LGE-DAB-APP-INTERFACEs

Version 1.3 - 2016-03-31

LGE-DAB-APP-INTERFACEs-V1.3

Mobile Communication Company
Mobile Handset R&D Center
Development MM, 3Team

Copyright

Though every care has been taken to ensure the accuracy of this document, LG Electronics Co, Ltd. cannot accept responsibility for any errors or omissions or for any loss occurred to any person, whether legal or natural, from acting, or refraining from action, as a result of the information contained herein. Information in this document is subject to change at any time without obligation to notify any person of such changes.

LG Electronics Co, Ltd. may have patents or patent pending applications, trademarks copyrights or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give the recipient or reader any license to these patents, trademarks copyrights or other intellectual property rights.

No part of this document may be communicated, distributed, reproduced or transmitted in any form or by any means, electronic or mechanical or otherwise, for any purpose, without the prior written permission of LG Electronics Co, Ltd.

The document is subject to revision without further notice.

All brand names and product names mentioned in this document are trademarks or registered trademarks of their respective owners.

1 Introduction

This chapter describes the purpose, scope, and history of this document.

1.1 Purpose

The purpose of this document is to provide LG DMB SDK for the development of DAB/DMB applications. The DAB/DMB service is provided in the Android platform at the request of EU broadcasting unions (IDAG, EBU and World DMB).

1.2 Scope

This document describes the APIs used in applications for watching and listening to broadcasts in the Android platform.

1.3 Conventions

1.3.1 IN/OUT

The meaning of IN/OUT when describing APIs in this document shall be following:

- [IN]: Input Parameter
- [OUT] : Output Parameter
- [IN/OUT] : Input and output parameter

1.4 Revision history

Version	Date	Comment
1.0	30/11/2015	First distribution of LG DMB SDK document

1.5 References

This document comes with the following file:

Macro File	Description

Caution

Applications are provided in APK file format through the Android platform.

2 Data Type and Structure

This chapter describes the data types and structures used in this document.

2.1 Data Type

This section describes various data types used in this document.

2.1.1 op_mode

The op_mode indicates the mode of DAB/DMB service.

1 : DAB (Radio Service)
2 : DMB (General video Service)
3 : VISUAL_DMB (Visual Radio Service) – Korea only
4 : DATA (Data Service) – MOT only for packet mode SubChannel
5 : TPEG (TPEG Service) – Not Supported
6 : ENSQUERY (Channel Search Service)

2.1.2 DataCallback msg type

0x1001 : DLS
0x1002 : SLS
0x1004 : FIC (Parsed FIC Data After Channel Searching)
0x1010 : DMB Device Signal Information (BER , Antenna level etc)
0x1040 : FIC Raw Data
0x1080 : PAD DataGroup
0x1100 : PACKET DataGroup

2.1.3 EventCallback msg type

0x0801 : DMB_SRV_OK
0x0807: DMB_VISUAL_RADIO_RECEIVED
0x0811: DMB_SRV_INFO_UPDATE
0x0812: DMB_PLAY_READY_SRV
0x0C01: DMB_ERROR_SET_CHANNEL

2.1.4 SignalCallback msg type

0x0100 : DMB_SIG_MSG_TYPE (unlocking)
0x0101: DMB_SIG_LOCK (locking)

2.2 Structure

This section describes various structures used in this document.

2.2.1 DLS Data

DLS Data is sent to the application via DataCallback Msg Type 0x1001.
(DL Plus is not supported)

메모 [u1]: Add sentence.

DLS Data Structure consists of the following components:

```
int16                   DLSCount;
DLS_TYPE_T              DLSType;
API_SI_DLS_LABEL_T      DLS;
uint8                   DLS_COMMAND;
```

DLS_TYPE_T and API_SI_DLS_LABEL_T Structure consist of the following components:

```
typedef enum DLS_TYPE
{
    DLS_LABEL = 0,              /* Label type*/
    DLS_COMMAND                 /* Command type*/
} DLS_TYPE_T;
```

```
typedef struct API_SI_DLS_LABEL
{
    CHAR_SET_T    charSet;
    uint8         segCount;
    uint8         length[MAX_DLS_SEGNUM];
    uint8         dynamicLabel[MAX_DLS_SEGNUM][MAX_SEGMENT_LENGTH];
} API_SI_DLS_LABEL_T;
```

CHAR_SET_T consists of the following components:

```
typedef enum CHAR_SET
{
    COMPLETE_EBU_LATIN = 0x0, /* complete EBU Latin based repertoire */
    COMMON_CORE_EBU_LATIN = 0x1, /* EBU Latin based common core, Cyrillic,
```

```
Greek */
    CORE_EBU_LATIN = 0x2, /* EBU Latin based core, Arabic, Hebrew, Cyrillic and
Greek */
    ISO_LATIN_ALPHABET2 = 0x3, /* ISO Latin Alphabet No 2 */
    ISO_LATIN_ALPHABET1 = 0x4, /* ISO Latin Alphabet No 1 (only for MOT) */
    ISO_10646_1 = 0xF, /* ISO 10646-1 using UTF-8 transformation format (only for
FIC)*/
    KSX1005_1_UNICODE = 0x4, /* KS X 1005-1 (Unicode) */
    KSX1001_WANSUNG = 0x6, /* KS X 1001 (Wansung) */
    UNKNOWN_CHAR_SET = 0xFF
} CHAR_SET_T;
```

2.2.2 SLS Data

SLS Data is sent to the application via DataCallback Msg Type 0x1002.

SLS Data Structure consists of the following components:

```
uint32                  objectId;
uint8                   *pImageBuf;
uint32                  size;
uint16                  width;
uint16                  height;
MOT_CONTENT_SUBTYPE_T   type;
```

MOT_CONTENT_SUBTYPE_T is a set of enum values. Refer to the sample application code for information on most frequently used content types.

2.2.3 PAD DataGroup

PAD DataGroup is sent to the application as raw data. Therefore, detailed description on the PAD DataGroup is not discussed here.

2.2.4 PACKET DataGroup

Packet DataGroup is sent to the application via DataCallback Msg Type 0x1100.

Packet DataGroup Structure consists of the following components:

```
uint16      packet_address;
uint8       buffer[MAX_PKT_BUF];
```

```
uint16          length;
```

메모 [u2]: Add section(2.2.5)

2.2.5 Channel DB

Channel DB is sent to the application via DataCallback Msg Type 0x1004.

Channel DB Structure consists of the following components:

```
int     op_mode;                    // Operation mode   1 : DAB, 2 : DMB
int     history;                    // Not used
int     preset;                     // Not Used
int     freq_no;                    // Frequence Block No. (Ensemble Id)
int     eid;                        // Not Used
int     sid;                        // Service Id
int     subchid;                    // SubChannel Id
int     s_pd;                       // Not Used
int     s_ps;                       // Not Used
int     s_tmid;                     // Not Used
int     ch_num_ui;                  // Channel Number ( in UI )
int     el_flag;                    // Not Used
int     sl_flag;                    // Not Used
int     scl_flag;                   // Not Used
int     scid;                       // Not Used
int     DG_flag;                    // Not Used
int     DSCTy;                      // data (or audio) service component type
int     userapptype;                // Not Used
int     pack_addr;                  // Not Used
int     storeType;                  // Not Used
int     sync_pid;                   // Not Used
int     start_addr;                 // Not Used
int     table_index;                // Not Used
int     userappdata;                // Not Used
int     protection_level;           // Not Used
int     subch_size;                 // Not Used
char el[TDMB_CH_MAX_LABEL_LEN];     // Ensemble Label
char sl[TDMB_CH_MAX_LABEL_LEN];     // Service Label
```

```
char scl[TDMB_CH_MAX_LABEL_LEN];        // Service component Label
int     charset;                        // charset
int     fec_scheme;                     // 0 : no fec scheme , 1 : fec scheme
```

메모 [u3]: Add section(2.2.6)

2.2.6 Signal Information

Signal Information is sent to the application via DataCallback Msg Type 0x1010.

Signal Information Structure consists of the following components:

```
uint32 dab_ok;                  // 1 : Broadcast signal is OK, 0: Broadcast Signal is NOK
uint32 msc_ber;                 // msc ber value (0 ~ 20000)
uint32 sync_lock;               // 1 : Base Band is locked, 0 : Base Band is unlocked
uint32 afc_ok;                  // not used
uint32 cir;                     // not used
uint32 fic_ber;                 // not used
uint32 tp_lock;                 // not used
uint32 sch_ber;                 // not used
uint32 tp_err_cnt;              // not used
uint32 va_ber;                  // not used
uint32 srv_state_flag;          // not used
uint32 antenna_level;           // antenna level for display on indicator bar (0 ~ 4)
int8    rssi;                   // rssi
```

3 APP Interfaces

This chapter describes the basic form and functions of the APIs for DAB/DMB services.

3.1 Operation

In applications, the APIs provided by LG DMB SDK can be used to control the device and DAB/DMB framework as well as the channel search and broadcast watching and listening functions.

3.2 Dmb Class Basic methods

This section describes the methods for the Dmb Class necessary for DAB/DMB service initialization and shutdown, channel search, and broadcast watching and listening.

3.2.1 Dmb.open

```
public static Dmb open( )
```

The Dmb class instance is created, and the DAB/DMB service is connected. release() shall be used to release the resource in order to close the DAB/DMB service after creating Dmb class instance.

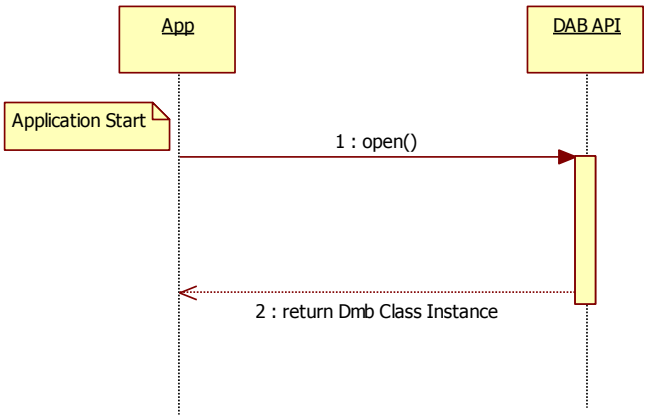
Parameter

None

Return value

Dmb: Dmb class instance

Call Flow



3.2.2 Dmb.release

```
public static void release( )
```

The DAB/DMB service connection is ended, and the resource is released. release() and open() shall be used in pair and perform directly opposed operations.

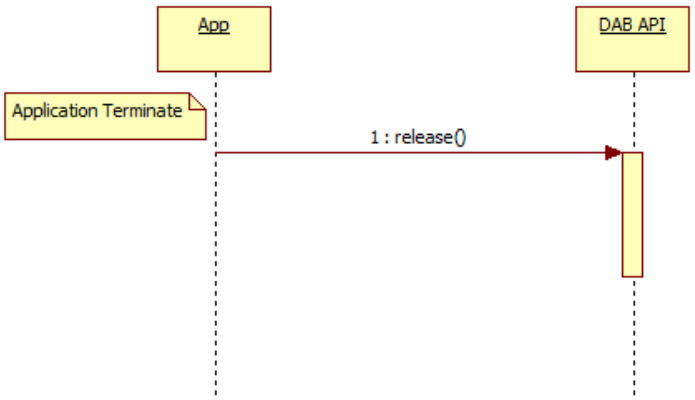
Parameter

None

Return value

None

Call Flow



3.2.3 Dmb.init

```
public final void init
(
int op_mode,
int live_flag,
int nation_info,
EventCallback eventCB
)
```

The device is opened and turned on to provide the DAB/DMB services. The framework is initialized, and op_mode is set.

Parameter

op_mode: DAB/DMB service mode (See 2.1.1. op_mode)

live_flag: 1 is on-air, 0 is local

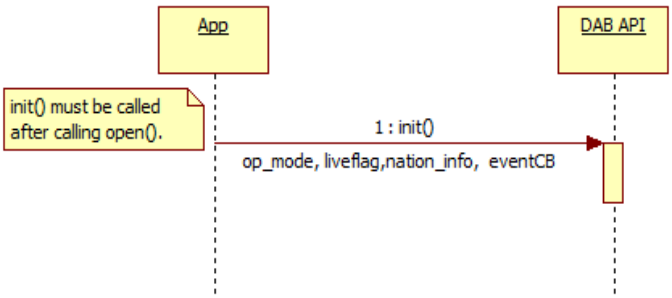
nation_info : 1 is EU DAB, 0 is Korea TDMB

eventCB: Callback function to receive events that occur in the DAB/DMB service

Return value

None.

Call Flow



3.2.4 Dmb.exit

```
public final void exit
(
void
 )
```

The device is turned off and the framework is released to end the active DAB/DMB service. exit() and init() shall be used in pair and perform directly opposed operations.

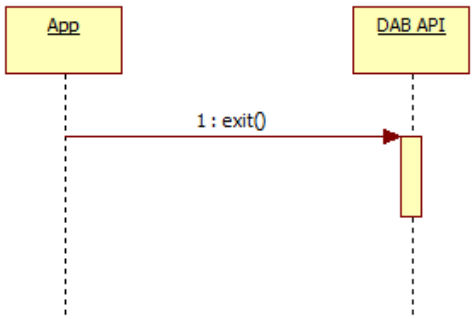
Parameter

None

Return value

None

Call Flow



3.2.5 Dmb.find

```
public final void find
( int freq, SignalCallback sigCB )
```

Searching for channels is executed.

Parameter

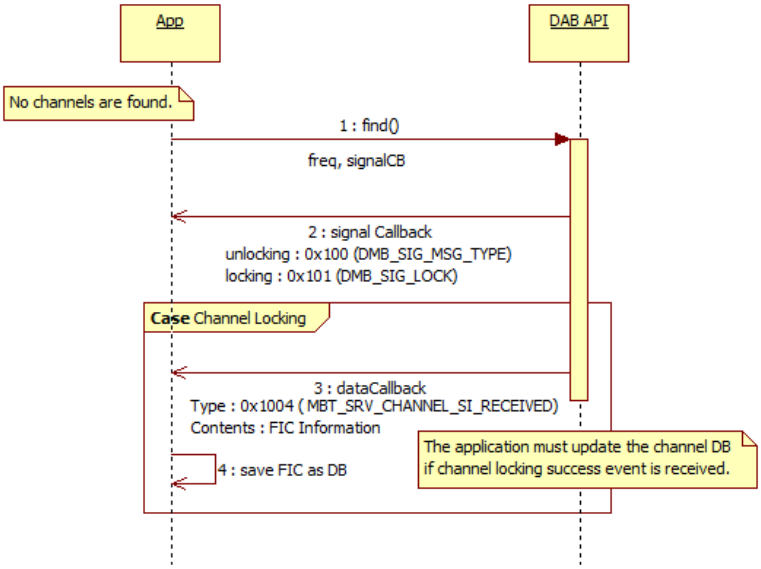
freq : ensemble block number (e.g 7A →71, 7B→72, 7C→73, 13A→131, 13B→132 etc)

sigCB : SignalCallback function to inform the search result whether the channel is locked or not

Return value

None

Call Flow



Note

The existence of channels shall be examined in case of searching channels. SignalCallback function informs whether the channel is locked or not(unlocking: 0x100, locking:0x101) to the application.

If the channel is locked, the application registers a callback function to receive FIC data of the channel from Dmb.data(). Then, the framework sends FIC data of the channel to the application by using the registered callback function.

Next, the application parses the FIC data(See 2.2.5 Channel DB) received and saves the channel information in the DB.

3.2.6 Dmb.select

```
public final void select
( int op_mode, int freq, int channelID, int xSCTy, int subChannelSize, int fecScheme,
VideoCallback videoCB, AudioCallback audioCB)
```

The desired channels are set to watch or listen to broadcasts provided by the DAB/DMB service. And the mode for receiving reconfiguration information is set.

메모 [u4]: Add sentence.

Parameter

op_mode: DAB/DMB service mode (See 2.1.1. op_mode)

freq: ensemble block number (e.g., 5A →71, 7B→72, 13A→131, 13B→132, etc.)

channel_ID: subchannel ID of the service to be set

xSCTy: service component type of the current channel

subChannelSize: subchannel size of the current channel

fecScheme: Whether FEC (Forward Error Corruption) applies to the subchannel or not

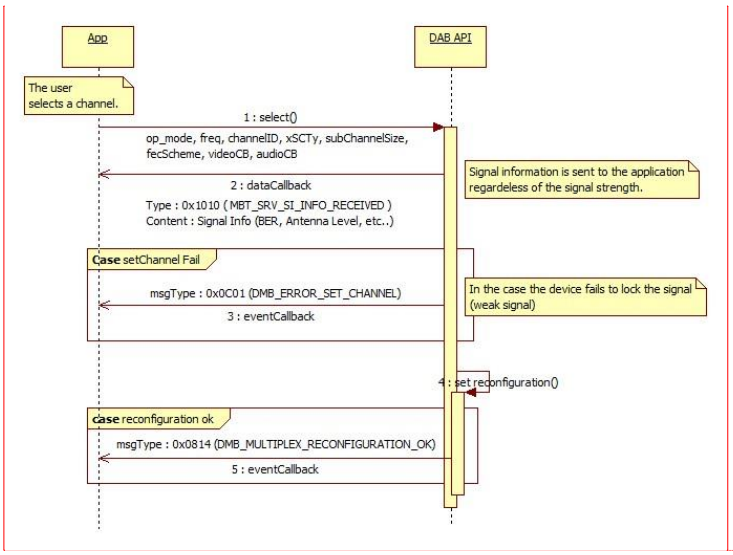
videoCB : Not used

audioCB : Not used

Return value

None

Call Flow



메모 [u5]: Modify UML

Note

The Signal Information includes the information related to signal strength (See 2.2.6 Signal Information).

The application parsed the signal information received from datacallback (type : 0x1010).

메모 **[u6]:** Add Note

3.2.7 Dmb.data

```
public final void data
(
int datatype,
int onOff,
DataCallback DataCB
)
```

This is the callback function that receives data to be sent from the framework to the application. The DataCallback function is registered to receive FIC (Fast Information Channel), SLS/DLS, etc.

Parameter

datatype: Data type to be received (See 2.1.2 dataCallback msg type)

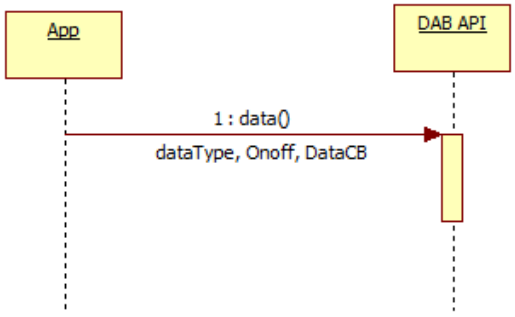
onOff: Whether data is sent or not

DataCB: Callback function to receive data

Return value

None.

Call Flow



3.2.8 Dmb.SetvideoDimension

```
public final void select
( int x, int y, int w, int h, int r, SurfaceHolder sh )
```

The size, position, and rotation of the video displayed on the device are set.

Parameter

x: x coordinate of the video

y: y coordinate of the video

w: Width of the video

h : Height of the video

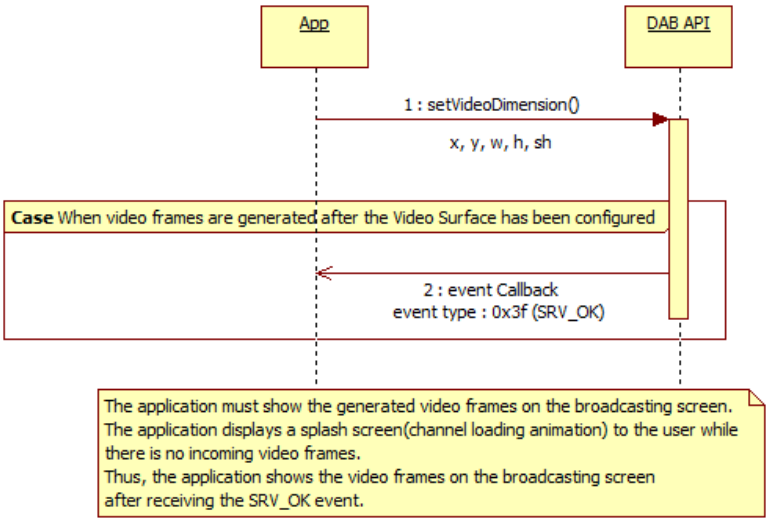
r: Rotation

sh: Surface for the video

Return value

None

Call Flow



3.2.9 SignalCallback

```
public interface SignalCallback
{
 void onSignal(int lock, Dmb mDmb);
}
```

This is a callback interface to indicate the locking/unlocking status for channel search. This must be implemented in case a SignalCallback is received from the application. The parameter of onSignal() must be matched. (See 2.1.4 signalCallback msg type)

Parameter

lock : msg type of SignalCallback from DmbService to DAB/DMB application. Refer to the sample code as to how processing on the application is handled.

mDmb : Dmb class instance from return value of DmbService while the application performs Dmb.open()

Return value

None

Call Flow

None

3.2.10 DataCallback

```
public interface DataCallback
{
 void onData(int type, int size, byte[] data, Dmb mDmb);
}
```

This is a callback interface for receiving data such as DLS, SLS, Signal Information, FIC, Pad Datagroup, and Packet Datagroup. (See 3.2.7 Dmb.data.)

This must be implemented in case a DataCallback is received from the application. The parameter of onData() must be matched.

Please refer to 2. Data Type and Structure for more information on msg types and structures of the messages that are sent to the application via DataCallback.

Parameter

type : msg type of DataCallback from DmbService to DAB/DMB application. Refer to sample code regarding to used variety and processing on the application

size : data size to be sent from DataCallback to DAB/DMB application

data : data to be sent from DataCallback to DAB/DMB application

mDmb : Dmb class instance from return value of DmbService while the application performs Dmb.open()

Return value

None

Call Flow

None

3.2.11 EventCallbck

```
public interface EventCallback
{
 void onEvent(int event, Dmb mDmb);
}
```

This is a callback interface for sending to the application events occurring in the framework. (See 3.2.3 Dmb.init and 2.1.3 EventCallback msg type)

This must be implemented in case a EventCallback is received from the application. The parameter of onEvent() must be matched.

Parameter

None

Return value

Event : msg type of EventCallback from DmbService to Dmb Application. Refer to sample code as to how processing on the application is handled.

mDmb : Dmb class instance from return value of DmbService while the application performs Dmb.open()

Call Flow

None

3.2.12 Dmb.getFIC

```
public final int getFIC
(
 int Freq
)
```

The raw FIC (Fast Information Channel) data of the channel is sent to DataCallback. This is used to search channels or get the FIC information on the current channel.

(Raw FIC Data format : see ETSI EN 300 401 V1.4.1)

메모 [u7]: Add sentence

Parameter

Freq: Frequency of the channel to receive FIC information

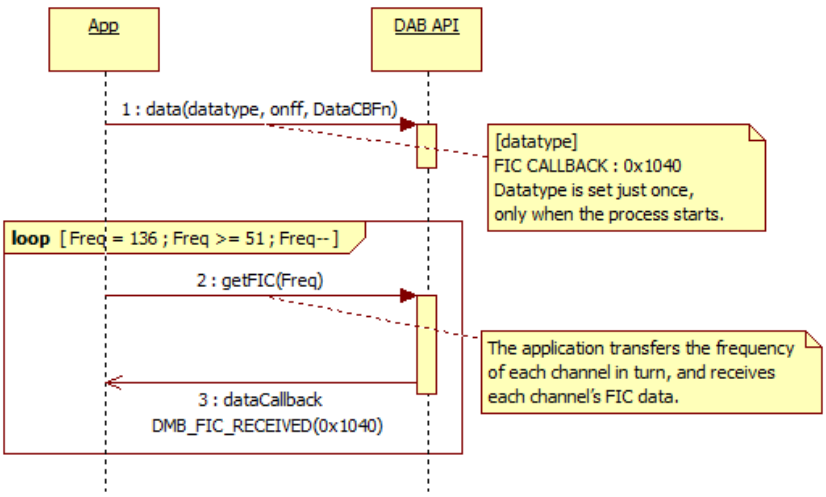
Return value

0 : NO ERROR

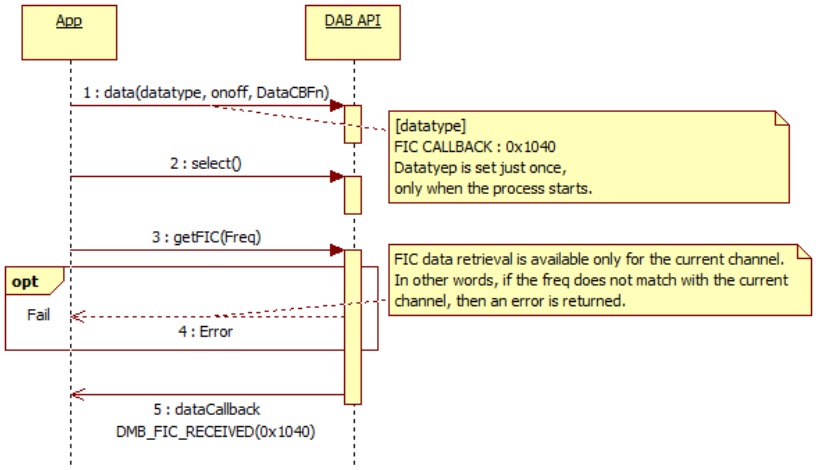
Others : ERROR

Call Flow

Channel search



Acquisition of FIC information while watching broadcasts



Note

getFIC() shall be called after applying select() to receive FIC data while watching broadcasts.
If the Freq value is 0, then it stops receiving FIC data.

3.2.13 Dmb.getDataGroup

```
public final void getDataGroup
(
 int DataGroup
)
```

The MOT DataGroup of a particular channel is sent to the DataCallback function.

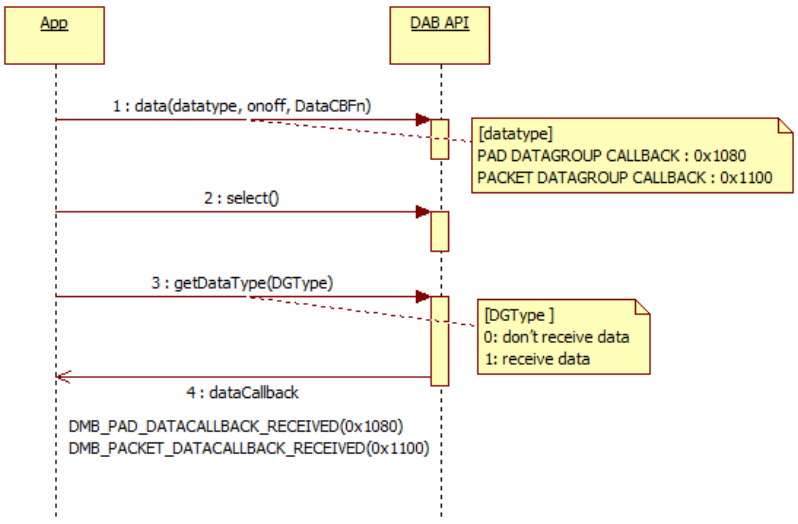
Parameter

DataGroup: When this value is 0, callback for the datagroup is not received. When this value is 1, callback is received for the datagroup. PAD DataGroup is sent to the application as raw data via DataCallback. Please refer to 2.2.4 PACKET DataGroup for more information on the structure of PACKET DataGroup.

Return values

None

Call Flow



Note

getDataGroup() shall be called after applying select() since the purpose of this is to receive the MOT DataGroup.