG: INFO 3 Switch Registers DISABLED AMC SWITCH`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 4 PHY Registers SUCCESS AMC PHY`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 5 DSP Control DISABLED AMC DSP`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 6 DSP Ping Data Pa DISABLED AMC DSP`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO Ekey register check is disabled`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO Ekey interrupt check is disabled`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 7 FPGA Registers SUCCESS RTM FPGA`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 8 Framer Registers SUCCESS RTM FRAMER`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 9 PHY Registers DISABLED RTM PHY`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 10 DPLL Registers SUCCESS RTM DPLL`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 11 Interrupt DISABLED RTM FPGA`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 12 DSP Packet DISABLED AMC DSP`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 13 DSP TDM DISABLED RTM DSP`
				`Apr 7 05:34:14 py1atcatvl9 DIAG: INFO 14 DSP heartbeat SUCCESS AMC DSP`
				The error message indicates that the diagnostic utility was temporarily unable to poll the firmware as a subsequent execution of the diagnostic utility shows that all the tests were successful.

Issues Sorted by Type, Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Issue Type	Defect No.	SU No.	Product or Component	Description

Known	IPY00037784		Firmware	The following error message is seen in **/var/log/messages** when the RTM is not present in the system. The message can be ignored.
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot: tvl2RtmDrvInit : *** Failed to register the device driver`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot: QERROR_KILLTASK(0) qkernerr.h 78029 qkernerr.h 30005`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot:`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot: Error Level : KILLTASK`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot: Location Tag: 78029, Location File: qkernerr.h`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot: Cause-Tag : 30005, Error File :qkernerr.h`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot:`
				`Mar 26 17:08:31 py1atcatvl29 ssp_x86Linux_boot: qFormatCrashString is not inited yet`

Known	IPY00037667		Firmware	There is loud noise caused by excessive handset gain, sometimes heard during 2G to 3G call bridging. ***Workaround:*** Reduce gain using parameter 0x4002 (PARMCH_RX_ADJVOLUME). Set the parameters, 0x4002 and 0x4f0e, to the same value. It has been verified through the *HMP.Uconfig* file for the entire board. The user has to create the *HMP.Uconfig* file in the **/usr/dialogic/data** directory. Also, it should work when set in code with **ipm_SetParm( )**.
Known	IPY00037586		Firmware	The firmware will segment fault during multimedia record operations if the disk is full. ***Workaround:*** Ensure there is sufficient space on the disk (local or network) prior to running the application.
Known	IPY00038149		Hardware	Shut down is not supported by opening of the Multimedia Blade latch. The Blue LED Hot Swap LED on the blade will blink indefinitely but never turn solid indicating shutdown complete. ***Workaround:*** Blade can be shutdown by logging on locally through the serial port or connect over the network and initiate a shutdown. Blade can be initialized by removing and reseating in the chassis.
Known	IPY00038148		Hardware	Activation and de-activation via the shelf manager is not supported on the Multimedia Blade latch. The Blue LED Hot Swap LED on the blade will blink indefinitely but never turn solid indicating the shutdown is complete. ***Workaround:*** Blade can be shutdown by logging on locally through the serial port or connect over the network and initiate a shutdown. Blade can be initialized by removing and reseating in the chassis.

Issues Sorted by Type, Dialogic® Multimedia Software for AdvancedTCA Release 1.1

Issue Type	Defect No.	SU No.	Product or Component	Description
Known	IPY00038147		Hardware	If the Multimedia Blade is deactivated using the shelf manager, the OS kernel will panic and not complete shut down. ***Workaround:*** Blade can be shutdown by logging on locally through the serial port or connect over the network and initiate a shutdown. Blade can be initialized by removing and reseating in the chassis.
Known	IPY00037689		Hardware	If the Multimedia Blade is shutdown with a RTM present, the baseboard Hot Swap (Blue) LED remains in a blinking state and never goes to a steady on LED state as expected to indicate shutdown is complete. ***Workaround:*** Wait for all software services to shut down before extracting the board.
Known	IPY00038605		IP Media	When called to retrieve RTP info for Nb UP, **ipm_GetLocalMediaInfo( )** returns an extra IPMEV_GET_LOCAL_MEDIA_INFO event for each call.
Known	IPY00038104		Multimedia	Under certain circumstances, completion event for mm_play (MMEV_PLAY) is not received. Once this happens, subsequent calls to mm_play also do not result in the completion event being received.
Known	IPY00038079		Multimedia	When recording to memory using **dx_reciottdata( )**, there is distortion at the end of file.
Known (permanent)			Multimedia	For best video quality, it is recommended that you use an external file server for multimedia play/record.
Known	IPY00044651		OA&M	The Dialogic® software uses INTEL_DIALOGIC variables, such as INTEL_DIALOGIC_INC and INTEL_DIALOGIC_LIB. These variables provide a standardized way of referencing directories that contain header files and shared objects. If these variables are deleted or not used, your system may not function properly.
Known	IPY00037286		OS	The OS race condition between closing and opening of socket will cause **ipm_StartMedia( )** to fail (IPMEV_ERROR) if it is called immediately after ipm_Stop.
Known	IPY00037590		Video Record	During video record, dropped packets may be experienced in less than 1.5% of recorded video files. The video files will show blockiness if video packets are dropped until the next IFrame is encountered.

Documentation Updates 3

The documentation updates for Dialogic® Multimedia Software for AdvancedTCA Release 1.1 are divided into the following sections, which correspond to the top level categories used on the online documentation navigation page:

3.1 System Release Documentation

This section contains updates to the following documents:

- Dialogic® Multimedia Software for AdvancedTCA Release 1.1 Release Guide

3.1.1 Dialogic® Multimedia Software for AdvancedTCA Release 1.1 Release Guide

There are currently no updates to this document.

3.2 Installation and Configuration Documentation

This section contains updates to the following documents:

- Dialogic® Multimedia Blade for AdvancedTCA Technical Product Specification

3.2.1 Dialogic® Multimedia Blade for AdvancedTCA Technical Product Specification

Update to Getting Started chapter.

The following new section should be added to the chapter:

Preserving Data in User Configuration Files

Configuration settings unique to your environment can be preserved and re-applied whenever an 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 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 INTEL_DIALOGIC_DIR, the environment variable for the directory in which the HMP software is installed. The contents of the *Hmp.Uconfig* file are not overwritten but are merged into the generated configuration file whenever a new Service Update is installed or a new HMP license is activated. You should also make a copy of the *Hmp.Uconfig* file and save it in a safe location as a backup.

Example

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

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

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

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

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

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

Setting Transparent Mode

To configure transparent mode on a system wide basis, add the following section to the Hmp.Uconfig file. Set the parameter to 1 to enable transparent mode on all PSTN channels and to 0 to disable transparent mode on all PSTN timeslots. Default is disabled.

```
[0x5a]
SetParm=0x1221,   1  ! 0 = normal, 1 = 3G clear channel mode
```

3.3 Programming Libraries Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- 3G-324M API Library Reference
- Dialogic® Conferencing API Library Reference
- Dialogic® Conferencing API Programming Guide
- Dialogic® Device Management API Library Reference
- Dialogic® Fax Software Reference
- Dialogic® Global Call API Library Reference
- Dialogic® Global Call API Programming Guide
- Dialogic® Global Call IP Technology Guide
- Dialogic® IP Media Library API Library Reference
- Dialogic® IP Media Library API Programming Guide
- Dialogic® Multimedia API Library Reference
- Dialogic® Multimedia API Programming Guide
- Dialogic® Multimedia File Conversion Tools User Guide
- Dialogic® Standard Runtime Library API Library Reference
- Dialogic® Standard Runtime Library API Programming Guide
- Dialogic® Voice API Library Reference
- Dialogic® Voice API Programming Guide

3.3.1 3G-324M API Library Reference

There are currently no updates to this document.

3.3.2 Dialogic® Conferencing API Library Reference

There are currently no updates to this document.

3.3.3 Dialogic® Conferencing API Programming Guide

There are currently no updates to this document.

3.3.4 Dialogic® Device Management API Library Reference

There are currently no updates to this document.

3.3.5 Dialogic® Fax Software Reference

There are currently no updates to this document.

3.3.6 Dialogic® Global Call API Library Reference

Update to the **gc_SetConfigData( )** function

The **gc_SetConfigData( )** function, which supports the Global Call Real Time Configuration Management (RTCM) feature, now allows the configuration of transparent mode on a PSTN channel basis at runtime. When transparent mode is **enabled**, there is no processing on the front end data coming in from the PSTN network before passing it to the TDM bus, nor is there any processing of data being transmitted to the PSTN network from the TDM bus. The default setting is transparent mode **disabled**.

> ***Note:*** This feature is supported only on Multimedia Platform for ATCA applications and not with any other products that use Global Call.

The **gc_SetConfigData( )** function uses a GC_PARM_BLK structure that contains the configuration element. The GC_PARM_BLK is populated using the **gc_util_insert_parm_val( )** function. To enable/disable transparent mode, the following setID/parmID pair is used:

CCSET_DM3FW_PARM is the setID

CCPARM_TRANSPARENTMODE is the parmID, with values:

- CCDM3FW_TRANSPARENTMODE_DISABLE
- CCDM3FW_TRANSPARENTMODE_ENABLE

The size of the parameter is a char or UINT8.

Once the GC_PARM_BLK has been populated with the desired values, the **gc_SetConfigData( )** function can be issued to perform the configuration. Use the target type GCTGT_CCLIB_CHAN.

To configure transparent mode on a system wide basis, you can add a parameter to the *Hmp.Uconfig* user configuration file. For more information, see Section 3.2.1, "Dialogic® Multimedia Blade for AdvancedTCA Technical Product Specification", on page 20.

Update to the **gc_GetAlarmConfiguration( )** function (GCAMS)

Not supported for the Multimedia Platform for ATCA.

Update to the **gc_SetAlarmConfiguration( )** function (GCAMS)

Not supported for the Multimedia Platform for ATCA.

3.3.7 Dialogic® Global Call API Programming Guide

Update to the **gc_GetAlarmConfiguration( )** function (GCAMS)

Not supported for the Multimedia Platform for ATCA.

Update to the **gc_SetAlarmConfiguration( )** function (GCAMS)

Not supported for the Multimedia Platform for ATCA.

3.3.8 Dialogic® Global Call IP Technology Guide

Update to the **gc_GetAlarmConfiguration( )** function (GCAMS)
Not supported for the Multimedia Platform for ATCA.

Update to the **gc_SetAlarmConfiguration( )** function (GCAMS)
Not supported for the Multimedia Platform for ATCA.

Update to **Section 4.3.2, Setting Coder Information**
In Table 2, "Coders Supported for Host Media Processing (HMP)," the following note should be added:

- Global Call supports AMR-NB and EVRC coders in third-party call control (3PCC) mode only. These coder types cannot be explicitly set using the Global Call API, but can be specified using the IP Media Library (IPML) API.

See the documentation updates in Section 3.3.10, "Dialogic® IP Media Library API Programming Guide", on page 24 of this Release Update for more information on using these coders in 3PCC mode.

Update to **Chapter 10, IP-Specific Data Structures**
Because of a feature introduced in the Service Update, a new data structure, SIP_STACK_CFG, has been added for configuring SIP stack parameters. Related to this, the IP_VIRTBOARD data structure has been updated to point to the new structure. For further information, see Section 1.5, "Configuring SIP Stack Parameters with Global Call", on page 10 of this Release Update.

3.3.9 Dialogic® IP Media Library API Library Reference

There are currently no updates to this document.

3.3.10 Dialogic® IP Media Library API Programming Guide

Using AMR coders for narrow band audio and using enhanced variable rate codecs
The following information, which is applicable to the ATCA Multimedia Platform, should be added to the programming guide, possibly as a new chapter.

Using AMR Coders for Narrow Band Audio

Description

AMR is an adaptive multi-rate speech codec. During operation, both local and remote sides can request a change in the bit rate and dynamically adjust the bandwidth. The protocol uses the following:

- A Frame Type (FT) to indicate the transmitted bit rate
- A Codec Mode Request (CMR) value to request a particular bit rate in every packet

To control the bit rate, AMR assumes that all connections are bi-directional.

This feature is specific to AMR-NB and excludes support for AMR-WB and AMR-WB+, which support wideband audio and some other formats not addressed by AMR.

The codec supports the following bit rates:

- 12.2 kbit/s (GSM EFR)
- 10.2 kbit/s
- 7.95 kbit/s
- 7.40 kbit/s (IS-641)
- 6.70 kbit/s (PDC-EFR)
- 5.90 kbit/s
- 5.15 kbit/s
- 4.75 kbit/s
- 1.80 kbit/s (assuming SID frames are continuously transmitted)

Note: The 1.80 kbit/s rate is not actually a voice signal, but the bit rate consumed when Voice Activation Detection (VAD) is processing a silence.

This feature is only supported when using the Session Initiated Protocol (SIP). None of the available SDP options are currently supported through Global Call (that is, direct first-party call control). The options are only available using third-party call control (3PCC), where the application is responsible for interpreting received SDP text strings and for constructing all outbound SDP text strings.

IP Media Library API Support

The bit rate can be controlled using a new media type in the IPM_MEDIA structure. The IP Media Library API allows the application to provide a preferred bit rate and a rule to determine how changes in the received CMR value control the transmitted bit rate.

Preferred bit rate

The preferred bit rate is specified by setting the eMediaType in the IPM_MEDIA structure to MEDIATYPE_AUDIO_REMOTE_CODER_INFO and setting the CoderInfo.eCoderType to the desired transmit bit rate.

CMR value

The transmitted CMR value is specified by setting the eMediaType in the IPM_MEDIA structure to MEDIATYPE_AUDIO_LOCAL_CODER_INFO and setting the CoderInfo.eCoderType to the desired CMR value.

The CMR rules are specified by a new media type called MEDIATYPE_AUDIO_REMOTE_CODER_OPTIONS_INFO. A new field AudioCoderOptionsInfo (of type IPM_AUDIO_CODER_OPTIONS_INFO) has been added to the IPM_MEDIA union. The CMR control is implemented by allowing the host application to specify one of two rules.

- The first rule, “CMR Tracking,” indicates that the transmit bit rate should follow the CMR value in the received packet.

- The second rule, “CMR Limit,” indicates that the transmit bit rate should follow the CMR value in the received packet with a maximum value specified by the preferred bit rate.

With both CMR rules, it is necessary to specify a preferred rate to avoid the specific case of determining what rate to transmit at before the first CMR value is received. The software will transmit at the preferred rate before the first packet is received or when a CMR value of 15 (don't care) is received from the opposite side.

The CMR rules are set in the AudioCoderOptionsInfo.unCoderOptions field by ORing in either CODER_OPT_AMR_CMR_TRACK or CODER_OPT_AMR_CMR_LIMIT.

Specifying a CMR rule is mandatory and the rules are mutually exclusive.

Note: The CMR rules are not used by the MEDIATYPE_AUDIO_LOCAL_CODER_OPTIONS_INFO eMediaTypes.

Supported RTP Payload Format

AMR supports two different formats for the RTP payload:

- “Bandwidth efficient,” to minimize the amount of network bandwidth.
- “Octet-aligned,” to make the packet parsing easier for the AMR application.

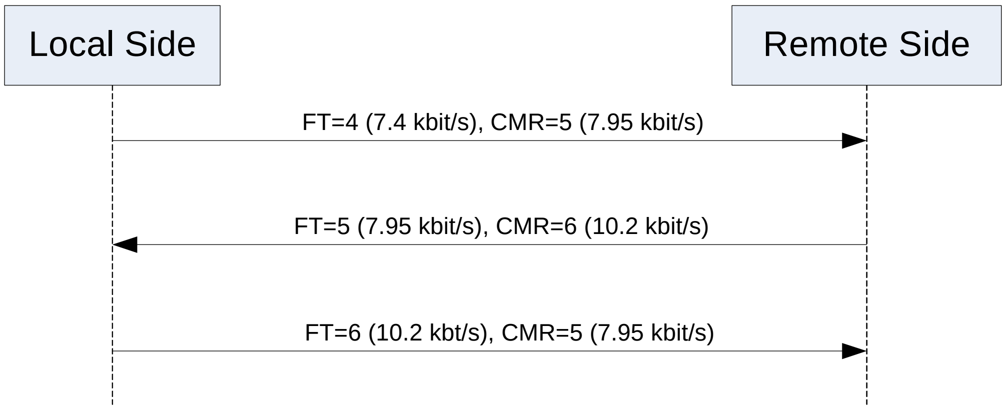
The AMR RTP payload format is specified in the new IPM_AUDIO_CODER_OPTIONS_INFO AudioCoderOptionsInfo structure (in the IPM_MEDIA structure). The MEDIATYPE_AUDIO_LOCAL_CODER_OPTIONS_INFO media type controls the receive side, and the MEDIATYPE_AUDIO_REMOTE_CODER_OPTIONS_INFO media type controls the transmit side.

The IPM_MEDIA structure contains a union with an IPM_AUDIO_CODER_OPTIONS_INFO structure. The AMR RTP payload format is controlled by ORing in either CODER_OPT_AMR_EFFICIENT or CODER_OPT_AMR_OCTET in the AudioCoderOptionsInfo.unCoderOptions field.

Note: Specifying an AMR RTP payload is mandatory and the formats are mutually exclusive.

Example 1

In this example, the host application wants to transmit at the bit rate requested by the incoming CMR value. The following diagram depicts this use case, where the local side is the host application.


Local Side
Remote Side
FT=4 (7.4 kbit/s), CMR=5 (7.95 kbit/s)
FT=5 (7.95 kbit/s), CMR=6 (10.2 kbit/s)
FT=6 (10.2 kbt/s), CMR=5 (7.95 kbit/s)

Note: While the diagram above implies an immediate reaction to a CMR from the other side, in reality, the other side's response to a CMR may take a few packets.

The sequence of activities is as follows:

1. The host is transmitting at the preferred bit rate set by the host application via the remote audio coder settings, in this case 7.4 kbit/s.

2. The host sets its CMR value to 7.95 kbit/s via the local audio coder setting.

3. The host sets its RTP payload to efficient bandwidth and CMR tracking mode.

4. The host receives its first packet from the remote side with the CMR value higher than the preferred rate.

5. The host changes the transmitted bit rate to match the CMR value.

The following code demonstrates the configuration required to handle this scenario.

```
...
/* Setup IP address here */

// Local Audio Coder
ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eCoderType = CODER_TYPE_AMRNB_7_95k;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eFrameSize = CODER_FRAMESIZE_20;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unFramesPerPkt = 1;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eVadEnable = CODER_VAD_ENABLE
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unCoderPayloadType = 96;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unRedPayloadType = 0
unCount++;

// Remote Audio Coder
ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eCoderType = CODER_TYPE_AMRNB_7_4k;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eFrameSize = CODER_FRAMESIZE_20;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unFramesPerPkt = 1;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eVadEnable = CODER_VAD_ENABLE
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unCoderPayloadType = 96;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unRedPayloadType = 0
```

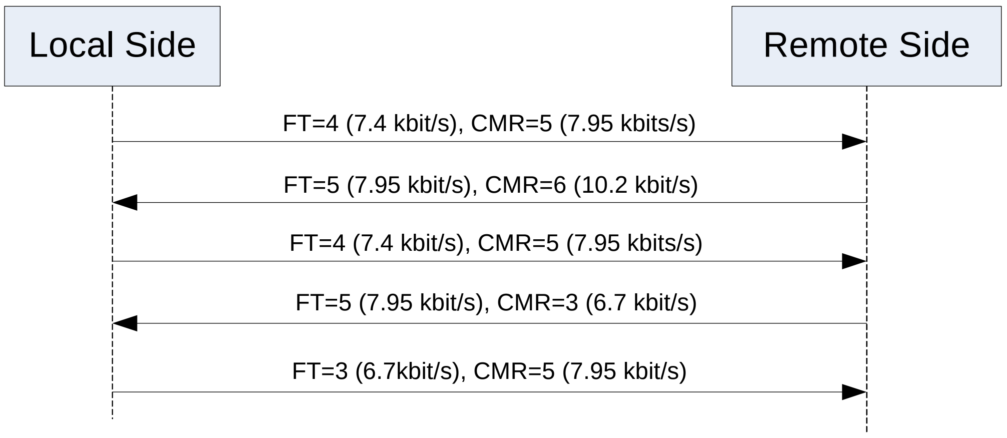
```
unCount++;

ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_OPTIONS_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unVersion =
                                                    IPM_AUDIO_CODER_OPTIONS_INFO_VERSION;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unCoderOptions=
                                                    CODER_OPT_AMR_EFFICIENT;
unCount++;

ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_OPTIONS_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unVersion =
                                                    IPM_AUDIO_CODER_OPTIONS_INFO_VERSION;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unCoderOptions=
                                            CODER_OPT_AMR_CMR_TRACK | CODER_OPT_AMR_EFFICIENT;
unCount++;
ipmMediaInfo.unCount = unCount;
...
```

Example 2

In this example, the host application wants to limit the transmitted bit rate to less than the preferred value. The following diagram shows this case, where the local side is the host application.


Local Side
Remote Side
FT=4 (7.4 kbit/s), CMR=5 (7.95 kbits/s)
FT=5 (7.95 kbit/s), CMR=6 (10.2 kbit/s)
FT=4 (7.4 kbit/s), CMR=5 (7.95 kbits/s)
FT=5 (7.95 kbit/s), CMR=3 (6.7 kbit/s)
FT=3 (6.7kbit/s), CMR=5 (7.95 kbit/s)

Note: While the diagram above implies an immediate reaction to a CMR from the other side, in reality, the other side's response to a CMR may take a few packets.

The sequence of activities is as follows:

1. The host is transmitting at the preferred bit rate set by the host application via the remote audio coder settings, in this case 7.4 kbit/s.

2. The host sets its initial CMR value to 7.95 kbit/s via the local audio coder setting.

3. The host sets its RTP payload to efficient bandwidth and CMR limit mode.

4. The host receives it first packet from the remote side with the CMR value higher than the preferred rate.

5. The host leaves the transmitted bit rate at the preferred value.

6. The host receives a packet from the remote side with a CMR value less than the preferred rate.

7. The host changes the transmitted bit rate to match the received CMR value.

The following code demonstrates the configuration required to handle this scenario.

```
...
/* Setup IP address here */

// Local Audio Coder
ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eCoderType =
CODER_TYPE_AMRNB_7_95k;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eFrameSize = CODER_FRAMESIZE_20;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unFramesPerPkt = 1;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eVadEnable = CODER_VAD_ENABLE
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unCoderPayloadType = 96;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unRedPayloadType = 0
unCount++;

// Remote Audio Coder
ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eCoderType = CODER_TYPE_AMRNB_7_4k;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eFrameSize = CODER_FRAMESIZE_20;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unFramesPerPkt = 1;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.eVadEnable = CODER_VAD_ENABLE
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unCoderPayloadType = 96;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderInfo.unRedPayloadType = 0
unCount++;

ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_LOCAL_CODER_OPTIONS_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo = {0};
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unVersion =
                                                IPM_AUDIO_CODER_OPTIONS_INFO_VERSION;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unCoderOptions=
                                              CODER_OPT_AMR_CMR_LIMIT | CODER_OPT_AMR_EFFICIENT;
unCount++;

ipmMediaInfo.MediaData[unCount].eMediaType = MEDIATYPE_AUDIO_REMOTE_CODER_OPTIONS_INFO;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo = {0};
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unVersion =
                                                IPM_AUDIO_CODER_OPTIONS_INFO_VERSION;
ipmMediaInfo.MediaData[unCount].mediaInfo.AudioCoderOptionsInfo.unCoderOptions=
                                                CODER_OPT_AMR_EFFICIENT;
unCount++
ipmMediaInfo.unCount = unCount;
```

Using Enhanced Variable Rate Codecs

Enhanced Variable Rate Codecs (EVRCs) as specified in *RFC 3558 - RTP Payload Format for Enhanced Variable Rate Codecs (EVRC) and Selectable Mode Vocoders (SMV)* are supported.

When using EVRCs, more than one codec data frame may be included in a single interleaved/bundled packet by a sender. This is accomplished by:

- Bundling - A technique used to spread the transmission overhead of the RTP and payload header over multiple vocoder frames.
- Interleaving - A technique used to reduce the listener's perception of data loss by spreading such a loss over non-consecutive vocoder frames.

EVRC and similar vocoders can compensate for an occasional lost frame, but speech quality degrades exponentially with consecutive frame loss.

Bundling is done using the Frames Per Packet field in the MEDIATYPE_AUDIO_REMOTE_CODER_INFO structure, as is the case with most codecs. Frames per packet values of 1 to 3 are allowed. Note that the RFC definition of bundle describes the number of additional frames per packet, so a frame per packet value of 1 would be the same as a bundle value of 0.

Interleaving is accomplished using parameter 1 of the MEDIATYPE_AUDIO_REMOTE_CODER_OPTIONS_INFO structure. This parameter can be set from 0 to 7; 0 indicates do not use interleaving. Interleaving should only be used when the frames per packet value is 2 or more.

When using the IP Media Library API, interleaving option can be specified in the IPM_AUDIO_CODER_OPTIONS_INFO data structure.

Resource reservation of audio coders

The following information, which is applicable to the ATCA Multimedia Platform, should be added to the programming guide, possibly as a new chapter.

Resource Reservation of Audio Coders

The following sections describe the use of the IPML API and Device Management API in various scenarios for resource reservation of audio coders.

Outbound Call

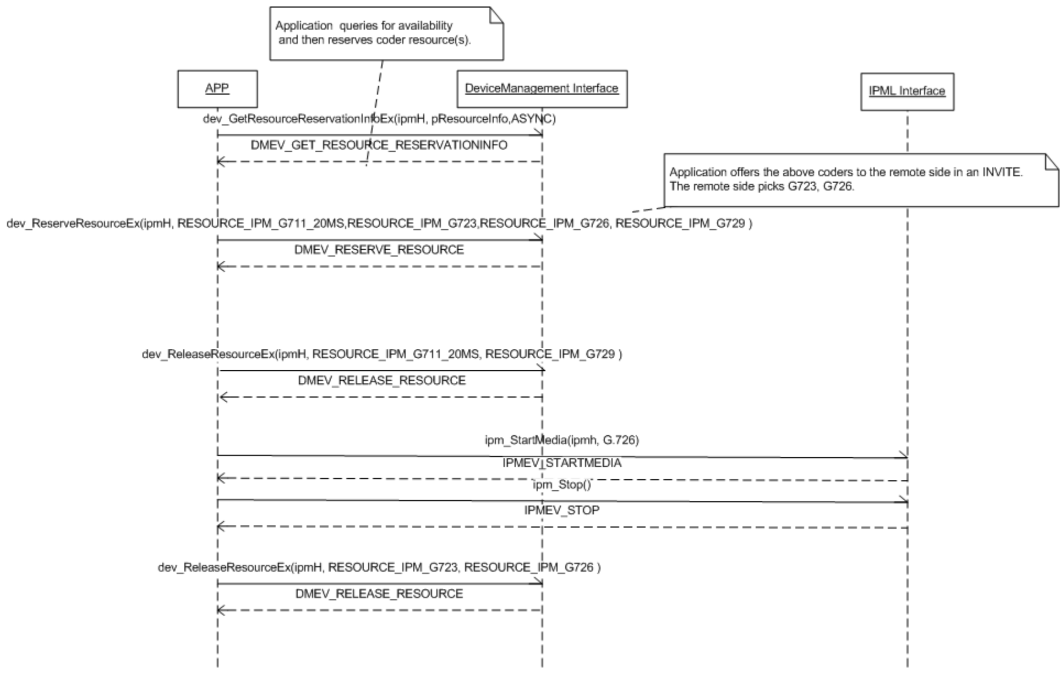
1. Application reserves coders (RESOURCE_IPM_G711_20MS, RESOURCE_IPM_G723, RESOURCE_IPM_G726, RESOURCE_IPM_G729) and offers the reserved coders to the remote side.

2. The remote side picks G723, G726.

3. Application releases (RESOURCE_IPM_G711_20MS, RESOURCE_IPM_G729). This leaves (RESOURCE_IPM_G723, RESOURCE_IPM_G726) as the reserved audio coders.

4. The application proceeds with the call.

5. After the call is done, the application releases the coders (RESOURCE_IPM_G723, RESOURCE_IPM_G726). (Alternatively, the application can release all the reserved coders by specifying the enum RESOURCE_IPM_ALL_AUDIO_CODERS.)



Application queries for availability and then reserves coder resource(s).
APP
DeviceManagement Interface
IPML Interface
dev_GetResourceReservationInfoEx(ipmH, pResourceInfo,ASYNC)
DMEV_GET_RESOURCE_RESERVATIONINFO
Application offers the above coders to the remote side in an INVITE. The remote side picks G723, G726.
dev_ReserveResourceEx(ipmH, RESOURCE_IPM_G711_20MS,RESOURCE_IPM_G723,RESOURCE_IPM_G726, RESOURCE_IPM_G729)
DMEV_RESERVE_RESOURCE
dev_ReleaseResourceEx(ipmH, RESOURCE_IPM_G711_20MS, RESOURCE_IPM_G729)
DMEV_RELEASE_RESOURCE
ipm_StartMedia(ipmh, G.726)
IPMEV_STARTMEDIA
ipm_Stop()
IPMEV_STOP
dev_ReleaseResourceEx(ipmH, RESOURCE_IPM_G723, RESOURCE_IPM_G726)
DMEV_RELEASE_RESOURCE

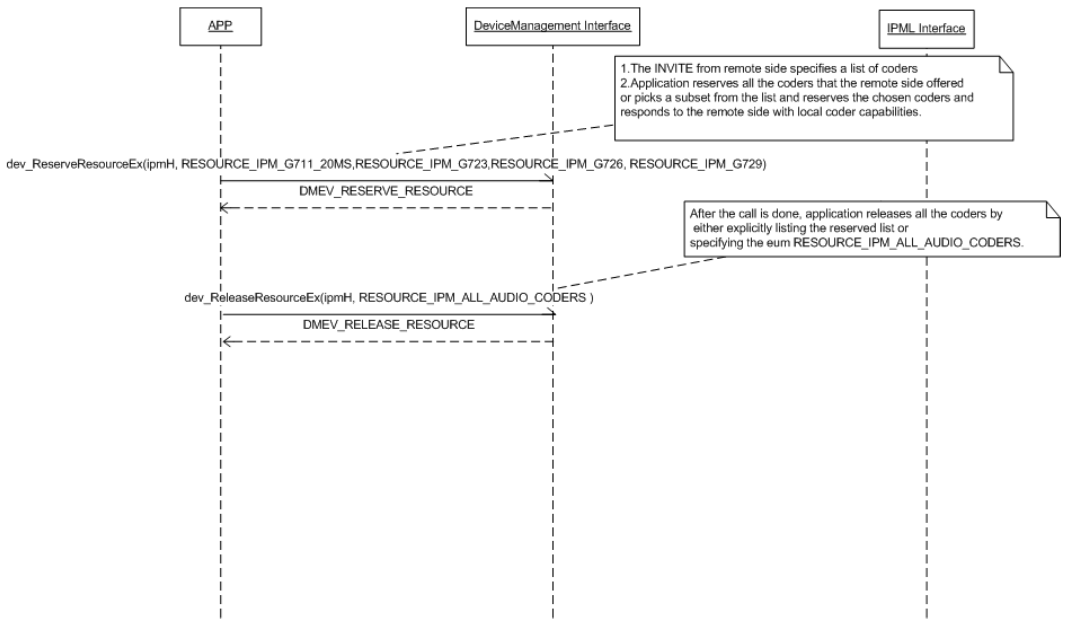
Inbound Call

1. The INVITE from remote side specifies a list of coders.

2. Application may choose to reserve all or some of the coders that the remote side offered and then respond to the remote side with local coder capabilities based on the reservation.

3. Application proceeds with the call.

4. After the call is disconnected, the application releases all the reserved coders by either explicitly listing the reserved list or specifying the enum RESOURCE_IPM_ALL_AUDIO_CODERS.


APP
DeviceManagement Interface
IPML Interface
1.The INVITE from remote side specifies a list of coders
2.Application reserves all the coders that the remote side offered or picks a subset from the list and reserves the chosen coders and responds to the remote side with local coder capabilities.
dev_ReserveResourceEx(ipmH, RESOURCE_IPM_G711_20MS,RESOURCE_IPM_G723,RESOURCE_IPM_G726, RESOURCE_IPM_G729)
DMEV_RESERVE_RESOURCE
After the call is done, application releases all the coders by either explicitly listing the reserved list or specifying the eum RESOURCE_IPM_ALL_AUDIO_CODERS.
dev_ReleaseResourceEx(ipmH, RESOURCE_IPM_ALL_AUDIO_CODERS)
DMEV_RELEASE_RESOURCE

Implicit Release by a Subsequent Successful Reserve Call

1. Application reserves (RESOURCE_IPM_G726, RESOURCE_IPM_G729).

2. Application uses RESOURCE_IPM_G726 for a media operation (ipm_StartMedia).

3. Application gets a REINVITE from the remote side requesting a change of coder.

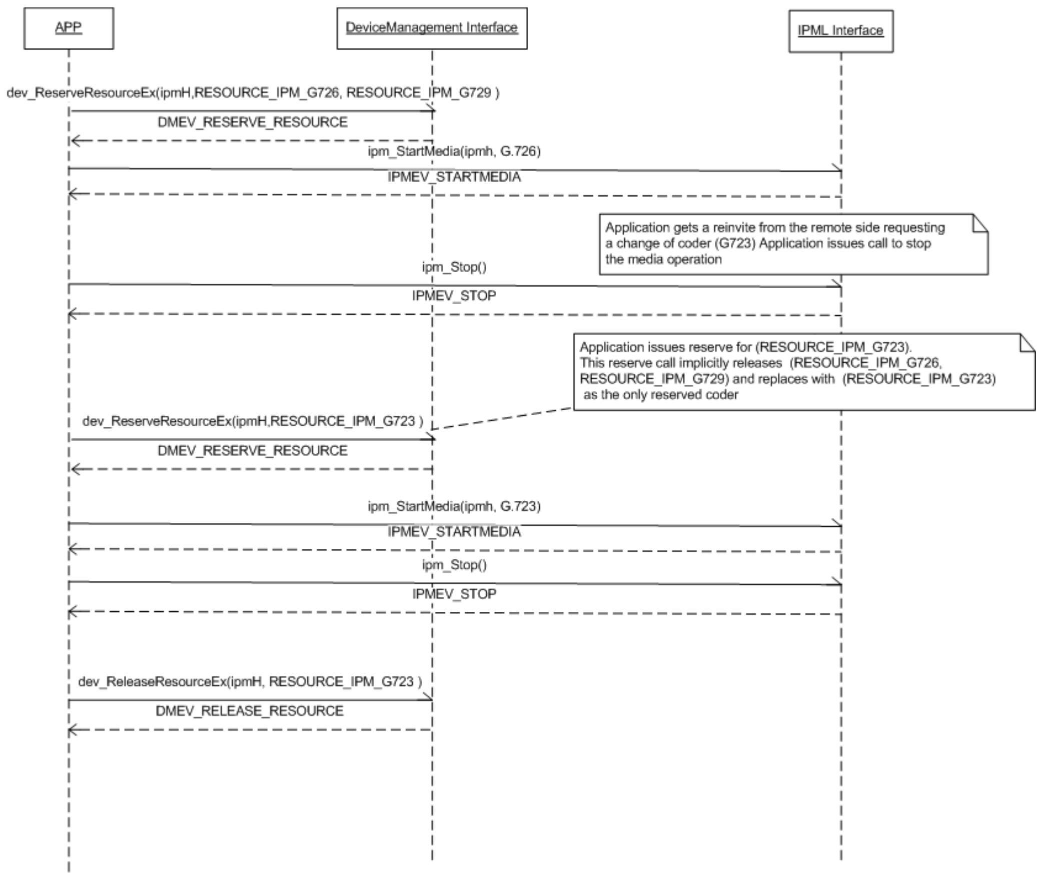
4. Application issues call to stop the media operation.

5. Application issues reserve for RESOURCE_IPM_G723. This reserve call implicitly releases (RESOURCE_IPM_G726,RESOURCE_IPM_G729) and replaces with RESOURCE_IPM_G723 as the only reserved coder.

6. Application uses RESOURCE_IPM_G723 for a media operation (ipm_StartMedia).

7. Application issues call to stop the media operation.

8. Application releases coder by issuing a call to release RESOURCE_IPM_G723.


APP
DeviceManagement Interface
IPML Interface
dev_ReserveResourceEx(ipmH,RESOURCE_IPM_G726, RESOURCE_IPM_G729)
DMEV_RESERVE_RESOURCE
ipm_StartMedia(ipmh, G.726)
IPMEV_STARTMEDIA
Application gets a reinvite from the remote side requesting a change of coder (G723) Application issues call to stop the media operation
ipm_Stop()
IPMEV_STOP
Application issues reserve for (RESOURCE_IPM_G723). This reserve call implicitly releases (RESOURCE_IPM_G726, RESOURCE_IPM_G729) and replaces with (RESOURCE_IPM_G723) as the only reserved coder
dev_ReserveResourceEx(ipmH,RESOURCE_IPM_G723)
DMEV_RESERVE_RESOURCE
ipm_StartMedia(ipmh, G.723)
IPMEV_STARTMEDIA
ipm_Stop()
IPMEV_STOP
dev_ReleaseResourceEx(ipmH, RESOURCE_IPM_G723)
DMEV_RELEASE_RESOURCE

Handling a Resource Reservation Failure

1. A call to reserve resources (RESOURCE_IPM_G726, RESOURCE_IPM_G729) fails for lack of available resources.

Note: The reserve fails when one or more of the resources requested is not available. The application can query to check on resource availability prior to issuing a reserve request. Otherwise, the application will need to query after the reserve fails to identify available/unavailable resources prior to re-issuing a reserve request.

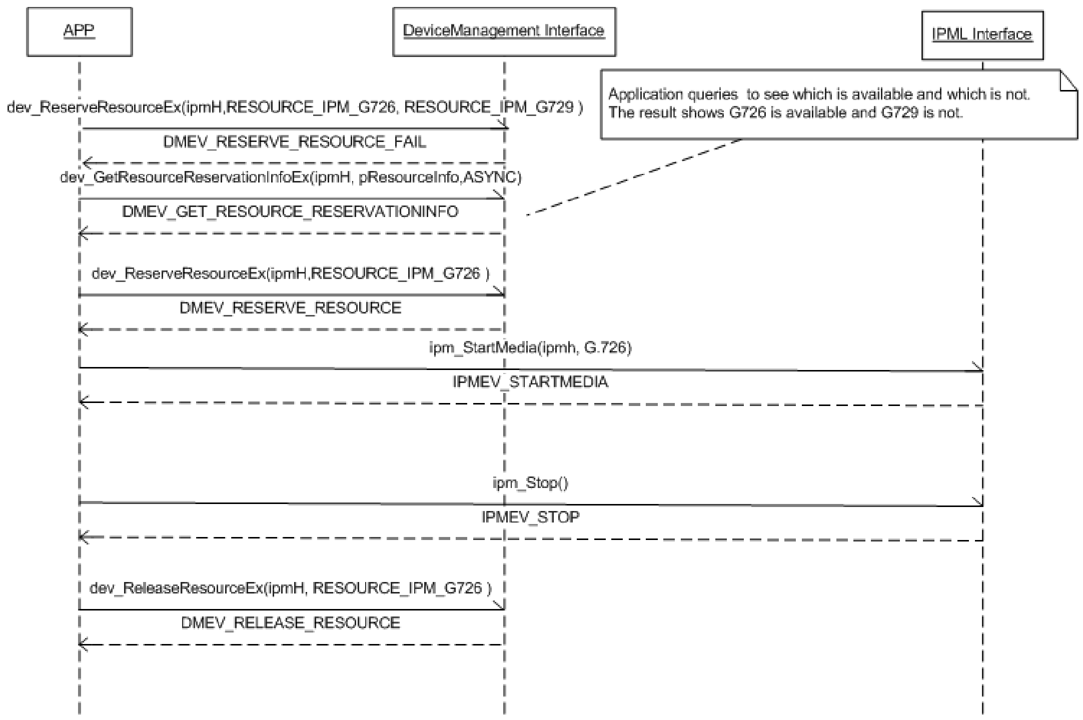
2. Application queries to check on resource availability (for this example, assume this returns RESOURCE_IPM_G726 as available).

3. Application reserves resource RESOURCE_IPM_G726.

4. Application starts media operation (ipm_StartMedia).

5. Application stops media operation (ipm_Stop).

6. Application issues call to release resource RESOURCE_IPM_G726.



APP
DeviceManagement Interface
IPML Interface
Application queries to see which is available and which is not. The result shows G726 is available and G729 is not.
dev_ReserveResourceEx(ipmH,RESOURCE_IPM_G726, RESOURCE_IPM_G729)
DMEV_RESERVE_RESOURCE_FAIL
dev_GetResourceReservationInfoEx(ipmH, pResourceInfo,ASYNC)
DMEV_GET_RESOURCE_RESERVATIONINFO
dev_ReserveResourceEx(ipmH,RESOURCE_IPM_G726)
DMEV_RESERVE_RESOURCE
ipm_StartMedia(ipmh, G.726)
IPMEV_STARTMEDIA
ipm_Stop()
IPMEV_STOP
dev_ReleaseResourceEx(ipmH, RESOURCE_IPM_G726)
DMEV_RELEASE_RESOURCE

Reservation State after a Subsequent Call to Resource Reservation Fails

1. Application issues call to reserve resources (RESOURCE_IPM_G726, RESOURCE_IPM_G729) and it succeeds.

2. Application starts media operation using G726.

3. Application receives a REINVITE request to use AMR coder.

4. Application stops media operation.

5. Application attempts to reserve RESOURCE_IPM_AMR – the call fails for lack of available resources. The previous successful reserved state is preserved. In this case, (RESOURCE_IPM_G726, RESOURCE_IPM_G729) were the previously reserved coders – that state is still preserved as the latest call to reserve failed.

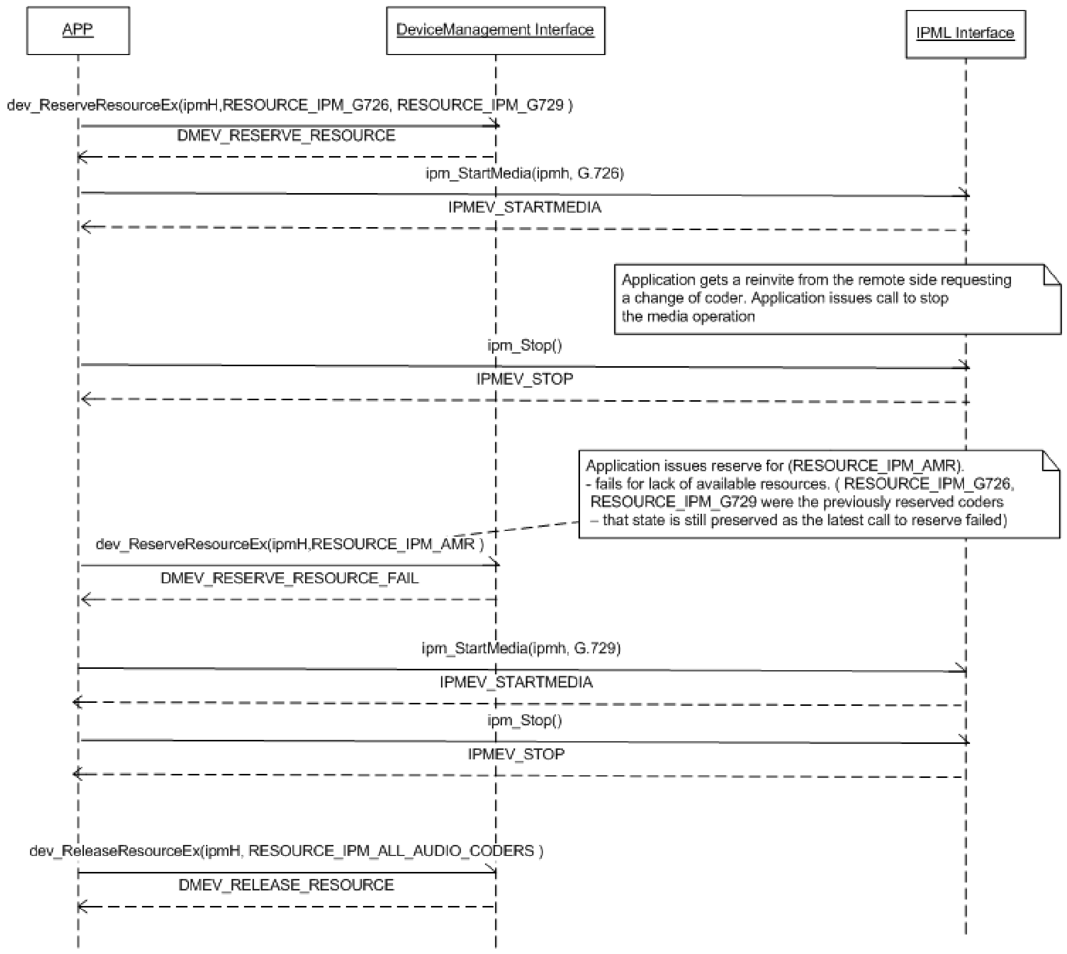
6. [Application may reject the REINVITE from the remote side and continue to use any of the reserved coders at this point]

7. Start Media Operation.

8. Stop Media Operation.

9. Application issues call to release resources (RESOURCE_IPM_G726, RESOURCE_IPM_G729). It may also use a single enum

RESOURCE_IPM_ALL_AUDIO_CODERS to release all the reserved audio coders as opposed to having to list every single one of them.


APP
DeviceManagement Interface
IPML Interface
dev_ReserveResourceEx(ipmH,RESOURCE_IPM_G726, RESOURCE_IPM_G729)
DMEV_RESERVE_RESOURCE
ipm_StartMedia(ipmh, G.726)
IPMEV_STARTMEDIA
Application gets a reinvite from the remote side requesting a change of coder. Application issues call to stop the media operation
ipm_Stop()
IPMEV_STOP
Application issues reserve for (RESOURCE_IPM_AMR). - fails for lack of available resources. (RESOURCE_IPM_G726, RESOURCE_IPM_G729 were the previously reserved coders – that state is still preserved as the latest call to reserve failed)
dev_ReserveResourceEx(ipmH,RESOURCE_IPM_AMR)
DMEV_RESERVE_RESOURCE_FAIL
ipm_StartMedia(ipmh, G.729)
IPMEV_STARTMEDIA
ipm_Stop()
IPMEV_STOP
dev_ReleaseResourceEx(ipmH, RESOURCE_IPM_ALL_AUDIO_CODERS)
DMEV_RELEASE_RESOURCE

3.3.11 Dialogic® Multimedia API Library Reference

Update to MM_AUDIO_CODEC data structure

The unCoding field should include additional values for native play and record. These values have not been added in the mmlib.h header file. The mmlib.h header file is available in the TAR file and should be copied to **/usr/dialogic/inc** after installation is complete.

```
#define MM_DATA_FORMAT_AMR_NB_4_75K                    0x1A
#define MM_DATA_FORMAT_AMR_NB_5_15K                    0x1B
#define MM_DATA_FORMAT_AMR_NB_5_90K                    0x1C
#define MM_DATA_FORMAT_AMR_NB_6_70K                    0x1D
#define MM_DATA_FORMAT_AMR_NB_7_40K                    0x1E
#define MM_DATA_FORMAT_AMR_NB_7_95K                    0x1F
#define MM_DATA_FORMAT_AMR_NB_10_20K                   0x20
#define MM_DATA_FORMAT_AMR_NB_12_20K                   0x21
#define MM_DATA_FORMAT_G723_1_5_30K                    0x22
#define MM_DATA_FORMAT_G723_1_6_30K                    0x23
```

Note: Field validation is now performed on the unCoding field. Select from the list of valid values. Setting an unsupported value results in MMEV_PLAY_ACK_FAIL event in **mm_play( )** and **mm_record( )**.

Update to MM_VIDEO_CODEC data structure

For **mm_record( )**, none of the fields in this data structure may be modified, as video transcoding is not currently supported.

3.3.12 Dialogic® Multimedia API Programming Guide

There are currently no updates to this document.

3.3.13 Dialogic® Multimedia File Conversion Tools User Guide

There are currently no updates to this document.

3.3.14 Dialogic® Standard Runtime Library API Library Reference

Update to **sr_getfdcnt( )** and **sr_getfdinfo( )** functions (IPY00045054).

The following caution should be added for **sr_getfdcnt( )** and **sr_getfdinfo( )** functions:

- The application must call **sr_getfdcnt( )** and **sr_getfdinfo( )** before calling any other Dialogic® API, if the application wants to use SELECT to retrieve SRL events on Linux.

3.3.15 Dialogic® Standard Runtime Library API Programming Guide

There are currently no updates to this document.

3.3.16 Dialogic® Voice API Library Reference

There are currently no updates to this document.

3.3.17 Dialogic® Voice API Programming Guide

There are currently no updates to this document.

3.4 Remote Control Interfaces Documentation

This section contains updates to the following documents:

- Dialogic® MSML Media Server Software User's Guide

3.4.1 Dialogic® MSML Media Server Software User's Guide

There are currently no updates to this document.

3.5 Demonstration Software Documentation

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Dialogic® 3G-324M Multimedia Gateway Demo Guide
- Dialogic® Global Call API Demo Guide
- Dialogic® Multimedia Demo Guide

3.5.1 Dialogic® 3G-324M Multimedia Gateway Demo Guide

There are currently no updates to this document.

Note: A new document (05-2643-002) is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.5.2 Dialogic® Global Call API Demo Guide

There are currently no updates to this document.

Note: A new document (05-2439-003) is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.

3.5.3 Dialogic® Multimedia Demo Guide

There are currently no updates to this document.

Note: A new document (05-2456-004) is now available on the documentation bookshelf. See the Revision History section of the document for a description of the changes.