Development Guide

eCall In-Band Modem Library

SIERRA WIRELESS®

4112356
6.0
February13, 2015

Important Notice

Due to the nature of wireless communications, transmission and reception of data can never be guaranteed. Data may be delayed, corrupted (i.e., have errors) or be totally lost. Although significant delays or losses of data are rare when wireless devices such as the Sierra Wireless modem are used in a normal manner with a well-constructed network, the Sierra Wireless modem should not be used in situations where failure to transmit or receive data could result in damage of any kind to the user or any other party, including but not limited to personal injury, death, or loss of property. Sierra Wireless accepts no responsibility for damages of any kind resulting from delays or errors in data transmitted or received using the Sierra Wireless modem, or for failure of the Sierra Wireless modem to transmit or receive such data.

Safety and Hazards

Do not operate the Sierra Wireless modem in areas where cellular modems are not advised without proper device certifications. These areas include environments where cellular radio can interfere such as explosive atmospheres, medical equipment, or any other equipment which may be susceptible to any form of radio interference. The Sierra Wireless modem can transmit signals that could interfere with this equipment. Do not operate the Sierra Wireless modem in any aircraft, whether the aircraft is on the ground or in flight. In aircraft, the Sierra Wireless modem **MUST BE POWERED OFF**. When operating, the Sierra Wireless modem can transmit signals that could interfere with various onboard systems.

Note: *Some airlines may permit the use of cellular phones while the aircraft is on the ground and the door is open. Sierra Wireless modems may be used at this time.*

The driver or operator of any vehicle should not operate the Sierra Wireless modem while in control of a vehicle. Doing so will detract from the driver or operator's control and operation of that vehicle. In some states and provinces, operating such communications devices while in control of a vehicle is an offence.

Limitations of Liability

This manual is provided "as is". Sierra Wireless makes no warranties of any kind, either expressed or implied, including any implied warranties of merchantability, fitness for a particular purpose, or noninfringement. The recipient of the manual shall endorse all risks arising from its use.

The information in this manual is subject to change without notice and does not represent a commitment on the part of Sierra Wireless. SIERRA WIRELESS AND ITS AFFILIATES SPECIFICALLY DISCLAIM LIABILITY FOR ANY AND ALL DIRECT, INDIRECT, SPECIAL, GENERAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES INCLUDING, BUT NOT LIMITED TO, LOSS OF PROFITS OR REVENUE OR ANTICIPATED PROFITS OR REVENUE ARISING OUT OF THE USE OR INABILITY TO USE ANY SIERRA WIRELESS PRODUCT, EVEN IF SIERRA WIRELESS AND/OR ITS AFFILIATES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR THEY ARE FORESEEABLE OR FOR CLAIMS BY ANY THIRD PARTY.

Notwithstanding the foregoing, in no event shall Sierra Wireless and/or its affiliates aggregate liability arising under or in connection with the Sierra Wireless product, regardless of the number of events, occurrences, or claims giving rise to liability, be in excess of the price paid by the purchaser for the Sierra Wireless product.

Customer understands that Sierra Wireless is not providing cellular or GPS (including A-GPS) services. These services are provided by a third party and should be purchased directly by the Customer.

SPECIFIC DISCLAIMERS OF LIABILITY: CUSTOMER RECOGNIZES AND ACKNOWLEDGES SIERRA WIRELESS IS NOT RESPONSIBLE FOR AND SHALL NOT BE HELD LIABLE FOR ANY DEFECT OR DEFICIENCY OF ANY KIND OF CELLULAR OR GPS (INCLUDING A-GPS) SERVICES.

Patents

This product may contain technology developed by or for Sierra Wireless Inc.

This product includes technology licensed from QUALCOMM®.

This product is manufactured or sold by Sierra Wireless Inc. or its affiliates under one or more patents licensed from InterDigital Group and MMP Portfolio Licensing.

Copyright

Trademarks

Sierra Wireless®, AirPrime®, AirLink®, AirVantage®, WISMO®, ALEOS®, and the Sierra Wireless and Open AT logos are registered trademarks of Sierra Wireless, Inc. or one of its subsidiaries.

Watcher® is a registered trademark of Netgear, Inc., used under license.

Windows® and Windows Vista® are registered trademarks of Microsoft Corporation.

Macintosh® and Mac OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries.

QUALCOMM® is a registered trademark of QUALCOMM Incorporated. Used under license.

Other trademarks are the property of their respective owners.

Contact Information

Sales Desk:	Phone:	1-604-232-1488
	Hours:	8:00 AM to 5:00 PM Pacific Time
	Contact:	http://www.sierrawireless.com/sales
Post:	Sierra Wireless 13811 Wireless Way Richmond, BC Canada V6V 3A4	
Technical Support:	support@sierrawireless.com	
RMA Support:	repairs@sierrawireless.com	
Fax:	1-604-231-1109	
Web:	http://www.sierrawireless.com/	

Consult our website for up-to-date product descriptions, documentation, application notes, firmware upgrades, troubleshooting tips, and press releases: www.sierrawireless.com

Document History

Version	Date	Updates
001	July 28, 2010	Creation
002	May 9, 2011	Updates
003	October 7, 2011	Added APIs inbm_SetOpts and inbm_GetOpts. Added supporting information for these APIs in the table presented in the eCall In-Band Modem Library APIs vs. Voice Call State section. Added information and table entries to the HLAP PSAP Messages Waiting Timer Values section.
4.0	July 14, 2014	Updated document legal boilerplate. Updated information for (T5) in Table 4. Updated section 4.6.5. Added information for event INBM_HLACK_RECEIVED_CLEARDOWN in Table 13.
5.0	February 13, 2015	Updated legal boilerplate content. Added INBM_WAIT_UNMUTE_AFTER_AL_ACK to sections 4.7.4 and 4.7.4.2.

Contents

List of Figures

List of Tables

1. Introduction

This document provides Sierra Wireless customers with full description of the APIs associated with the eCall In-Band Modem Library.

1.1. Related Documents

[1] eCall In-Band Modem Library AT Commands Interface Guide
Reference: WM_DEV_INBM_UGD_001

[2] Intelligent transport systems — ESafety — eCall high level application requirements (HLAP) (CEN/TC 278)

[3] 3GPP TS 26.267 V9.3.0 - V8.6.0

[4] 3GPP TS 26.268 V9.4.0 - V8.6.0

[5] 3GPP TS 26.269 V9.2.0 - V8.3.0

1.2. Abbreviations

Table 1. Abbreviations

Term/Acronym	Description/Explanation
ADL	Application development layer
ACK	Acknowledgement
GSM	Global System for Mobile communications
IVS	In-Vehicle System
MSD	Minimum Set of Data
NACK	Negative Acknowledgement
PCM	Pulse Code Modulation
PSAP	Public Safety Answering Point

1.3. Glossary

In/out/Glb: used in API parameters:

- “In” if the parameter is given to the API
- “Out” if the parameter is the result of the API
- “Glb” (for Global) if the parameter is used for both

2. Global Architecture

The eCall In-Band Modem Library interfaces with the AirPrime Embedded Modules using the OS and Open AT Application Framework Firmware as shown below.

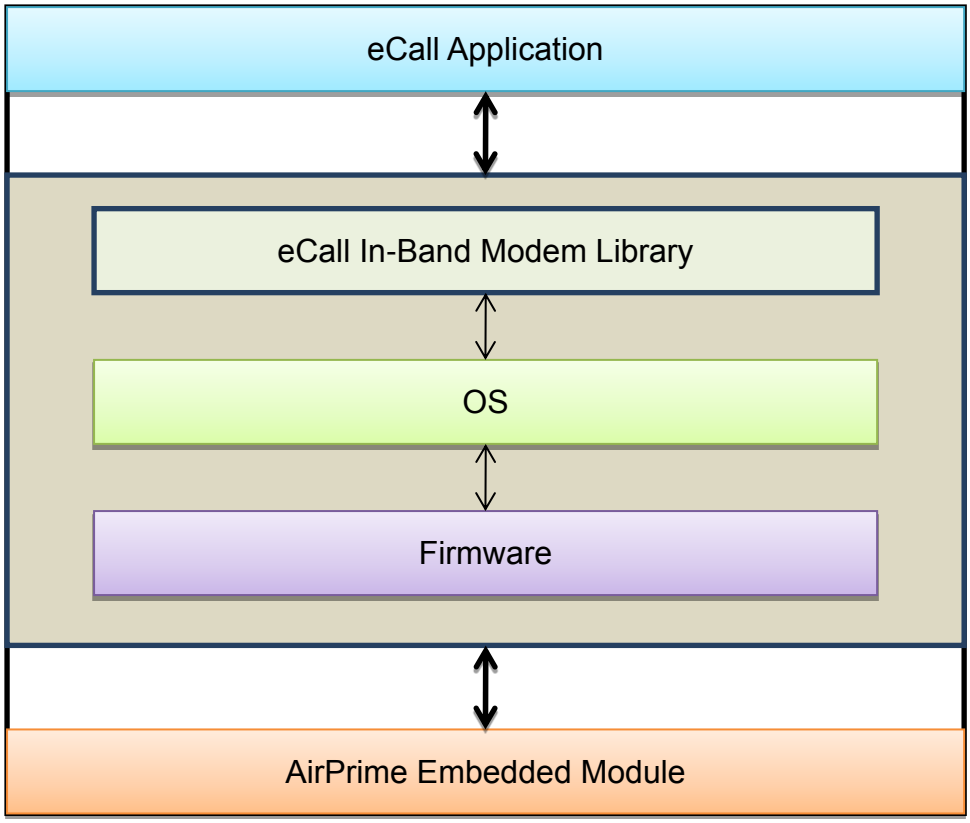

Figure 1. Interface Diagram of eCall In-Band Modem Library with Firmware/ OS and AirPrime Embedded Module

The eCall In-Band Modem Library interfaces directly to the OS which in turn uses Open AT Application Framework Firmware to access the Embedded Module Hardware.

Furthermore, an application compliant to the 3GPP specification for “eCall” service for vehicles can be developed using eCall In-Band Modem Library APIs.

The different layers of the system are presented in the table below.

Table 2. Global Architecture System Layers

Layer	Explanation
eCall Application	Application based on the guidelines of 3GPP and EU norms for eCall and according to the business logics.
eCall In-Band Modem Library	A Library implementing “eCall” IVS eCall In-Band Modem.
OS	The eCall In-Band Modem Library –OS APIs offer to eCall In-Band Modem Library access to the platform resources such as memory, IRQ service and audio service and other physical interfaces.
Open AT Application Framework Firmware	The Firmware provides interface to the OS to give access to the AirPrime Embedded Module.
AirPrime Embedded Module	The module provides GSM/GPRS capabilities to the “eCall” Application in particular Emergency call feature

3. Feature Description

Customers are provided with an advanced set of APIs that give them the access to the core IVS modem functionality based on 3GPP specification to develop vehicle eCall applications on AirPrime Embedded Modules running the OS.

The “eCall” application is an interoperable in-vehicle emergency call service devised by EU standardization bodies for emergency vehicle tracking in case of an event of any accident/emergency situation.

The “eCall” equipment makes automatically/after instruction from the vehicle drivers, an emergency call to the local emergency agencies i.e. PSAP and provides a set of MSD (Minimum Set of Data) which would assist the PSAP to locate the vehicle and provide help. The MSD can include, e.g. vehicle location information, time stamp, number of passengers, Vehicle Identification Number (VIN), and other relevant accident information.

The “eCall” MSD information is sent either immediately following the establishment of the voice call or at any point later during the emergency voice call. The integrity of the “eCall” data sent from the vehicle to the PSAP is ensured by the eCall In-Band Modem Library.

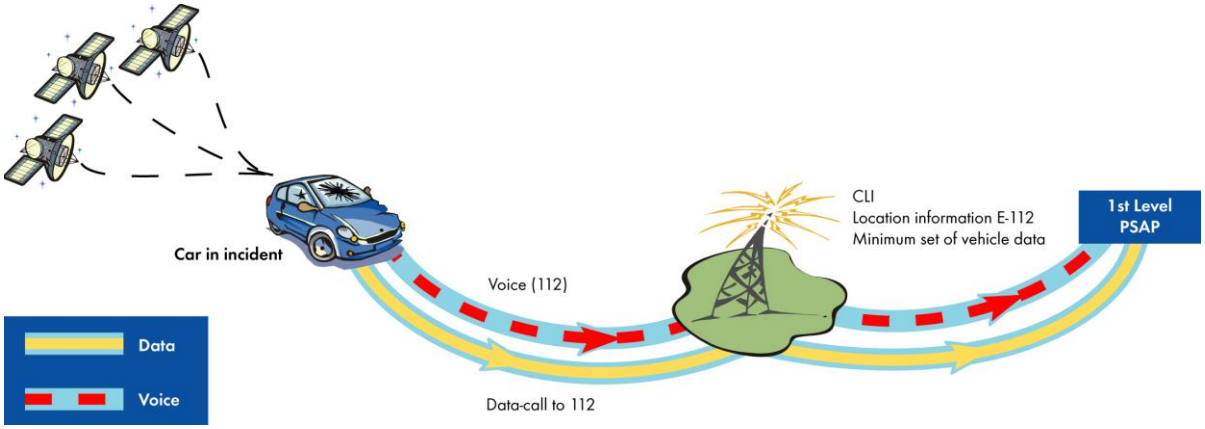

Figure 2. eCall System Overview

The car has an integrated device hosting AirPrime Embedded Module for GSM and GPS functionalities and managing “eCall” Service over GSM network.

3.1. Use Case

The APIs are an abstraction layer to the core IVS modem functionality of the eCall In-Band Modem Library. Using these APIs, the “eCall” Application can send MSD data over voice call either initiated by the vehicle application (PUSH mode) or by PSAP server (PULL mode).

The OS-based “eCall” Application has to open an eCall In-Band Modem channel to send MSD data over an established voice call (i.e. emergency or normal scenario). This eCall In-Band Modem channel is an abstract channel from an application perspective which is used to send MSD data.

During MSD data transmission the local audio interface (microphone and loud speaker) are muted by the eCall In-Band Modem Library and un-muted after successful data transmission.

3.1.1. PUSH Mode

In PUSH mode, the vehicle “eCall” Application initiates the communication with the PSAP server. The “eCall” Application requests the PSAP server to request for MSD data. For this, the “eCall” Application sends a SEND MSD REQUEST message to PSAP server. After receiving the SEND MSD REQUEST message, PSAP server requests for MSD data by sending SEND MSD messages. As soon as, “eCall” application receives the SEND MSD messages as per “eCall” specification, it sends the MSD data to the PSAP server.

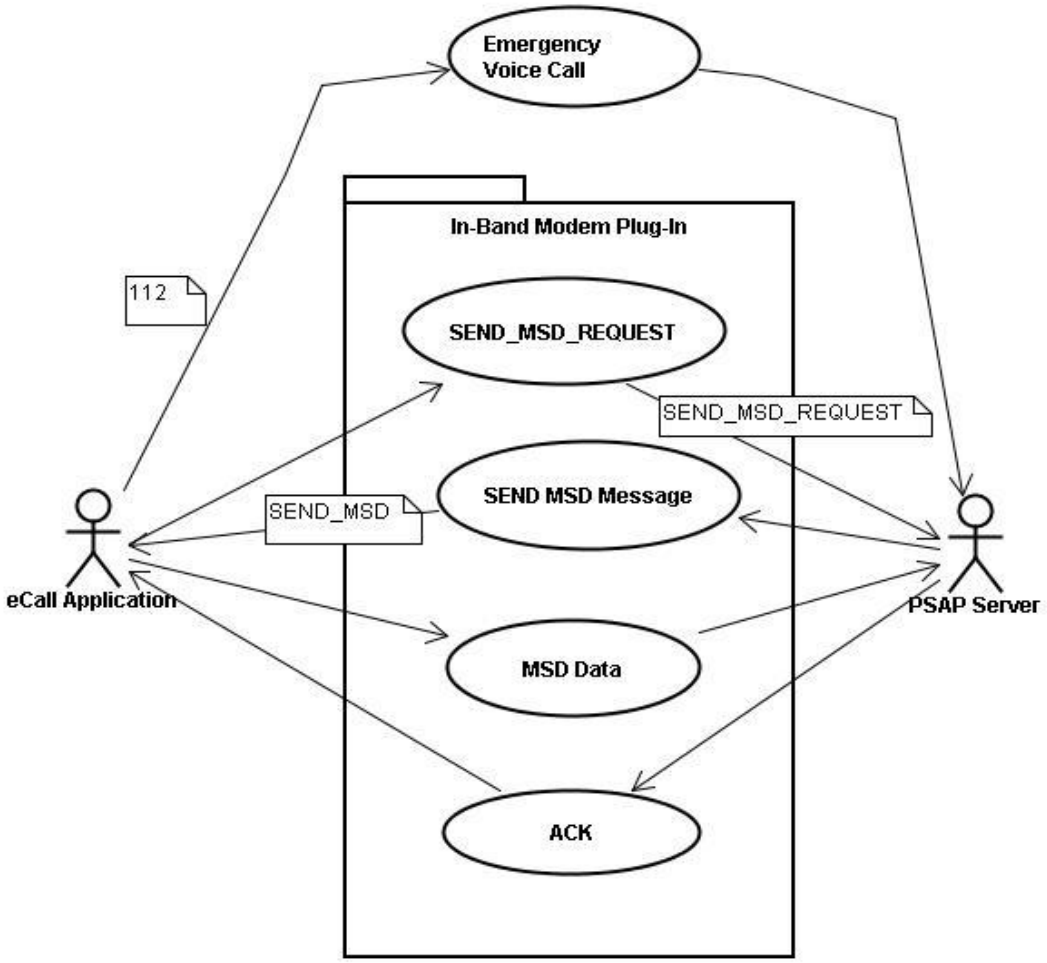

Figure 3. Use Case of Vehicle Triggered Emergency Call (PUSH Mode)

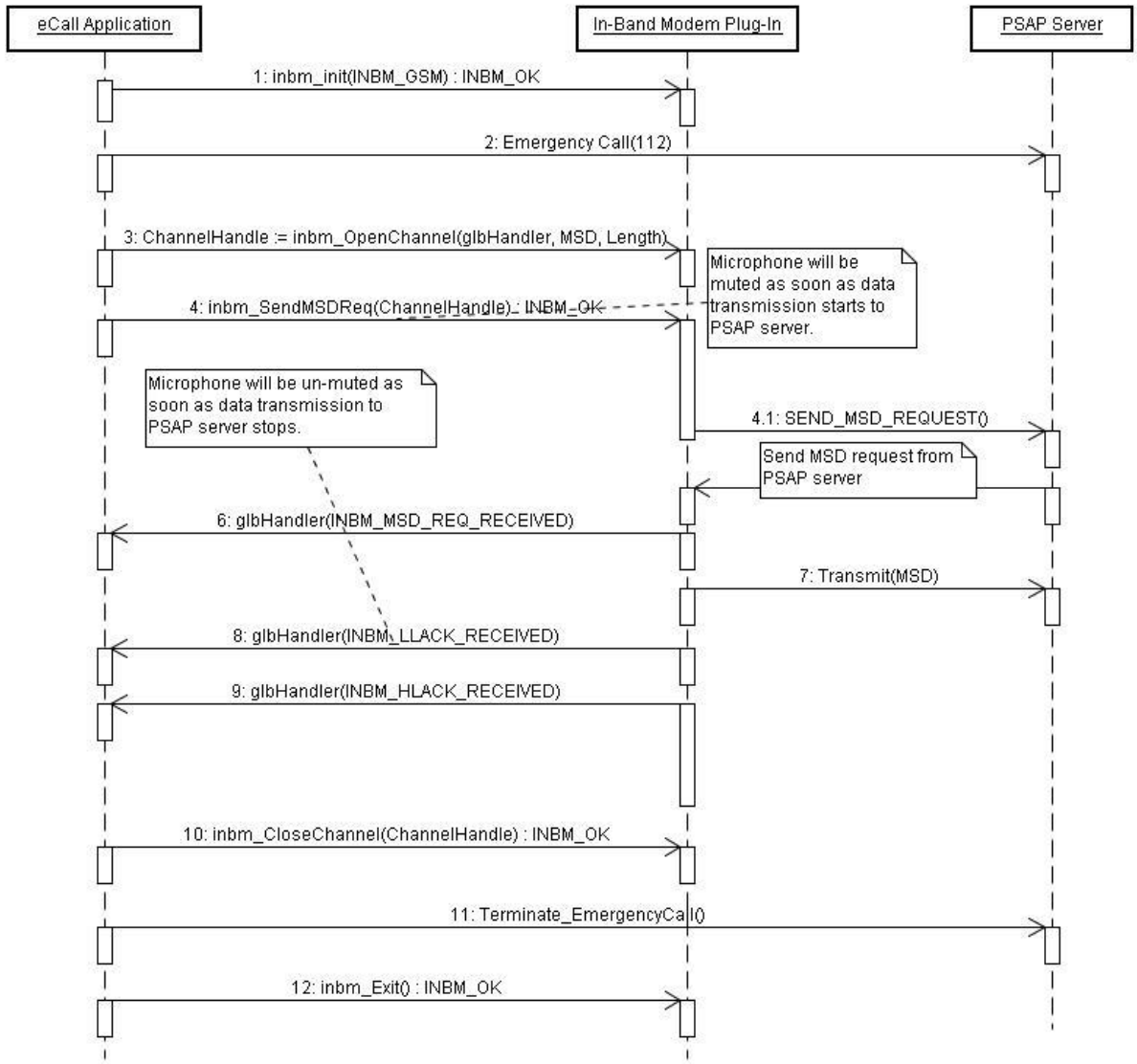

Figure 4. Sequence Diagram of PUSH mode

3.1.2. PULL Mode

In PULL mode, the PSAP server initiates the communication with the vehicle “eCall” application. The PSAP server requests for MSD data by sending SEND MSD message. As soon as, “eCall” application receives the SEND MSD message as per “eCall” specification, it sends the MSD data to the PSAP server.

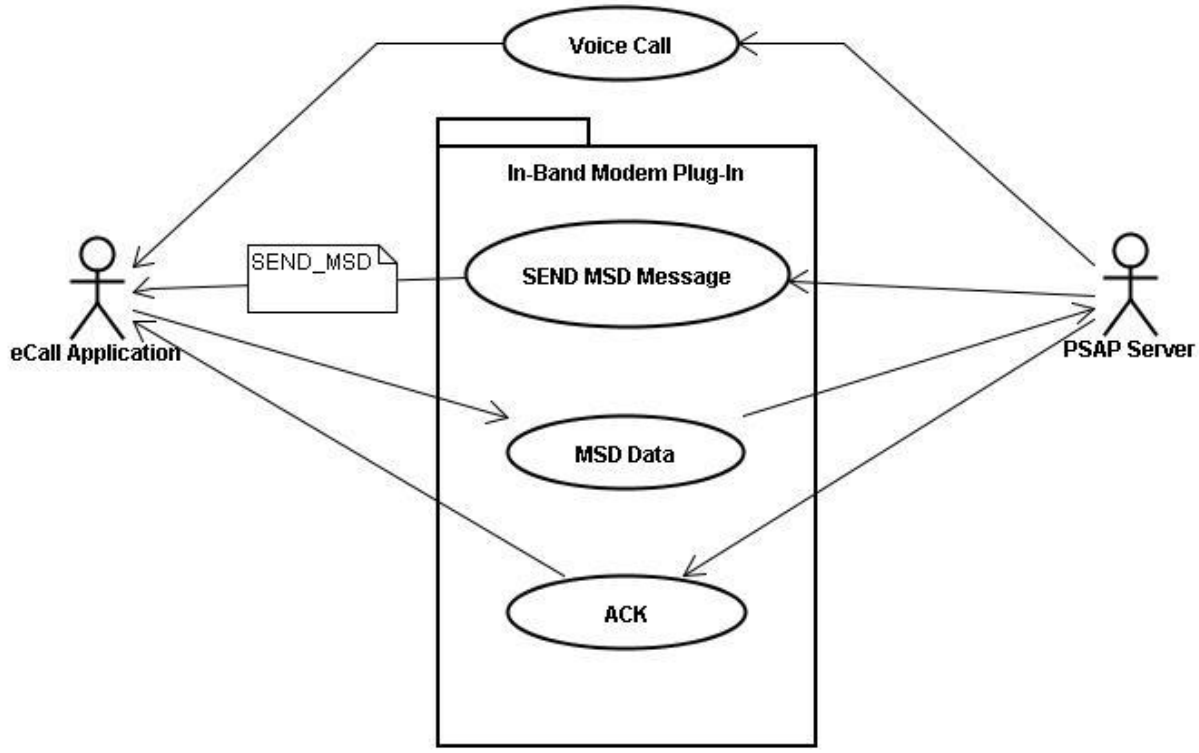

Figure 5. Use Case of PSAP Server Triggered Voice Call (PULL Mode)

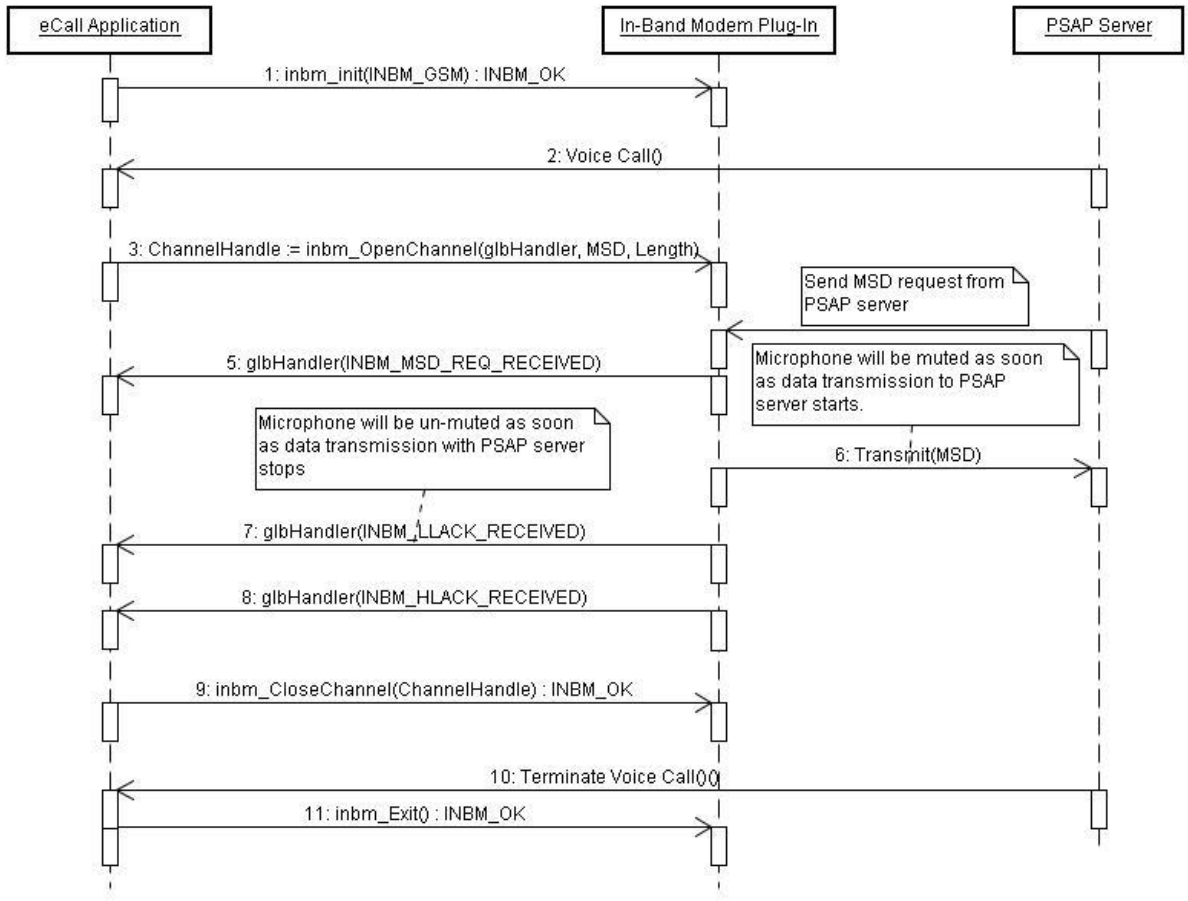

Figure 6. Sequence Diagram of PULL Mode

3.2. eCall In-Band Modem Library APIs

The eCall In-Band Modem Library APIs can be grouped into three categories:

- Library Control APIs: Init, Exit, Get State, Get Version
- Channel Control APIs: Open, Close, Get Channel State
- MSD API: Set MSD data and Request MSD sending to PSAP.

All the APIs are protected from re-entrance related problems.

Voice call is required for correct execution of Control APIs and MSD APIs. Note that none of the eCall In-Band Modem Library APIs is conflicting with the ongoing voice call.

3.2.1. eCall In-Band Modem Library APIs vs. Voice Call State

Table 3. eCall In-Band Modem Library APIs vs. Voice Call

API	Voice Call Requirement	Remarks
inbm_Init	Not required	
inbm_Exit	Not required	
inbm_GetState	Not required	
inbm_GetVersion	Not required	
inbm_OpenChannel	Not required	Inbm_OpenChannel can be invoked before voice call has been triggered or established. But this API will start effective processing the incoming PCM data and sending MSD (if requested for) only once call has been established.
inbm_CloseChannel	Not required	This API can be called after voice call has been terminated (or even before voice call has been established).
inbm_GetChannelState	Not required	
inbm_SetMSD	Not required	This API updates MSD data for an already opened eCall In-Band Modem channel. It does not require however a call to be established.
inbm_SendMSDReq	Required	Request for MSD transmission to PSAP server can only be sent if voice call is active and an eCall In-Band Modem channel is active.
inbm_SetOpts	Not required	This API must be called after INBM Library is initialized. To set the In-Band HLAP timers with user configured timer values.
inbm_GetOpts	Not required	This API must be called after INBM Library is initialized. To retrieve the In-Band HLAP timer values.

3.3. State Machine

3.3.1. eCall In-Band Modem Library State Machine

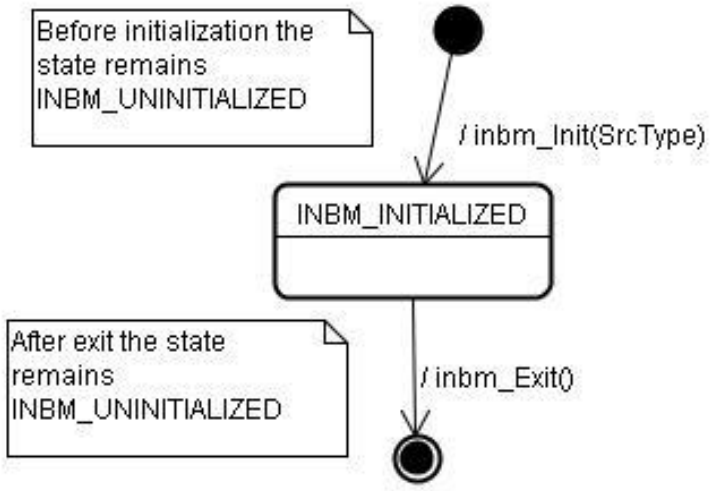

Figure 7. eCall In-Band Modem Library State Machine

3.3.2. eCall In-Band Modem Channel State Machine

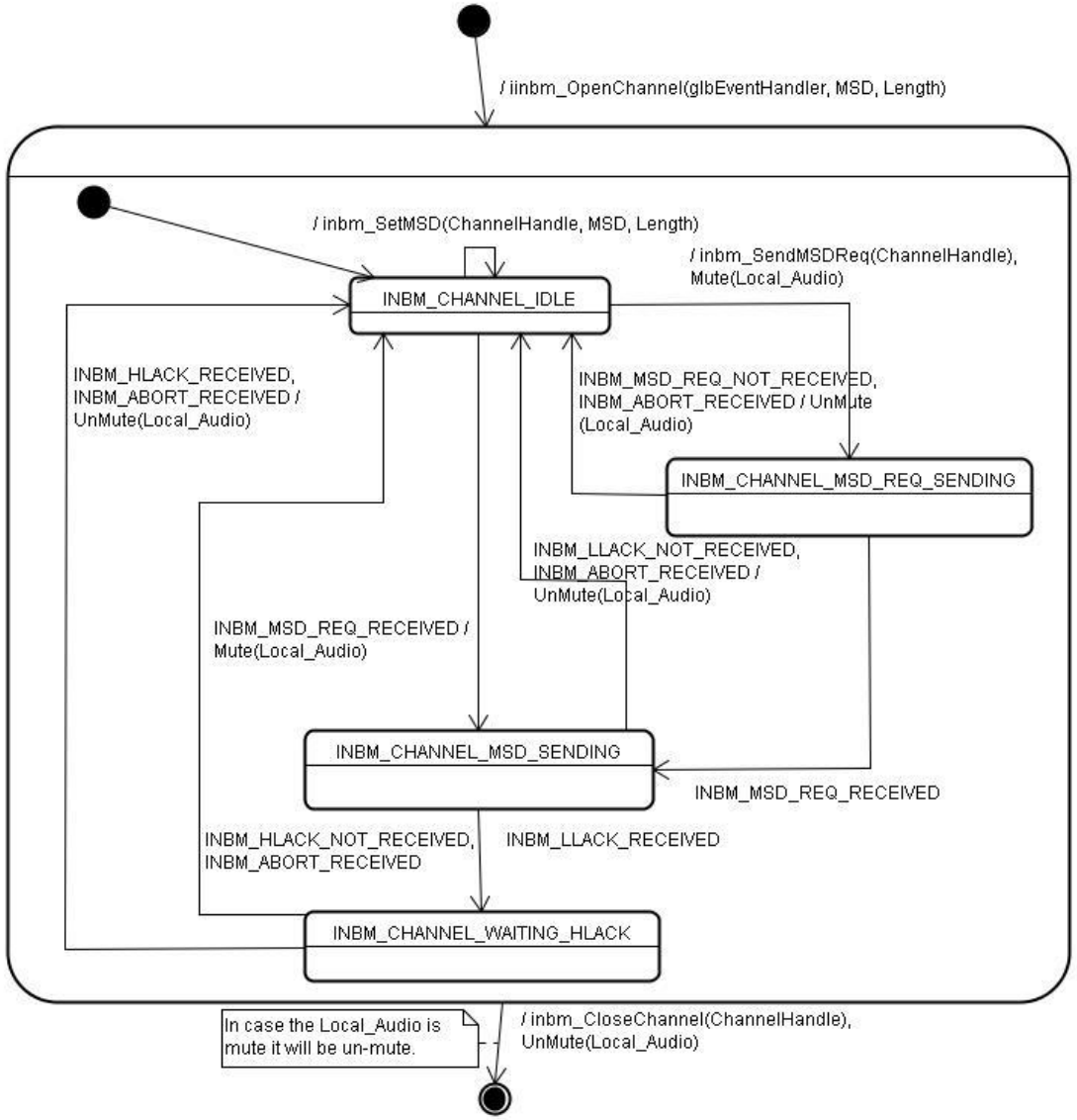

Figure 8. eCall In-Band Modem Channel State Machine

3.4. Handling PSAP Messages

3.4.1. IVS Modem wait mechanism for MSD Request in Push Mode

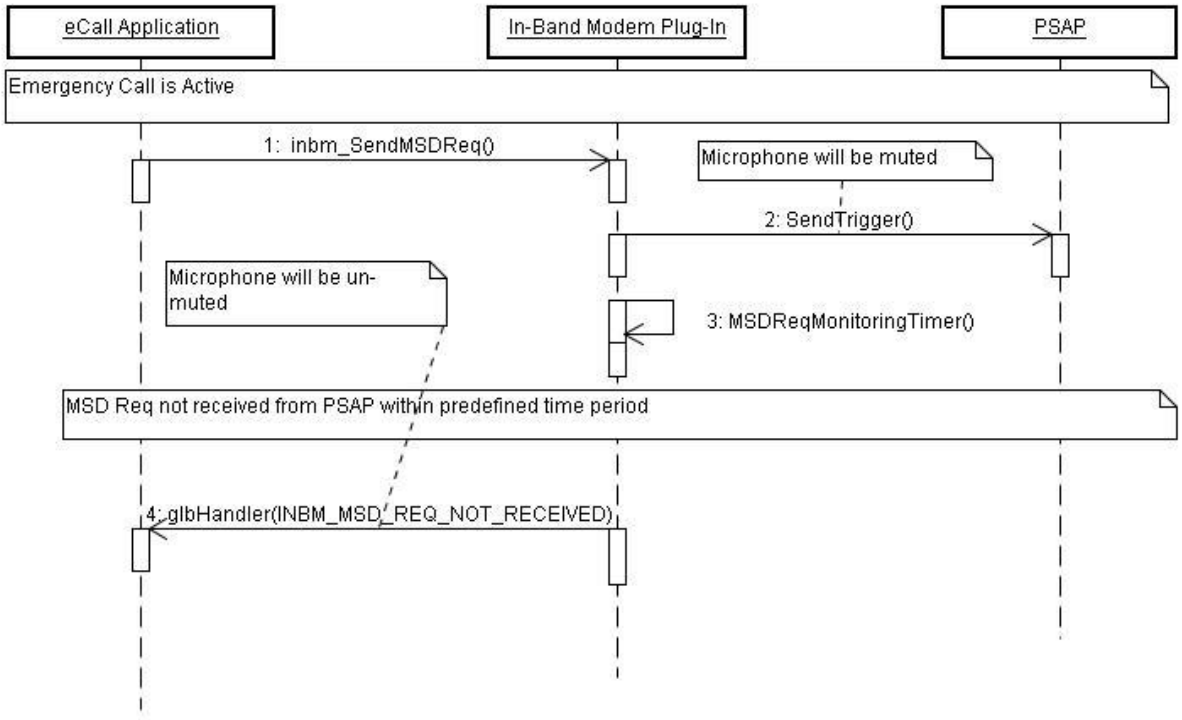

Figure 9. IVS Modem Wait Mechanism for MSD Request in Push Mode, Timeout Case

Note: After INBM_MSD_REQ_NOT_RECEIVED event is received and as long as channel state is INBM_CHANNEL_IDLE, "eCall" Application is allowed to update the MSD information using the "inbm_SetMSD" API.

Once timeout occurs for MSD_Req, eCall In-Band Modem Library doesn't resend a request to PSAP server for MSD_Req (i.e. PUSH Mode).

In successful scenario, PSAP server sends the MSD_Req within the predefined time period and on reception of the same, IVS transmits the MSD data.

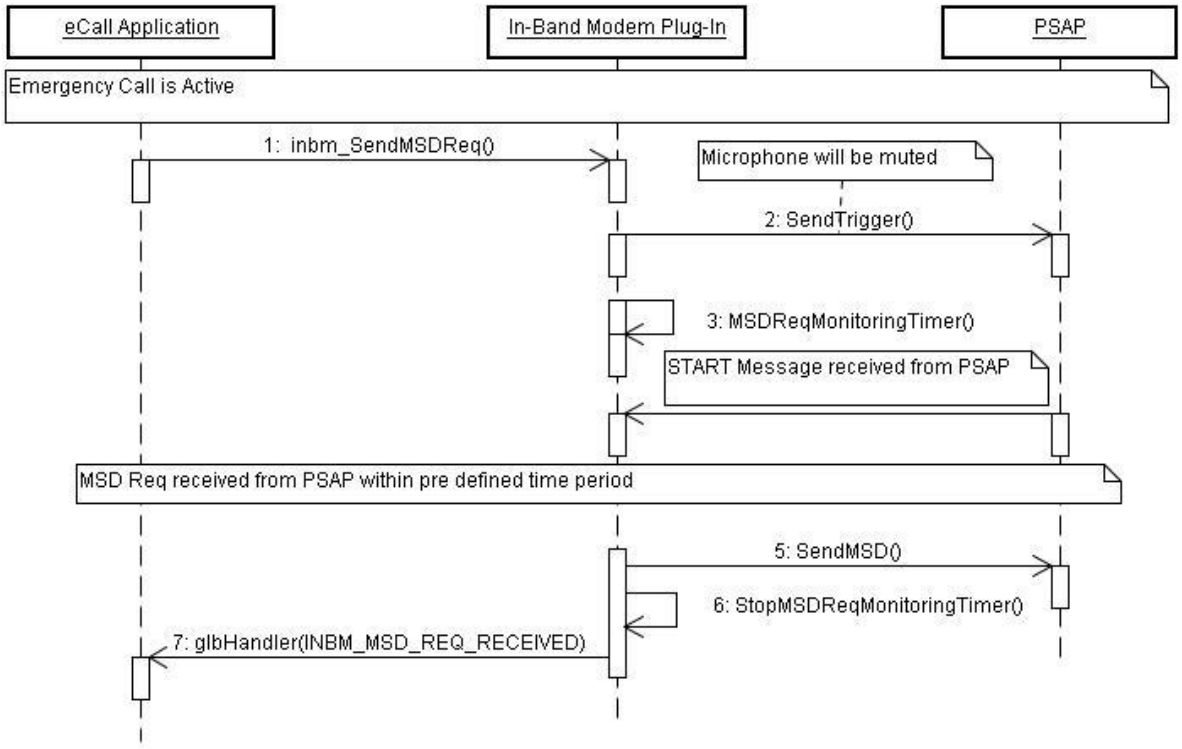

Figure 10. Successful PUSH Mode Scenario

See the HLAP PSAP Messages Waiting Timer Values section for details about the time period of receiving the MSD_Req from PSAP server.

3.4.2. IVS Modem wait mechanism for LLACK

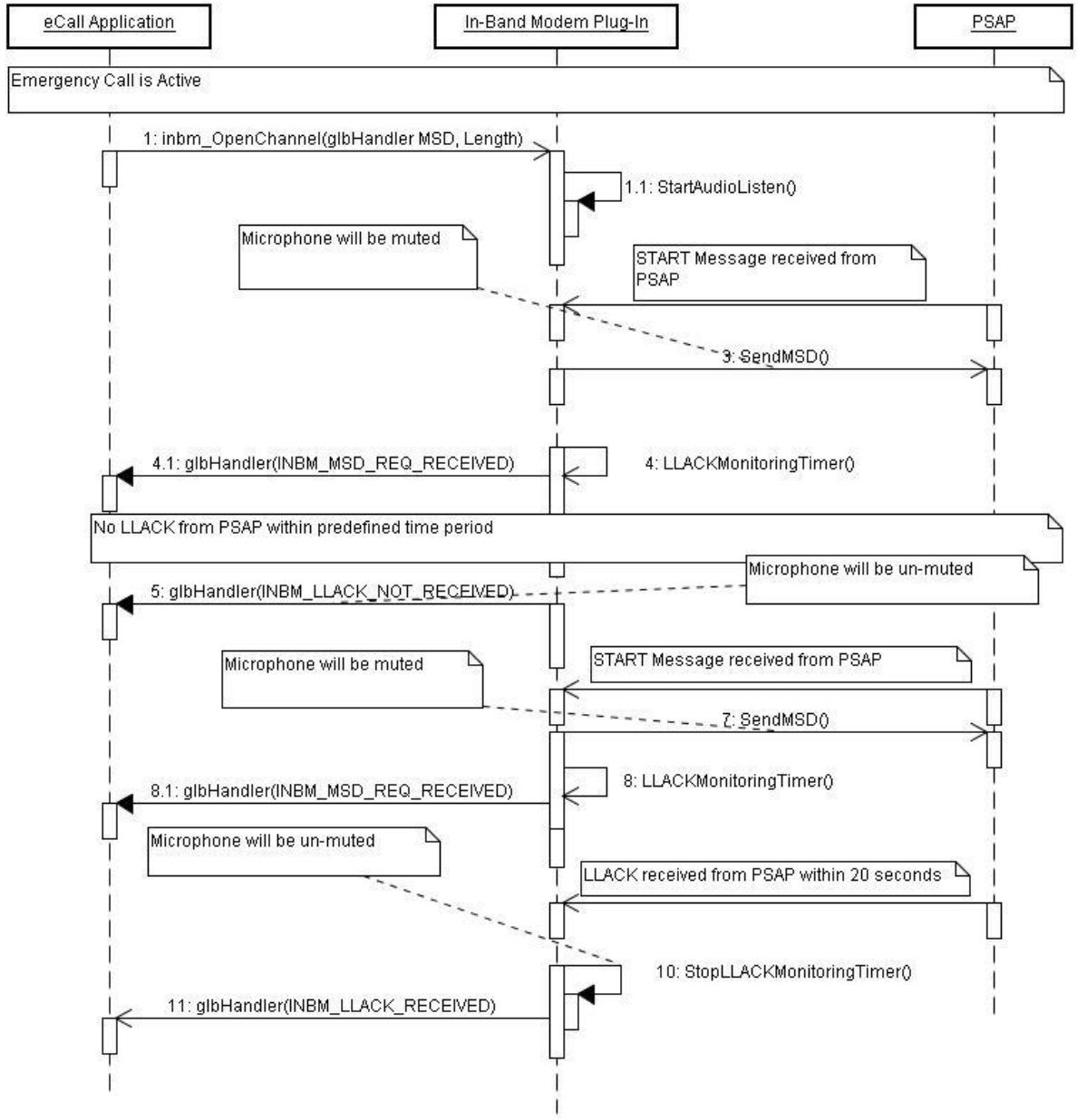

Figure 11. IVS Modem Wait Mechanism for LLACK

Note: After LLACK_NOT_RECEIVED event is received and as long as channel state is INBM_CHANNEL_IDLE, "eCall" Application is allowed to update the MSD information using the "inbm_SetMSD" API.

See the HLAP PSAP Messages Waiting Timer Values section for details about the time period of receiving the LLACK from PSAP server.

3.4.3. IVS Modem wait mechanism for HLACK

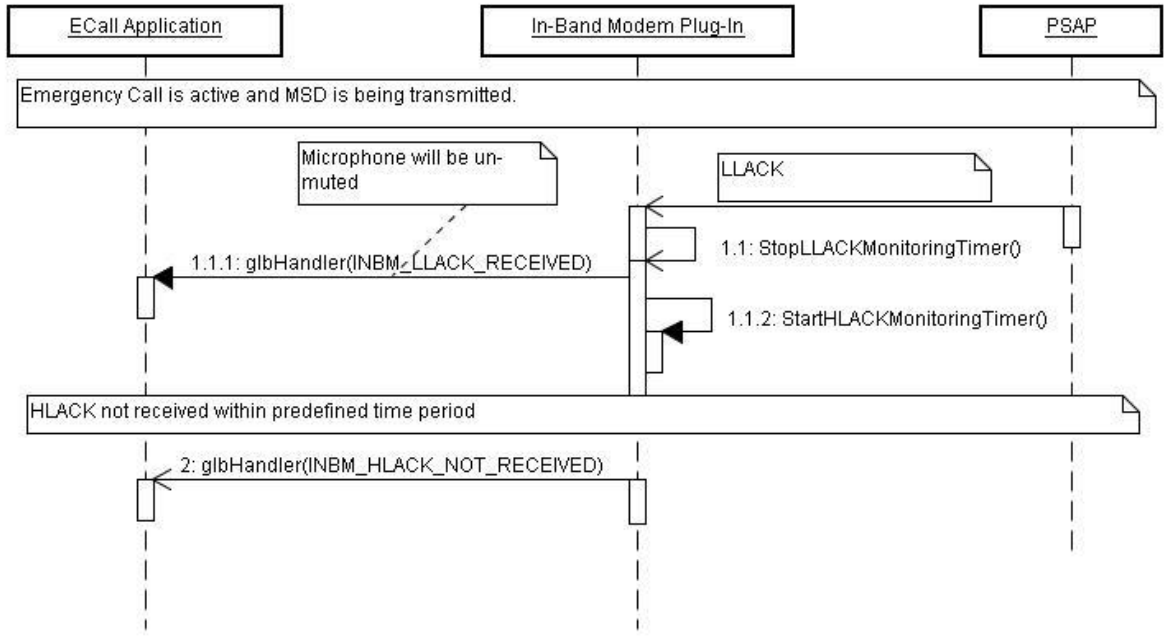

Figure 12. IVS Modem Wait Mechanism for HLACK

Note: After HLACK_NOT_RECEIVED event is received and as long as channel state is INBM_CHANNEL_IDLE, "eCall" Application is allowed to update the MSD information using the "inbm_SetMSD" API.

In successful scenario, PSAP server sends the HLACK within the predefined time period and on reception of the same, MSD transmission with the PSAP server completes successfully.

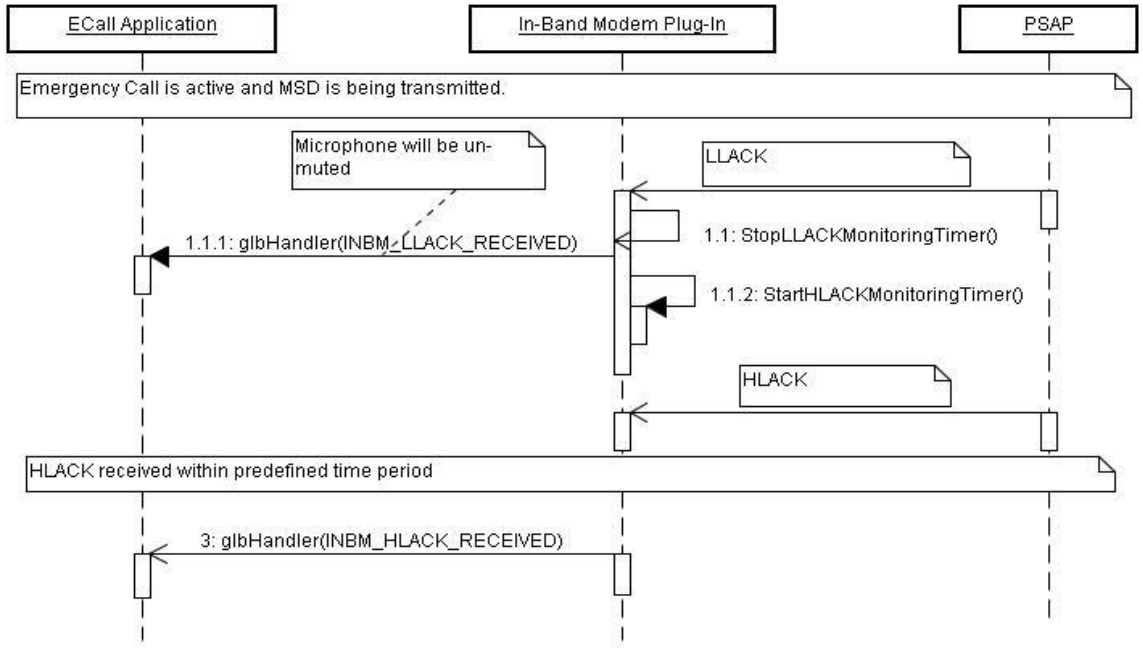

Figure 13. Successful HLACK Scenario

See the HLAP PSAP Messages Waiting Timer Values section for details about the time period of receiving the HLACK from PSAP server

3.4.4. HLAP PSAP Messages Waiting Timer Values

The HLAP timers T2, T3, T5,T6 & T7 are configurable and values can be different from CEN specified values. This is achieved by using inbm_SetOpts API. So behavior of eCall In-Band Modem Library might change, if specified timer values are other than CEN specified values.

Default values are set for these timers which is equal to timer values as in CEN specifications. Refer to: "w15-0310 prEN 16062 Post ENQ CR 100929 v8 FINAL CA.pdf" , "Annex A" 47th page

Table 4. HLAP PSAP Messages Waiting Timer Values

Timer Type	Default Timer Duration (in seconds)
IVS Call Clear-down Fallback Timer (T2)	3600
IVS INITIATION signal duration (T3)	2
IVS Modem wait mechanism for MSD Request in Push Mode (T5)	5
IVS Modem wait mechanism for LLACK (T7)	20
IVS Modem wait mechanism for HLACK (T6)	5

3.5. Resource Usage

3.5.1. Stack Size and Memory

The eCall In-Band Modem Library internally uses both Low Level IRQ Handler and High Level IRQ Handlers. Hence stack for both Low Level IRQ Handler and High Level IRQ Handlers must be declared inside the “eCall” Application. The Library memory footprint depends on the tool chain used, the Library release. Please refer to the eCall In-Band Modem Library Customer Release Note for all memory footprint figures. The eCall In-Band Modem Library doesn’t use Flash or A&D memory, so ROM memory is not used by the Library.

3.5.2. OS Resources

3.5.2.1. OS Timers

A total of three OS timers are used across the eCall In-Band Modem Library co incorporate the HLAP requirements. But at any given point of time, only one times remains active.

3.5.2.2. Synchronization

The semaphore service is used to protect eCall In-Band Modem Library APIs from re-entrance problem. To do so, one binary semaphore is used. The binary semaphore is initialized during Library initialization and used subsequently. When the “inbm_Exit” API is called, this binary semaphore is released.

Hence, all the eCall In-Band Modem Library APIs are blocking.

3.5.2.3. Inter-process communication

The Library internally uses the OS message service with the message ID 0xFFFFFFF2. Hence, the message ID must not be used by the “eCall” application and are reserved.

3.5.3. Embedded Module Load

As the incoming PCM data rate is at a rate of 20ms and for internal processing, during MSD transmission, the Library requires the AirPrime embedded module to switch to boost processing mode (104 MHz.).

Note:	*It is the responsibility of the “eCall” Application to switch to boost mode as soon as the “inbm_OpenChannel” API is called. The “eCall” Application can return to normal mode processing once the “inbm_CloseChannel” API is executed.*

3.5.4. Hardware Resource

None

4. Library Control APIs

The eCall In-Band Modem Library must be initialized by the application before using any other functionality of the same. During initialization, internal parameters (i.e. subscription to audio service and interrupt service etc.) of the Library get initialized.

4.1. Required Header File

The header file for the eCall In-Band Modem Library initialization is inbm_itf.h.

4.2. The inbm_Init API

This API would carry out the initial configuration of the eCall In-Band Modem Library. The necessary initial data-structures that need to be created and/or initialized will also be carried out in this API.

Initialization of the eCall In-Band Modem Library must be carried out by the application prior to making use of the same. The destination audio resource GSM Voice Call (Rx/Tx channel) needs to be specified.

4.2.1. Prototype

```
s8 inbm_Init ( inbm_SrcType_e SrcType );
```

4.2.2. Parameters

SrcType:

In: The destination audio resource upon which the eCall In-Band Modem channel will be created.

```
typedef enum _inbm_SrcType_e
{
    INBM_GSM,              /* Use GSM voice call audio */
    INBM_RESOURCE_LAST
}inbm_SrcType_e;
```

4.2.3. Returned Values

This API returns

- INBM_OK on success
- On error, negative error codes as defined in the table below.

Table 5. Error Codes During Initialization

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library already initialized
INBM_ERR_INVOPT	Invalid audio source type specified
INBM_ERR_INIT	Error in internal resource initialization
INBM_ERR_SEMAPHORE	Error in semaphore consumption
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Called from a Low Level IRQ Handler

4.3. The inbm_Exit API

This API stops the functioning of the eCall In-Band Modem Library. The initial data-structures that are created and/or initialized during inbm_Init() will be released in this API along with any other resources that were acquired.

4.3.1. Prototype

```
s8 inbm_Exit ( void );
```

4.3.2. Parameters

None

4.3.3. Returned Values

This API returns

- INBM_OK on success
- On error, negative error codes as described below:

Table 6. Error Codes During Exit

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized
INBM_ERR_CHANNEL_STILL_OPENED	eCall In-Band Modem channel is not closed
INBM_ERR_EXIT	Error in internal resource release
INBM_ERR_SEMAPHORE	Error in semaphore consumption
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Called from a Low Level IRQ Handler

4.4. The inbm_GetState API

This API retrieves the current state of the eCall In-Band Modem Library.

4.4.1. Prototype

```
void inbm_GetState ( inbm_State_e* State );
```

4.4.2. Parameters

State:

Out: The state of the eCall In-Band Modem Library

4.4.3. The inbm_State_e enumeration

This enumeration holds the different states of the eCall In-Band Modem Library.

```
typedef enum _inbm_State_e
{
        INBM_UNINITIALIZED,
        INBM_INITIALIZED,
        INBM_STATE_LAST
} inbm_State_e;
```

4.4.3.1. eCall In-Band Modem Library State Description

Table 7. eCall In-Band Modem Library States

Events	Description
INBM_UNINITIALIZED	eCall In-Band Modem Library not initialized
INBM_INITIALIZED	eCall In-Band Modem Library initialized
INBM_STATE_LAST	Delimiter of state enumeration

4.4.4. Returned Values

None

4.5. The inbm_GetVersion API

This API returns the current library version.

4.5.1. Prototype

```
const ascii* inbm_GetVersion ( void );
```

4.5.2. Parameters

None

4.5.3. Returned Values

This API returns the string containing the library version on success.

4.6. The inbm_SetOpts API

This API interface sets the HLAP timers of the INBM Library.

- The default values can be changed with s8 inbm_SetOpts(…) API. This API sets the configuration parameters of the INBM Library HLAP timers and is invoked by passing the parameter-option(s) whose value(s) need to be changed. Refer to the error codes table about any errors returned by the interface.
- The list of option names must each be followed by option values. This list must be terminated by INBM_OPT_END.
- If the number of option-value pair parameters specified is >=1 and <5, then only the specified number of timer values are set, and the previously set timer values are retained for the unspecified timer values.
- If the number of option-value pair parameters specified is <1 or >5 then the existing timer values are retained and ‘‘ERROR’’ string is returned.
- If none of the option-value pair parameters are passed along with the command, the existing timer values are retained and the INBM_ERR_INVOPT error is returned to indicate no option-value pair parameters were passed.
- This function checks if the specified timer values are numeric and returns the INBM_ERR_INVOPT error if non-numeric values are specified.
- This function checks that the specified values are with in min/max range of 1 to 3600 seconds. If the values are valid then the timer values are set accordingly. If any of the timer values are invalid i.e., <1 or >3600, then the INBM_ERR_INVOPT error is returned.
- Default timer values are retained if the IVS core is initialized.

Conditions under which the timer values are initialized or reset:

- All timer values are set to default values when the IVS core is initialized, i.e. AT+INBMINIT=0.
- Execution of the command without initializing the IVS core has no effect on the timer values and existing timer values are retained.
- Timer values are not reset when the channel is opened or closed.

Conditions under which the timer values are set to default or existing values retained:

- EXIT followed by INIT:
  In this scenario the default timer values are considered while subscribing for appropriate timers after the IVS core initialization.

- Channel Open & Close & Open:
  In this scenario the timer values that are already set are considered while subscribing for appropriate timers.

4.6.1. Prototype

```
s8 inbm_SetOpts (inbm_Opt_e option1, value1,
                 [inbm_Opt_e option2, value2],…);
```

4.6.2. Parameters

option1 (option2, …):

In: The options for configuring the INBM Library HLAP timers. inbm_SetOpts API supports several parameters. These parameters are a list of options.

value1 (value2, …):

In: The corresponding values to set for the options.

4.6.3. Returned Values

The function returns:

- INBM_OK on success
- In case of an error, an error code as described in the table below.

Table 8. Error Codes for SetOpts

Error Code	Description
INBM_ERR_INVOPT	Invalid option specified for command line parameters
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized

4.6.4. Timer Value Ranges

Table 9. Valid Range of HLAP Timer Values

Timer	Minimum (in seconds)	Maximum (in seconds)
T2	1	3600
T3	1	3600
T5	1	3600
T6	1	3600
T7	1	3600

4.6.5. Impact on Library Behavior

If a timer is already subscribed and MSD transmission in progress, then upon execution of the timer value configure command, the behavior of the Library will not change at that instant, but the timer values will be considered for the next iteration.

Existing Library behavior with CEN-specified timer values may change, based on user-specified timer values as described below:

- If default values (3600, 2, 5, 5, 20 as per CEN specifications) are set for T2, T3, T5, T6, and T7 timers, and then T5 timer will be active for 2 sec waiting for "MSD_REQ_RECEIVED", T7 timer will be active for 20 sec waiting for "LLACK_RECEIVED" and T6 timer will be active for 5 sec waiting for "HLACK_RECEIVED".
- If the user specifies different values (3600, 2, 1, 1, 1) other than CEN specified timer values, then T5 timer will be active for 1 sec waiting for "MSD_REQ_RECEIVED", T7 timer will be active for 1 sec waiting for "LLACK_RECEIVED" and T6 timer will be active for 1 sec waiting for "HLACK_RECEIVED". With these timer values, the eCall In-Band Modem Library behavior may change due to change in timer values.

4.7. The inbm_GetOpts API

This API retrieves the specified configuration parameters of the INBM Library HLAP timers.

4.7.1. Prototype

```
s8 inbm_GetOpts (inbm_Opt_e option1, &value1,
                 [inbm_Opt_e option2, &value2],…);
```

4.7.2. Parameters

option1 (option2, …):

In: The configuration options whose values need to be retrieved. inbm_GetOpts API supports several parameters. These parameters are a list of options.

value1 (value2, …):

Out: Variables to get the retrieved values.

4.7.3. Returned Values

This function returns the following:

- INBM_OK on success.
- In case of an error, an error code as described below is returned

Table 10. Error Codes for GetOpts

Error Code	Description
INBM_ERR_INVOPT	Invalid option specified for command line parameters
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized

4.7.4. The inbm_Opt_e enumeration

This enumeration represents the list of valid options supported by inbm_SetOpts() and inbm_GetOpts() APIs. The option values are specifically set to 2, 3, 5, 6, and 7 because the enumeration values are also used as part of the display message on the console in "inbm_SetOpts".

```
typedef enum _inbm_Opt_e
{
        INBM_CALL_CLEARDOWN_FALLBACK_TIMER = 2,
        INBM_INITIATION_SIGNAL_DURATION = 3,
        INBM_WAIT_FOR_SEND_MSD_PERIOD = 5,
        INBM_WAIT_AL_ACK_PERIOD = 6,
        INBM_MSD_MAX_TX_TIME = 7,
        INBM_WAIT_UNMUTE_AFTER_AL_ACK = 8
} inbm_Opt_e
```

4.7.4.1. INBM End of Optional Parameters to inbm_SetOpts and inbm_GetOpts

Use the macro INBM_OPT_END as the last parameter to inbm_SetOpts and inbm_GetOpts function to indicate the end of optional parameters passed to these functions.

4.7.4.2. INBM hlaptimer Supported Options Description

Table 11. List OF INBM SetOpts & GetOpts API Options

Options	Description
INBM_CALL_CLEARDOWN_FALLBACK_TIMER	Option to configure T2 timer
INBM_INITIATION_SIGNAL_DURATION	Option to configure T3 timer
INBM_WAIT_FOR_SEND_MSD_PERIOD	Option to configure T5 timer
INBM_WAIT_AL_ACK_PERIOD	Option to configure T6 timer
INBM_MSD_MAX_TX_TIME	Option to configure T7 timer
INBM_WAIT_UNMUTE_AFTER_AL_ACK	Option to delay unmute and AL ACK in steps of 100 ms seconds. Default value is 20, 20*100ms = 2 seconds.

5. Channel Control APIs

The Channel Control APIs allow the control of the eCall In-Band Modem channel.

5.1. The inbm_OpenChannel API

This API opens the eCall In-Band Modem channel. The associated event handler call back function needs to be specified with this API which will capture all the events related to the eCall In-Band Modem channel.

The MSD is taken as binary. The length of the MSD data must be less than or equal to 140 bytes. If less than 140 bytes is provided then the padding with '0' for the remaining bytes is internally handled by the eCall In-Band Modem Library.

5.1.1. Prototype

```
s8 inbm_OpenChannel ( inbm_glbEvtHandler_t glbEvtHandler, u8 *MSD, u8 Length );
```

5.1.2. Parameters

glbEvtHandler:

Glb: The global call back function which would be invoked to notify the InBand channel related events. Please note that the global callback handler will be executed in the same task context from which inbm_OpenChannel() API is called

MSD:

In: MSD data as defined in [2].

Length:

In: Length of the MSD data.

5.1.3. Returned Values

This API returns

- Channel handle on success
- On error, negative error codes as described below:

Table 12. Error Codes During Channel Creation

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized
INBM_ERR_MAX_CHANNEL_ALREADY_OPENED	The eCall In-Band Modem channel cannot be opened
INBM_ERR_INVOPT	Invalid option value

Error Code	Description
INBM_ERR_AUDIO	Error in audio stream listen
INBM_ERR_SEMAPHORE	Error in semaphore consumption
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Called from a Low Level IRQ Handler

5.2. Handling Channel Events

The global call back handler function will be called to notify the application about the eCall In-Band Modem channel related events.

5.2.1. Prototype

```
typedef void ( *inbm_glbEvtHandler_t ) ( inbm_Event_e event );
```

5.2.2. Parameters

event:

In: The event which triggers the call back function.

5.2.3. The inbm_Event_e enumeration

This enumeration holds the different events that notify of completion of certain actions and/or trigger further actions.

```
typedef enum _inbm_Event_e
{
        INBM_MSD_REQ_RECEIVED,
        INBM_MSD_REQ_NOT_RECEIVED,
        INBM_LLACK_RECEIVED,
        INBM_HLACK_RECEIVED,
        INBM_LLACK_NOT_RECEIVED,
        INBM_HLACK_NOT_RECEIVED,
        INBM_ABORT_RECEIVED,
        INBM_HLACK_RECEIVED_CLEARDOWN,
        INBM_EVENT_LAST
}inbm_Event_e;
```

5.2.3.1. eCall In-Band Modem Channel Event Description

Table 13. eCall In-Band Modem Channel Events Description

Events	Description	Remarks
INBM_MSD_REQ_RECEIVED	SEND MSD request received from PSAP server	
INBM_MSD_REQ_NOT_RECEIVED	SEND MSD request not received from PSAP server within 5 seconds from the first SEND MSD REQUEST message sending	This event is generated on expiration of CEN/TC T5 timer. eCall In-Band Modem channel will be reset to IDLE state. MSD data must be updated using the inbm_SetMSD API if further MSD data transmission is required.
INBM_LLACK_RECEIVED	LLACK received from PSAP server for the MSD transmission	
INBM_LLACK_NOT_RECEIVED	LLACK not received within 20 seconds from MSD transmission	This event is generated on expiration of CEN/TC T7 timer. eCall In-Band Modem channel will be reset to IDLE state. MSD data must be updated using the inbm_SetMSD API if further MSD data transmission is required.
INBM_HLACK_RECEIVED	HLACK received from PSAP server for the MSD transmission	eCall In-Band Modem channel will be reset to IDLE state. MSD data must be updated using the inbm_SetMSD API if further MSD data transmission is required.
INBM_HLACK_NOT_RECEIVED	HLACK not received within 5 seconds from the time LLACK received	This event is generated on expiration of CEN/TC T6 timer. eCall In-Band Modem channel will be reset to IDLE state. MSD data must be updated using the inbm_SetMSD API if further MSD data transmission is required.
INBM_ABORT_RECEIVED	Abort during data transmission	This event occurs when an internal IVS modem error has been detected.
INBM_HLACK_RECEIVED_CLEARDOWN	Cleardown indication received from PSAP after successful MSD transfer	This event is sent by PSAP after receiving the MSD successfully and the line will be freed up. On reception of this event at the IVS side, the voice call is terminated.
INBM_EVENT_LAST	Delimiter for the event enumeration	Reserved for internal use.

5.3. The inbm_CloseChannel API

This API closes the already opened eCall In-Band Modem channel.

5.3.1. Prototype

```
s8 inbm_CloseChannel ( s8 ChannelHandle );
```

5.3.2. Parameters

ChannelHandle:

In: Channel handle of already opened eCall In-Band Modem channel.

5.3.3. Returned Values

This API returns

- INBM_OK on success
- On error, negative error codes as described below:

Table 14. Error Codes During Channel Closing

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized
INBM_ERR_UNKNOWN_HANDLE	Unknown eCall In-Band Modem channel handle
INBM_ERR_CHANNEL	Error in closing eCall In-Band Modem channel
INBM_ERR_SEMAPHORE	Error in semaphore consumption
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Called from a Low Level IRQ Handler

5.4. The inbm_GetChannelState API

This API retrieves the current state of an eCall In-Band Modem channel.

5.4.1. Prototype

```
s8 inbm_GetChannelState ( s8 ChannelHandle, inbm_ChState_e* ChState );
```

5.4.2. Parameters

ChannelHandle:

In: Channel handle of already opened eCall In-Band Modem channel.

ChState:

Out: The state of the already created eCall In-Band Modem channel

5.4.3. The inbm_ChState_e enumeration

This enumeration holds the different states of eCall In-Band Modem channel.

```
typedef enum _inbm_ChState_e
{
        INBM_CHANNEL_IDLE,
        INBM_CHANNEL_MSD_REQ_SENDING,
        INBM_CHANNEL_MSD_SENDING,
        INBM_CHANNEL_WAITING_HLACK,
        INBM_CHANNEL_STATE_LAST
} inbm_ChState_e;;
```

5.4.3.1. eCall In-Band Modem Library State Description

Table 15. eCall In-Band Modem Channel States

Events	Description
INBM_CHANNEL_IDLE	eCall In-Band Modem channel opened and idle
INBM_CHANNEL_MSD_REQ_SENDING	Sending MSD SEND REQUEST to PSAP server
INBM_CHANNEL_MSD_SENDING	Sending MSD data to PSAP server
INBM_CHANNEL_WAITING_HLACK	Waiting for HLACK from PSAP server
INBM_CHANNEL_STATE_LAST	Delimiter of state enumeration

5.4.4. Returned Values

This API returns

- INBM_OK on success
- On error, negative error codes as described below:

Table 16. Error Codes for Get State of eCall In-Band Modem Channel

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized
INBM_ERR_UNKNOWN_HANDLE	Invalid eCall In-Band Modem channel handle

6. MSD APIs

The MSD APIs allow settings and sending the eCall MSD.

6.1. The inbm_SendMSDReq API

This API is for IVS to send a SEND_MSD_REQUEST to PSAP server over an already opened eCall In-Band Modem channel. It must be invoked just after inbm_OpenChannel() API for PUSH mode use case.

6.1.1. Prototype

```
s8 inbm_SendMSDReq (s8 ChannelHandle );
```

6.1.2. Parameters

ChannelHandle:

In: Channel handle of already opened eCall In-Band Modem channel.

6.1.3. Returned Values

This API returns

- INBM_OK on success
- On error, negative error codes as described below:

Table 17. Error Codes During Exit

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized
INBM_ERR_BAD_CHANNEL_STATE	Action not allowed in this channel state
INBM_ERR_UNKNOWN_HANDLE	Unknown eCall In-Band Modem channel handle
INBM_ERR_SEMAPHORE	Error in semaphore consumption
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Called from a Low Level IRQ Handler
INBM_ERR_TIMER	Error with MSD_Req monitoring timer subscription

6.2. The inbm_SetMSD API

This API updates the MSD data for any future MSD data sending over already created eCall In-Band Modem channel. It will override any previously specified MSD data by inbm_OpenChannel() or inbm_SetMSD() API call.

The MSD is taken as binary.The length of the MSD data must be less than or equal to 140 bytes. If less than 140 bytes is provided then the padding with '0' for the remaining bytes is internally handled by the eCall In-Band Modem Library.

Note: This API must not be called when the eCall In-Band Modem channel state other than "INBM_CHANNEL_IDLE". This API must not be called on reception of "INBM_MSD_REQ_RECEIVED" event.

6.2.1. Prototype

```
s8 inbm_SetMSD ( s8 ChannelHandle, u8* MSD, u8 Length );
```

6.2.2. Parameters

ChannelHandle: In: Channel handle of already opened eCall In-Band Modem channel.

MSD: In: MSD as defined in [2].

Length: In: Length of the MSD data.

6.2.3. Returned Values

This API returns

- INBM_OK on success
- On error, negative error codes as described below:

Table 18. Error Codes for Sending MSD

Error Code	Description
INBM_ERR_BAD_STATE	eCall In-Band Modem Library not initialized
INBM_ERR_UNKNOWN_HANDLE	Invalid eCall In-Band Modem channel handle
INBM_ERR_CHANNEL_BAD_STATE	Invalid eCall In-Band Modem channel state to set MSD data
INBM_ERR_INVOPT	Invalid data length or NULL MSD data provided
INBM_ERR_SEMAPHORE	Error in semaphore consumption
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Called from a Low Level IRQ Handler

7. API Call Requirements

This section defines the preconditions for an API to be called, based on the different states of the Library as well as of the channels.

7.1. Library Control API Calls Requirements

The following table shows the pre-requisites for calling the Library API. ‘X’ means the API is authorized in the corresponding state. ‘-’ means the API is NOT authorized or will return an error. The Library Control API calls are allowed according to the Library state as shown in the table below.

Table 19. Library Control APIs Calls Requirements

API	INBM_UNINITIALIZED	INBM_INITIALIZED
inbm_Init	X	-
inbm_Exit	-	X
inbm_GetState	X	X
inbm_GetVersion	X	X
inbm_OpenChannel	-	X
inbm_SetOpts	-	X
inbm_GetOpts	-	X
Other Channel related APIs: see also 0	-	X

7.2. Channel and MSD API Calls Requirements

The following table shows the pre-requisites for calling the Library API. ‘X’ means the API is authorized in the corresponding state. ‘-’ means the API is NOT authorized or will return an error.

All Channel and MSD API, except inbm_OpenChannel, require a valid channel to be opened and provided as input parameter. These API calls are allowed according to the specified channel state as shown in the table below.

Table 20. Channel Control APIs and MSD APIs Calls Requirements

API	INBM_CHANNEL_IDLE	INBM_CHANNEL_MSD_REQ_SENDING	INBM_CHANNEL_MSD_SENDING	INBM_CHANNEL_WAITING_HLACK
inbm_CloseChannel	X	X	X	X
inbm_SendMSDReq	X	-	-	-
inbm_SetMSD	X	-	-	-
inbm_GetChannelState	X	X	X	X

8. Error Codes

Table 21. Error Codes

Error Code	Description	Value
INBM_OK	On success	0
INBM_ERR_INIT	Unable to initialize library	-21
INBM_ERR_EXIT	Unable to stop library	-22
INBM_ERR_INVOPT	Invalid option specified for input arguments	-23
INBM_ERR_BAD_STATE	Incorrect state for the operation	-25
INBM_ERR_BAD_CHANNEL_STATE	Incorrect channel state for the operation	-26
INBM_ERR_MAX_CHANNEL_ALREADY_OPENED	No more eCall In-Band Modem channel can be opened	-27
INBM_ERR_AUDIO	Error with audio operation	-28
INBM_ERR_CHANNEL	Error during channel closing	-29
INBM_ERR_CHANNEL_STILL_OPENED	Channel still opened	-30
INBM_ERR_UNKNOWN_HANDLE	Invalid channel handle specified	-31
INBM_ERR_SEMAPHORE	Error with semaphore handling	-32
INBM_ERR_SEMAPHORE_SERVICE_LOCKED	Semaphore operation is called from a LOW IRQ handler	-33
INBM_ERR_TIMER	Error with timer handling	-34

Index

SIERRA
WIRELESS®