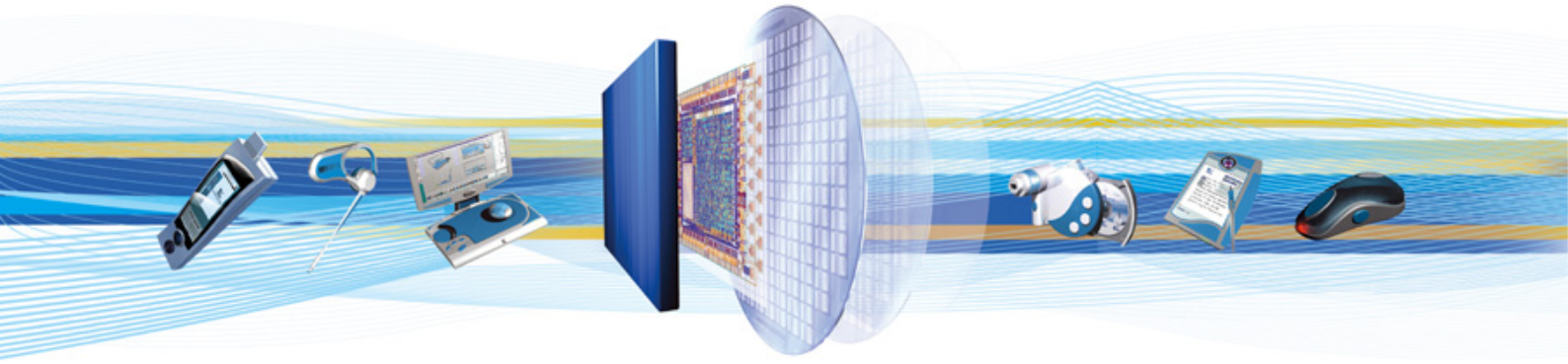
CSR Synergy Bluetooth 18.2.0

Message Access Profile Client

API Description

November 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the OBEX Message Access Profile Client (MAPC). The MAPC conforms to the Messaging client of the Message Access Profile, ref. [MAP]. The MAPC is intended to interoperable with devices supporting Messaging server role of the MAP, ref. [MSE].

1.2 Assumptions

The following assumptions and preconditions made:

- There is a secure and reliable transport between the profile part, i.e. MAPC and the application
- The MAPC shall handle only one request from application at any instant
- Bonding (pairing) is NOT handled by the MAPC
- MAPC doesn't not handle bMessage parsing and building

2 Description

2.1 Introduction

The following scenarios are covered by MAP profile:

- The handsfree unit in the car receives/sends messages from/to a mobile phone which provides capabilities both for network access and message repository
- PC acting as MAPC such that the user is able to use it's PC as IO=device for the messages stored in the cellular phone

The MAPC provides the following services to the application:

- Connection handling
- OBEX protocol handling
- Notify the arrival of new messages on the messaging devices
- Browsing messages in a messaging devices
- Uploading messages onto a messaging devices
- Deleting messages onto a messaging devices
- Sending messages to the network through a messaging devices

The application is responsible for handling the requests and confirms from the MAPC with correct data (object) as described in the IrOBEX specification. The MAPC does not validate the correctness of the data packet for example whether the data are packed correctly with white spaces in the right places, for details see ref. [MAP] and [OBEX].

2.2 Reference Model

The MAPC interfaces to the Connection Manager (CM).

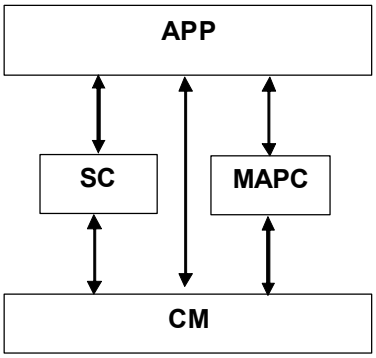


Figure 1: Reference model

2.3 Sequence Overview

In IDLE state, if a connect request is received from the application, the MAPC starts to connect to the specified device and on successful connection with MSE, the MAPC enters CONNECTED state, upon which the application receives a confirmation on the connect request. The application can then issue a request to perform MAP operations which includes folder browsing, getting message listing, get messages, change message status., push message and register for notification. All these operations can be performed any number of times without disconnecting the MAPC session. When the application disconnects the service, the MAPC re-enters IDLE state.

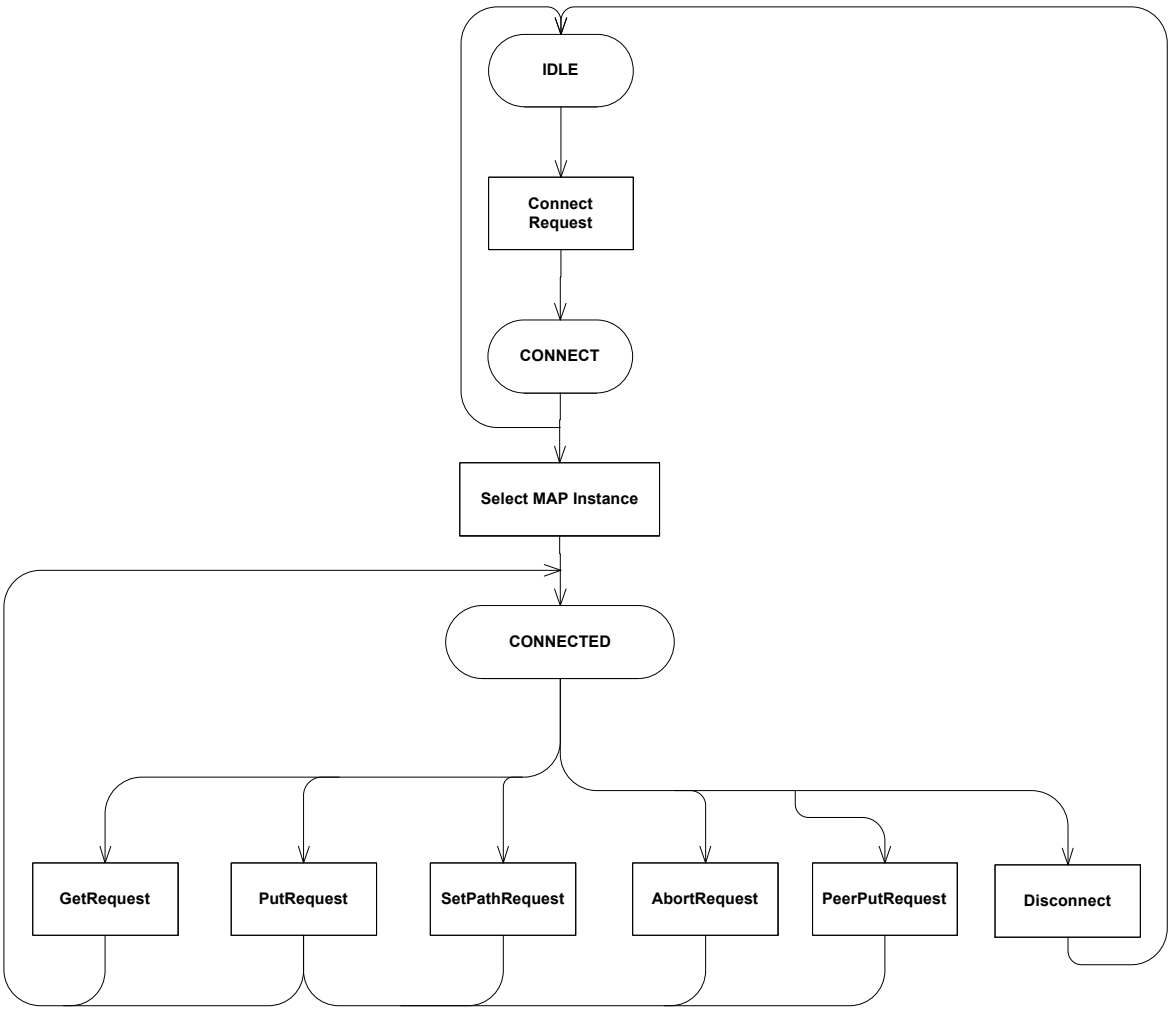


Figure 2: MAPC state diagram

3 Interface Description

3.1 Result codes

MAPC does not have any error code by its own. It uses the result codes from the CSR_BT_SUPPLIER_IRDA_OBEX as result supplier. However, in case of connect and disconnect messages, the result supplier maybe different depending upon the protocol on which the failure occurs.

3.2 Connect

When the application wants to connect to a Message Access Profile Server it has to send a CSR_BT_MAPC_CONNECT_REQ to the MAPC. In this message, the application shall specify which device to connect to. This message has a parameter called maxPacketSize, which indicates the maximum Obex packet size the application is capable of receiving from the MSE. The value can be between 255 bytes to 64Kbytes – 1, see definition in ref. [OBEX]. If the packet size is large it is optimizing for quick reception of message and folder/message listing, but need to trade-off by using large memory blocks.

On successful establishment of the underlying channel with the MSE, the MAPC presents the application with the serviceName, masInstanceId and supportedMessages that are supported by the MSE to the application through CSR_BT_MAPC_SELECT_MAS_INSTANCE_IND. The application have the choice to choose the instance which is of its interest and respond to the indication through CSR_BT_MAPC_SELECT_MAS_INSTANCE_RES, where the application needs to make sure that the masInstanceId matches to that of the CSR_BT_MAPC_SELECT_MAS_INSTANCE_IND. The MAPC then proceeds further in establishing the MAP connection to the MAP instance specified.

The MAPC sends a CSR_BT_MAPC_CONNECT_CFM message to the application, which has the result of the connection establishment - this is resultCode parameter. For successful request the resultCode shall be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE; any other resultCode indicates a failure in the connection. It is worth to mention that the application may not get CSR_BT_MAPC_SELECT_MAS_INSTANCE_IND, if it fails to establish the underlying channel for MAP. In all cases, the application shall receive CSR_BT_MAPC_CONNECT_CFM.

Once the Obex connection is established, the MAPC will make use of low power modes. This implies use of sniff on the Bluetooth link if supported by the connected MSE as well. Low power modes are controlled using a supervision timer. If no data transaction happens within the specified time interval, the MAPC manager will attempt a change to low power mode. The value of the timer is determined by the CSR_BT_MAPC_LP_SUPERVISION_TIMEOUT and is defined in the csr_bt_usr_config.h file.

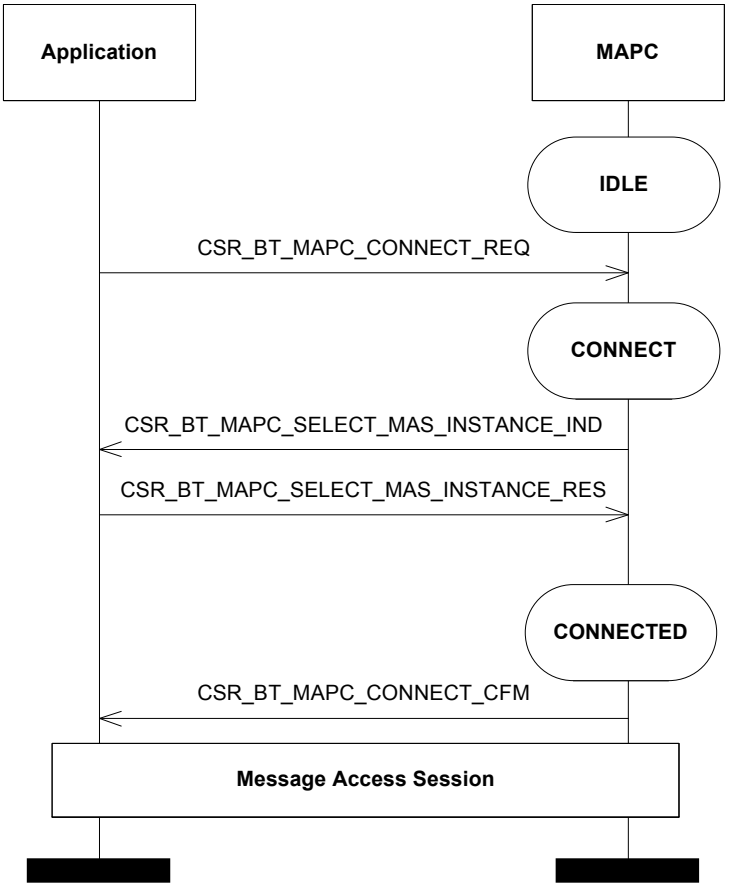


Figure 3: Connection handling

3.3 Cancel Connect

The application can cancel a pending outgoing connection request by sending a CSR_BT_MAPC_CANCEL_CONNECT_REQ. If the outgoing connection was successfully cancelled, then the CSR_BT_MAPC_CONNECT_CFM with carry a response code different to that of CSR_BT_OBEX_SUCCESS_RESPONSE_CODE.

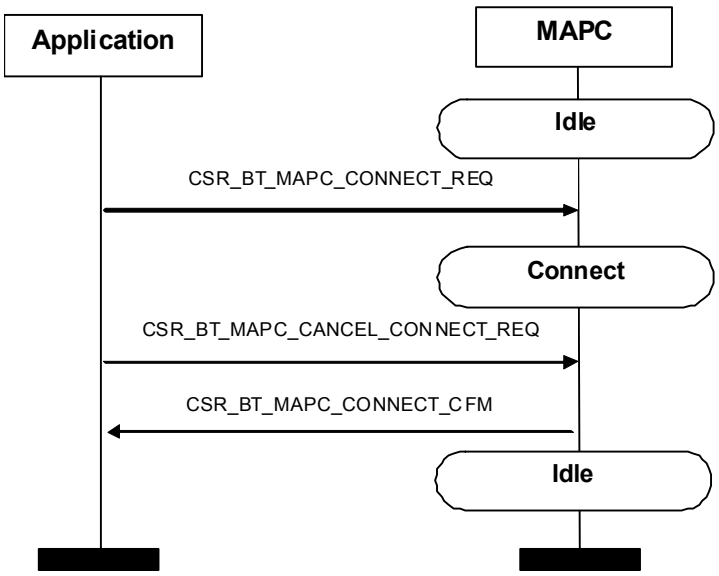


Figure 4: Cancel Connect I

Please note that in the scenario where the application request of CSR_BT_MAPC_CANCEL_CONNECT_REQ is processed after the MAPC sent a CSR_BT_MAPC_CONNECT_CFM with the response code CSR_BT_OBEX_SUCCESS_RESPONSE_CODE to the application, then MAPC will consider the CSR_BT_MAPC_CANCEL_CONNECT_REQ as a CSR_BT_MAPC_DISCONNECT_REQ and the application will receive a CSR_BT_MAPC_DISCONNECT_IND.

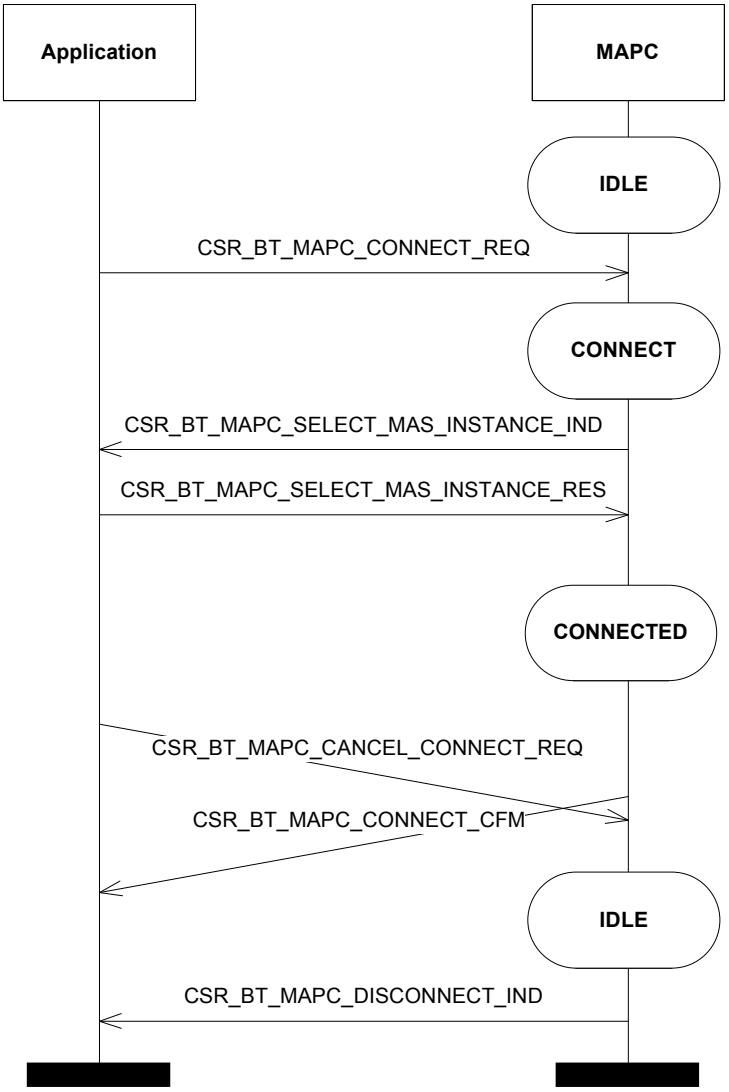


Figure 5: Cancel Connect II

3.4 Folder Browsing

Browsing an object store involves retrieving folder contents and changing the ‘current folder’ forwards and backwards. The CSR_BT_MAPC_SET_FOLDER_REQ is used for changing the current folder forward in the folder hierarchy. There are two ways to go back in the folder hierarchy – the first way is the CSR_BT_MAPC_SET_ROOT_FOLDER_REQ changing the current folder to the root folder (the root folder is the start up folder after a successful CSR_BT_MAPC_CONNECT_CFM). The second way is through CSR_BT_MAPC_SET_BACK_FOLDER_REQ which changes the current folder back to the parent folder of the current one. To retrieve a folder hierarchy starting with the root folder, the client must read the folder content using CSR_BT_MAPC_GET_FOLDER_LISTING_REQ and the application responses with the folder list in the body of the CSR_BT_MAPC_GET_FOLDER_LISTING_IND message.

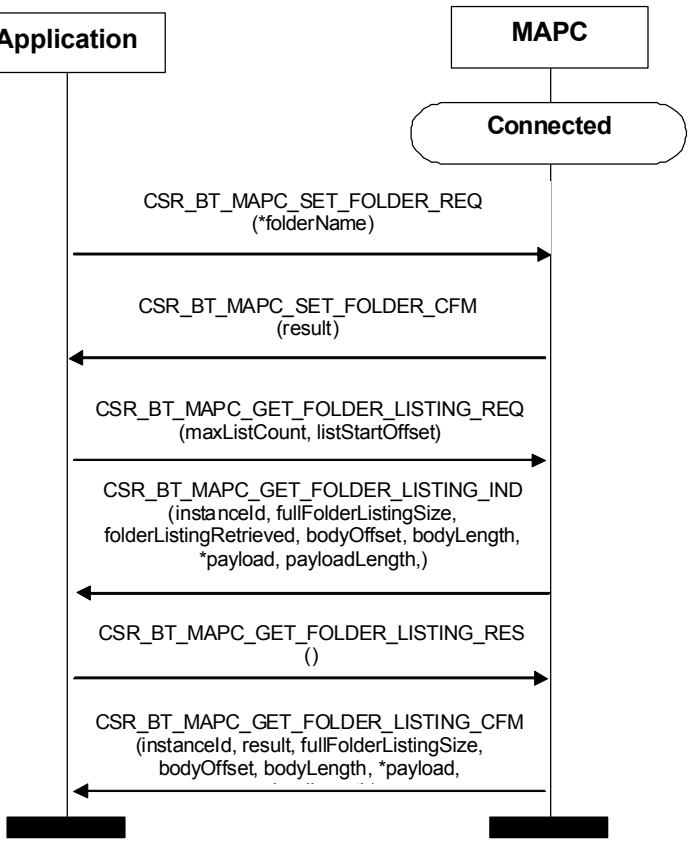


Figure 6: Folder browsing handling

3.5 Message Browsing

Browsing a message store involves retrieving of message content. The CSR_BT_MAPC_GET_MESSAGE_LISITNG_REQ is used for retrieving the message listing of the current or child folder. To retrieve the message listing of the child folder, the name of the child folder shall be send through folderName parameter. Setting the folderName to NULL will retrieve the message listing of the current folder. If the application wants to retrieve the number of messages available in the folder instead of entire message listing, then the maxListCount needs to be set to zero. For other information on how to use the filters and parameterMask, please refer to MAP specification.

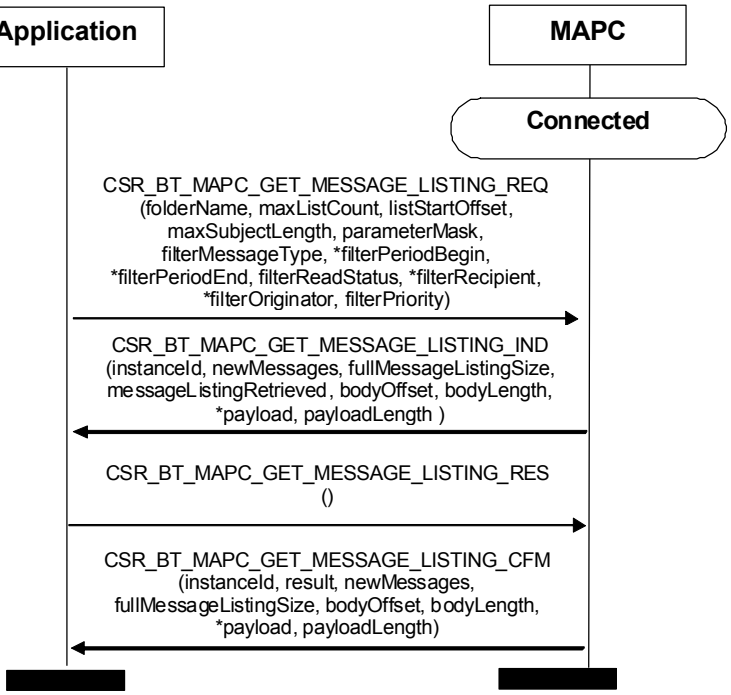


Figure 7: Message browsing handling

3.6 Push Message

Messages are transmitted to the server by using CSR_BT_MAPC_PUSH_MESSAGE_REQ signal followed by a CSR_BT_MAPC_PUSH_MESSAGE_RES. The server side responds with the result of the operation in a CSR_BT_MAPC_PUSH_MESSAGE_CFM signal. In case the application wants to fragment the body due to memory considerations it can do so, by sending a CSR_BT_MAPC_PUSH_MESSAGE_RES with finalFlag set to FALSE. On confirmation, the application can continue to send the next fragment. The application can continue sending the message fragment until it sets the finalFlag to TRUE. For other parameters, please refer to MAP specification.

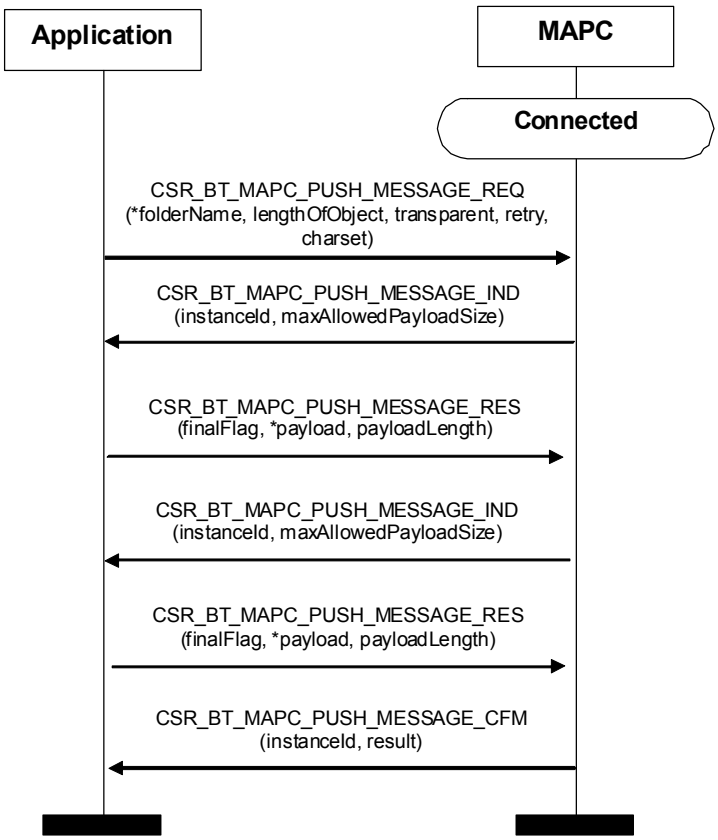


Figure 8: Push Message

3.7 Get Message

Messages can be pulled from the server by using CSR_BT_MAPC_GET_MESSAGE_REQ signal. The server responds with the result of the operation in CSR_BT_MAPC_GET_MESSAGE_CFM signal. If the server responds with multiple fragments, then the application will be notified through CSR_BT_MAPC_GET_MESSAGE_IND on which, the application shall respond with CSR_BT_MAPC_GET_MESSAGE_RES to get the next fragment. This procedure continues until CSR_BT_MAPC_GET_MESSAGE_CFM is received by the application.

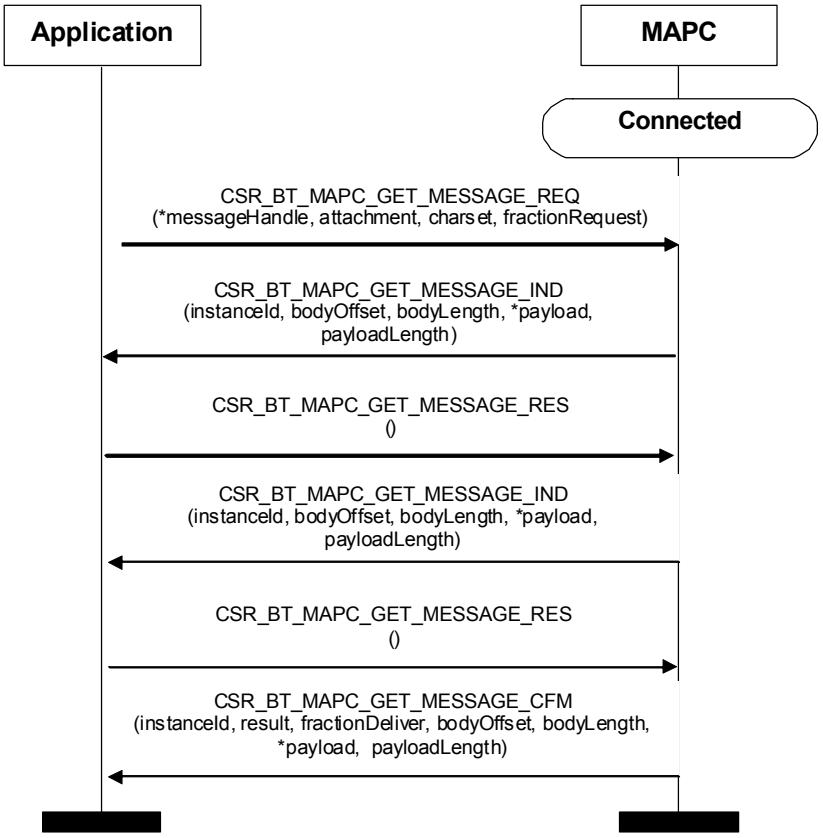


Figure 9: Get Message

3.8 Set Message Status

Manipulating messages includes deleting messages and setting the message as read/unread. The value of statusIndicator, statusValue and the behaviour MSE are defined in MAP specification.

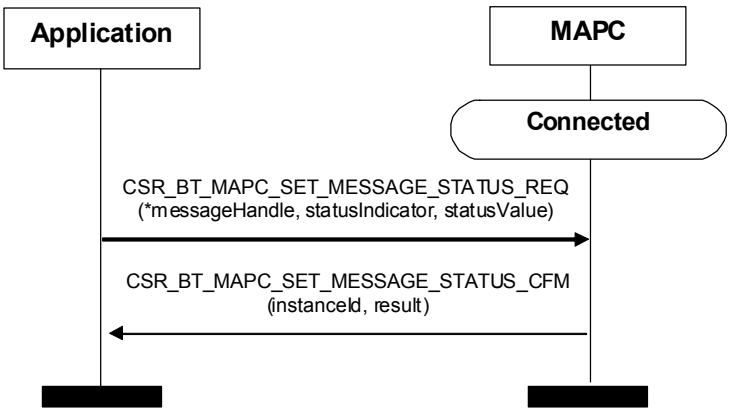


Figure 10: Set Message Status

3.9 Abort Operation

The abort request (CSR_BT_MAPC_ABORT_REQ) can be used when the application decides to terminate a multi-packet operation (such as PUT or GET) before it is completed. The response (CSR_BT_MAPC_ABORT_CFM) indicates that the abort request is successful. It also indicates that the abort request received by the MAP server is resynchronized with the client. If the operation is not successful, then the MAPC will disconnect the MAP session and send CSR_BT_MAPC_DISCONNECT_IND to the application.

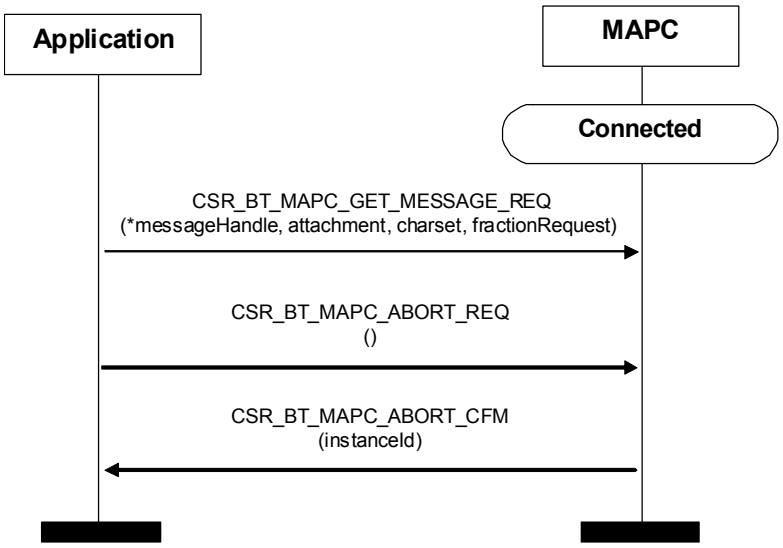


Figure 11: Abort operation

3.10 Notification Service

The application can register the notification service with MSE, if the application is required to track the arrival of NewMessages or to track the status of the message pushed to outbox folder. This can be achieved by using CSR_BT_MAPC_NOTIFICATION_REQ signal. The registering/deregistering of the notification service can be achieved by toggling the enableNotifications parameter in CSR_BT_MAPC_NOTIFICATION_REQ . The status of the request is notified to the application through CSR_BT_MAPC_NOTIFICATION_CFM.

The parameter mnsController in CSR_BT_MAPC_NOTIFICATION_REQ shall be used with care. If there are more than one MSE instances in the peer MSE this flag should only be set as TRUE for the first Notification registration (i.e. where "enableNotifications == TRUE") made to an MSE instance in the peer MSE. This is because the specification only allows for one Message Notification Session [MNS] channel even though there are multiple Message Access Session [MAS] channels to the same device. This MNS registration will be valid until the MNS session is discontinued by either the peer MSE (in which case the task which sent this signal mnsController == TRUE will receive a CSR_BT_MAPC_REGISTRATION_NOTIFICATION_OFF_IND) or when the task which sent this signal with mnsController == TRUE disconnects its MAS session. If either of the above happens, the registration notification for all other tasks that sent CSR_BT_MAPC_NOTIFICATION_REQ with mnsController == FALSE and which are connected to the same MSE device will become void and the task would have to re-register if the tasks continue to wish to receive notification. In addition, the task, which sends CSR_BT_MAPC_NOTIFICATION_REQ with mnsController == TRUE and enableNofitications == TRUE, shall be capable of handling continuing notifications even though it uses this signal to deregister for notifications. The latter can only happen in case other tasks connected other MSE instances on the same MSE device has also registered for notifications with mnsController == FALSE and enableNotifications == TRUE.

In case of a notification event from MSE, the application will receive CSR_BT_MAPC_EVENT_NOTIFICATION_IND, for which the application shall respond with CSR_BT_MAPC_EVENT_NOTIFICATION_RES. The parameter finalFlag in CSR_BT_MAPC_EVENT_NOTIFICATION_IND is to indicate whether the indication is final fragment for this notification.

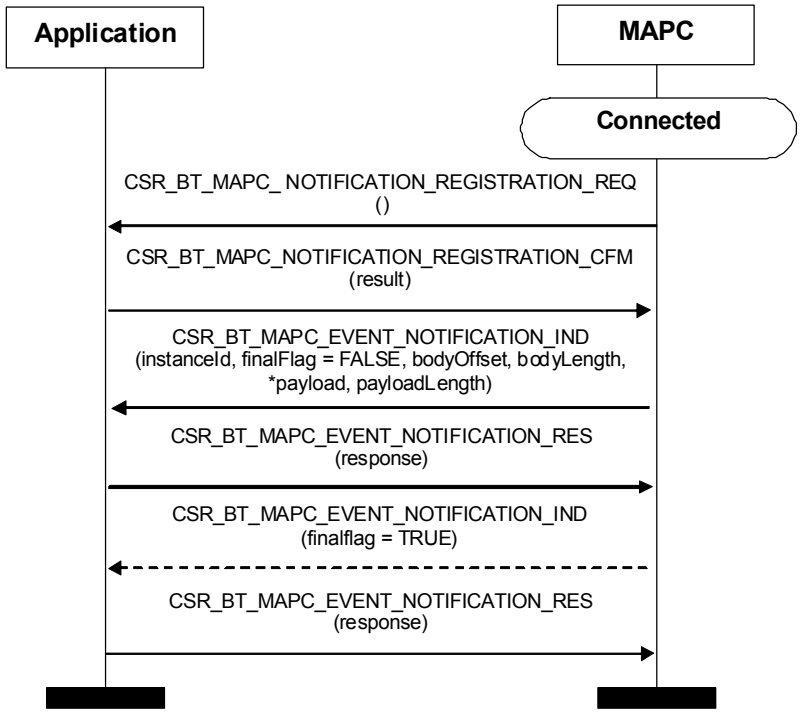


Figure 12: Notification Service

3.11 Disconnect

Sending CSR_BT_MAPC_DISCONNECT_REQ signal to the MAPC disconnects the current connection (if any). The disconnect might take ample amount of time and is confirmed with a CSR_BT_MAPC_DISCONNECT_IND signal.

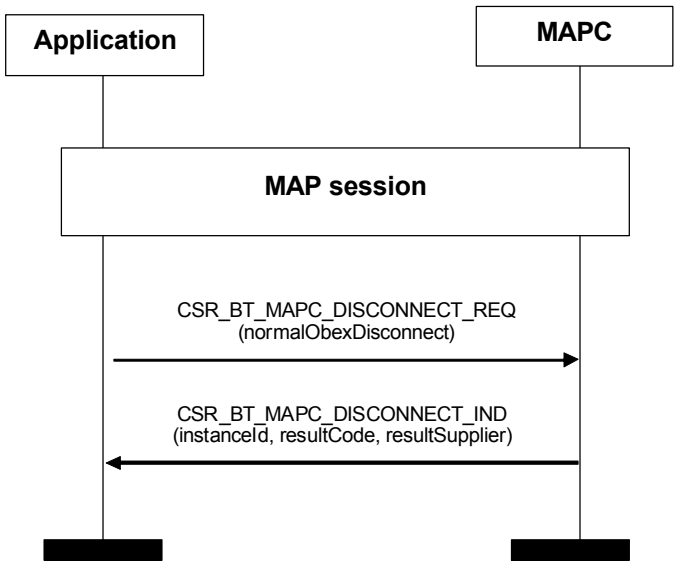


Figure 13: Normal disconnect

In case the peer side prematurely disconnects, the MAPC sends a CSR_BT_MAPC_DISCONNECT_IND to the application and enters IDLE state.

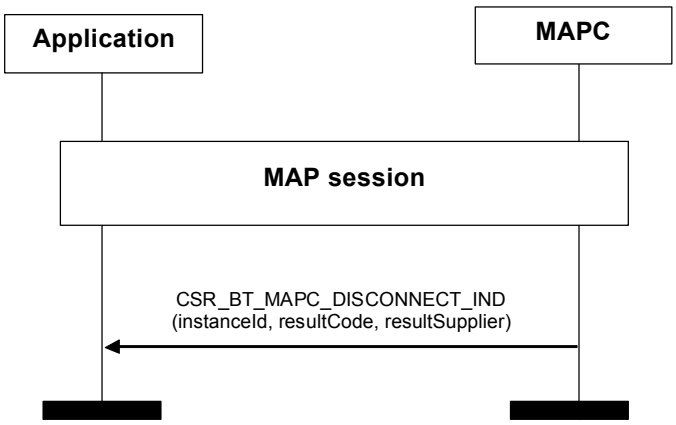


Figure 14: Abnormal disconnect

3.12 Payload Encapsulated Data

3.12.1 Using Offsets

Since many OBEX messages contain multiple parameters with variable length, some of the parameters are based on *offsets* instead of standard pointers to the data. Signals with offset-based data can easily be recognized as they have both a *payload* and a *payloadLength* parameter. The *payload* contains the actual data, on which the offset is based. For example, a typical signal may contain the following:

```
CsrCommonPrim  type;
CsrUint8       result;
CsrUint16  ucs2nameOffset;
CsrUint16  bodyOffset;
CsrUint16  bodyLength;
CsrUint16  payloadLength;
CsrUint8       *payload;
```

In this example, two offset parameters can be found, namely *ucs2nameOffset* and *bodyOffset*. To obtain the actual data, the offset value is added to the *payload* pointer, which yields a pointer to the data, i.e.:

```
CsrUint8  *ucs2name;
ucs2name = (CsrUint8*)(primitive->payload + primitive->ucs2nameOffset);
```

As can be seen, the offset contains the number of bytes within the *payload* where the information begins. Similarly, the body data can be retrieved using the following:

```
CsrUint8 *body;
body = (CsrUint8*)(primitive->payload + primitive->bodyOffset);
```

And to illustrate the usage of the *length* parameter, which is also a common parameter, to copy the body one would typically use:

```
CsrMemCpy( copyOfBody, body, primitive->bodyLength );
```

Offset parameters will always have an “Offset” suffix on the name, and offsets are *always* relative to the “payload” parameter.

If the `bodyOffset` or the `bodyLength` is 0 (zero) it means that the signal does not contain any body. The same holds when the `payloadLength` is 0 (zero), which means that there is not payload.

3.12.2 Payload Memory

When the application receives a signal, which has a *payload* parameter, the application must always free the payload pointer to avoid memory leaks, for example

```
CsrPfree(primitive->payload);
CsrPfree(primitive);
```

will free both the payload data and the message itself. Note that when the payload has been freed, offsets cannot be used anymore, as the actual data is contained within the payload.

Signals that do not use the *payload* parameter must still have each of their pointer-based parameters freed.

4 OBEX Message Access Profile Client Primitives

This section gives an overview of the primitives and parameters in the interface. Detailed information are available in the corresponding csr_bt_mapc_prim.h file.

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_MAPC_GET_INSTANCE_IDS_REQ	See section 4.2
CSR_BT_MAPC_GET_INSTANCE_IDS_CFM	See section 4.2
CSR_BT_MAPC_CONNECT_REQ	See section 4.3
CSR_BT_MAPC_CONNECT_CFM	See section 4.3
CSR_BT_MAPC_SELECT_MAS_INSTANCE_IND	See section 0
CSR_BT_MAPC_SELECT_MAS_INSTANCE_RES	See section 0
CSR_BT_MAPC_CANCEL_CONNECT_REQ	See section 4.5
CSR_BT_MAPC_DISCONNECT_REQ	See section 4.6
CSR_BT_MAPC_DISCONNECT_IND	See section 4.6
CSR_BT_MAPC_SET_FOLDER_REQ	See section 4.7
CSR_BT_MAPC_SET_FOLDER_CFM	See section 4.7
CSR_BT_MAPC_SET_BACK_FOLDER_REQ	See section 4.8
CSR_BT_MAPC_SET_BACK_FOLDER_CFM	See section 4.8
CSR_BT_MAPC_SET_ROOT_FOLDER_REQ	See section 4.9
CSR_BT_MAPC_SET_ROOT_FOLDER_CFM	See section 4.9
CSR_BT_MAPC_GET_FOLDER_LISTING_REQ	See section 4.10
CSR_BT_MAPC_GET_FOLDER_LISTING_RES	See section 4.10
CSR_BT_MAPC_GET_FOLDER_LISTING_IND	See section 4.10
CSR_BT_MAPC_GET_FOLDER_LISTING_CFM	See section 4.10
CSR_BT_MAPC_GET_MESSAGE_LISTING_REQ	See section 4.11
CSR_BT_MAPC_GET_MESSAGE_LISTING_RES	See section 4.11
CSR_BT_MAPC_GET_MESSAGE_LISTING_IND	See section 4.11
CSR_BT_MAPC_GET_MESSAGE_LISTING_CFM	See section 4.11
CSR_BT_MAPC_GET_MESSAGE_REQ	See section 4.12
CSR_BT_MAPC_GET_MESSAGE_RES	See section 4.12
CSR_BT_MAPC_GET_MESSAGE_IND	See section 4.12
CSR_BT_MAPC_GET_MESSAGE_CFM	See section 4.12
CSR_BT_MAPC_SET_MESSAGE_STATUS_REQ	See section 4.13
CSR_BT_MAPC_SET_MESSAGE_STATUS_CFM	See section 4.13
CSR_BT_MAPC_PUSH_MESSAGE_REQ	See section 4.14
CSR_BT_MAPC_PUSH_MESSAGE_IND	See section 4.14
CSR_BT_MAPC_PUSH_MESSAGE_RES	See section 4.14
CSR_BT_MAPC_PUSH_MESSAGE_CFM	See section 4.14
CSR_BT_MAPC_UPDATE_INBOX_REQ	See section 4.15
CSR_BT_MAPC_UPDATE_INBOX_CFM	See section 4.15
CSR_BT_MAPC_ABORT_REQ	See section 4.16
CSR_BT_MAPC_ABORT_CFM	See section 4.16
CSR_BT_MAPC_NOTIFICATION_REGISTRATION_REQ	See section 4.17

Primitives:	Reference:
CSR_BT_MAPC_NOTIFICATION_REGISTRATION_CFM	See section 4.17
CSR_BT_MAPC_NOTIFICATION_REGISTRATION_OFF_IND	See section 4.17
CSR_BT_MAPC_EVENT_NOTIFICATION_IND	See section 4.18
CSR_BT_MAPC_EVENT_NOTIFICATION_RES	See section 4.18
CSR_BT_MAPC_EVENT_NOTIFICATION_ABORT_IND	See section 4.19
CSR_BT_MAPC_SECURITY_IN_REQ	See section 4.20
CSR_BT_MAPC_SECURITY_IN_CFM	See section 4.20
CSR_BT_MAPC_SECURITY_OUT_REQ	See section 0
CSR_BT_MAPC_SECURITY_OUT_CFM	See section 0

Table 1: List of all primitives

4.2 CSR_BT_MAPC_GET_INSTANCE_IDS

Parameters / Primitives	type	appHandle	instanceIdsListSize	*instanceIdsList
CSR_BT_MAPC_GET_INSTANCE_IDS_REQ	✓	✓		
CSR_BT_MAPC_GET_INSTANCE_IDS_CFM	✓		✓	✓

Table 2: CSR_BT_MAPC_GET_INSTANCE_IDS Primitives

Description

This signal is used for getting the list of registered MAPC instances that is active.

Parameters

type	Signal identity, CSR_BT_MAPC_GET_INSTANCE_IDS_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
instanceIdsListSize	Number of _items_ in instanceIdsList, not length in bytes.
*instanceIdsList	List of instance

4.3 CSR_BT_MAPC_CONNECT

Primitives \ Parameters	type	appHandle	maxPacketSize	deviceAddr	instanceId	masInstanceId	serviceName	supportedMessages	obexPeerMaxPacketSize	resultCode	resultSupplier	length	count	btConnId	windowSize	srmEnable
CSR_BT_MAPC_CONNECT_REQ	✓	✓	✓	✓								✓	✓		✓	✓
CSR_BT_MAPC_CONNECT_CFM	✓				✓	✓	✓	✓	✓	✓	✓			✓		

Table 3: CSR_BT_MAPC_CONNECT Primitives

Description

To start an MAP session with a MAP server, the application needs to send a CSR_BT_MAPC_CONNECT_REQ with the desired maxPacketSize that it can receive from MAP server. The server responds with a CSR_BT_MAPC_CONNECT_CFM. In successful establishment of MAP session, a resultCode with CSR_BT_OBEX_SUCCESS_RESULT_CODE is sent in CSR_BT_MAPC_CONNECT_CFM. Any other resultCode indicates a failure in the connection establishment.

The connect messages between the OBEX MAPC and MSE is guarded by a timer, thus if for some reason the server does not reply to the OBEX connect request within a fixed time interval the Bluetooth channel (RFCOMM) is disconnected abruptly. The timeout functionality is per default set to five seconds. The timeout value can be disabled, or changed by modifying CSR_BT_OBEX_CONNECT_TIMEOUT, which is define in csr_bt_user_config_default.h. Note if the value of CSR_BT_OBEX_CONNECT_TIMEOUT is change, it will influence all OBEX based profiles.

The function:

CsrBtMapcConnectReqSend (CsrSchedQid appHandle, CsrUint16 maxPacketSize,
CsrBtDeviceAddr deviceAddr, CsrUint32 length, CsrUint32 count,
CsrUint16 windowSize, CsrBool srmEnable);

defined in csr_bt_mapc_lib.h, builds and sends the CSR_BT_MAPC_CONNECT_REQ primitive to the MAPC profile. The function *CsrBtMapcConnectExtReqSend* is use if the count and length header must be included in the OBEX CONNECT packet.

Parameters

type	Signal identity, CSR_BT_MAPC_CONNECT_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
maxPacketSize	The maximum obex packet size that the application can handle at any instance.
deviceAddr	Bluetooth Device address to which MAP session needs to be establishment to.
instanceId	Identifier of the MAPC instance that generated this event.
masInstanceId	The MASInstanceID reported by the peer MSE SDP record to which the MAP session is established
serviceName	ServiceName of MSE from SDP record

supportedMessages	Bit pattern of supported message types in the server. Refer to SDP record of MAP specification.
obexPeerMaxPacketSize	Max OBEX packet size supported by the MAP server. The application shall not exceed the size when sending messages to the MSE
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
length	Length is use to express the approximate total length of the bodies of all the objects in the transaction. If set to 0 this header will not be include.
count	Count is use to indicate the number of objects that will be sent during this connection. If set to 0 this header will not be include.
btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend`-function.
windowSize	Controls how many packets the OBEX profile (and lower protocol layers) are allowed to cache on the data receive side. A value of zero (0) will cause the system to auto-detect this value.
srmEnable	Enable local support for Single Response Mode.

4.4 CSR_BT_MAPC_SELECT_MAS_INSTANCE

Primitives \ Parameters	type	instanceId	*masInstanceList	masInstanceListSize	proceedWithConnection	masInstanceId
CSR_BT_MAPC_SELECT_MAS_INSTANCE_IND	✓	✓	✓	✓		
CSR_BT_MAPC_SELECT_MAS_INSTANCE_RES	✓				✓	✓

Table 4: CSR_BT_MAPC_SELECT_MAS_INSTANCE Primitives

Description

The signal is used for notifying the application about the MAP instances supported by the MSE. The application shall respond with CSR_BT_MAPC_SELECT_MAS_INSTANCE_RES with the desired masInstanceId to which the application is interested to connect.

Parameters

type	Signal identity, CSR_BT_MAPC_SELECT_MAS_INSTANCE_IND/RES.
instanceId	Identifier of the MAPC instance that generated this event
*masInstanceList	Pointer to the list of available MAS instances on the peer MSE. This pointer is of the type CsrBtMapcMasInstance and is described below.
masInstanceListSize	Number of items in masInstanceList
proceedWithConnection	If TRUE, the connection establishment continues with the given masInstanceId. If FALSE, the MAPC will abort the connection establishment and send a CSR_BT_MAPC_CONNECT_CFM with resultCode other than CSR_BT_OBEX_SUCCESS_RESULT_CODE
masInstanceId	MASInstanceId to which MAP session connection needs to be attempted

Description of the type CsrBtMapcMasInstance:

serviceName	ServiceName of MSE from SDP record
masInstanceId	MASInstanceID of MSE from SDP record
supportedMessages	Bit pattern of supported message types in this MAS instance. Please refer to MAP specification for details.

4.5 CSR_BT_MAPC_CANCEL_CONNECT

Primitives \ Parameters	type
CSR_BT_MAPC_CANCEL_CONNECT_REQ	✓

Table 5: CSR_BT_MAPC_CANCEL_CONNECT Primitives

Description

The CSR_BT_MAPC_CANCEL_CONNECT_REQ can be used for cancelling an ongoing connection procedure. If the MAPC succeeds in cancelling the ongoing connection attempt the application will receive a CSR_BT_MAPC_CONNECT_CFM with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. If the application request of CSR_BT_MAPC_CANCEL_CONNECT_REQ is processed after the MAPC sent a CSR_BT_MAPC_CONNECT_CFM with the response code CSR_BT_OBEX_SUCCESS_RESPONSE_CODE to the application, then MAPC will consider the CSR_BT_MAPC_CANCEL_CONNECT_REQ as a CSR_BT_MAPC_DISCONNECT_REQ and the application will receive a CSR_BT_MAPC_DISCONNECT_IND.

Parameters

type Signal identity, CSR_BT_MAPC_CANCEL_CONNECT_REQ.

4.6 CSR_BT_MAPC_DISCONNECT

Parameters / Primitives	type	normalDisconnect	instanceId	reasonCode	reasonSupplier
CSR_BT_MAPC_DISCONNECT_REQ	✓	✓			
CSR_BT_MAPC_DISCONNECT_IND	✓		✓	✓	✓

Table 6: CSR_BT_MAPC_DISCONNECT Primitives

Description

To disconnect a connection to a MAP server (if any), the application needs to send a CSR_BT_MAPC_DISCONNECT_REQ to the MAPC. When disconnected, the MAPC will respond with a CSR_BT_MAPC_DISCONNECT_IND. If the Bluetooth link or the underlying channel is dropped in the middle of a MAP session, the application will receive a CSR_BT_MAPC_DISCONNECT_IND indicating that the OBEX MAP session is terminated, and MAPC is ready to handle a re-connection from the application..

The disconnect messages between the OBEX Message Access client and Server is guarded by a timer, thus if for some reason the server do not reply to the OBEX disconnect request within a fixed time interval the Bluetooth connection is disconnected direct. The timeout functionality is per default set to five seconds. The timeout value can be disabled, or changed by modifying CSR_BT_OBEX_DISCONNECT_TIMEOUT, which is define in csr_bt_user_config_default.h. Note if the value of CSR_BT_OBEX_DISCONNECT_TIMEOUT is change, it will influence all OBEX based profiles.

Parameters

type	Signal identity, CSR_BT_MAPC_DISCONNECT_REQ/IND.
normalDisconnect	If TRUE, an OBEX disconnect is issued; this will lead to a graceful disconnection of the MAP session. If FALSE, the, the underlying transport channel (RFCOMM) is disconnected without sending the OBEX disconnect (abrupt disconnection).
instanceId	Identifier of the MAPC instance that generated this event.
reasonCode	The reason code of the operation. Possible values depend on the value of reasonSupplier. If e.g. the reasonSupplier == CSR_BT_SUPPLIER_CM then the possible reason codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
reasonSupplier	This parameter specifies the supplier of the reason given in reasonCode. Possible values can be found in csr_bt_result.h

4.7 CSR_BT_MAPC_SET_FOLDER

Primitives \ Parameters	type	*folderName	instanceId	result
CSR_BT_MAPC_SET_FOLDER_REQ	✓	✓		
CSR_BT_MAPC_SET_FOLDER_CFM	✓		✓	✓

Table 7: CSR_BT_MAPC_SET_FOLDER Primitives

Description

This signal is used for changing the current folder on the MAP server to a folder specified with the folderName parameter. This signal can be used for navigating down in the directory hierarchy on the server. The result of the change folder operation is specified in the confirm signal. The result may contain error codes corresponding to the reason for failure in case the folder does not exist or in case the server does not permit this operation.

Parameters

type	Signal identity, CSR_BT_MAPC_SET_FOLDER_REQ/CFM.
*folderName	Null terminated name string of the folder to change.
instanceId	Identifier of the MAPC instance that generated this event.
result	OBEX result of the CSR_BT_MAPC_SET_FOLDER_REQ operation.

4.8 CSR_BT_MAPC_SET_BACK_FOLDER

Primitives \ Parameters	type	instanceId	result
CSR_BT_MAPC_SET_BACK_FOLDER_REQ	✓		
CSR_BT_MAPC_SET_BACK_FOLDER_CFM	✓	✓	✓

Table 8: CSR_BT_MAPC_SET_BACK_FOLDER Primitives

Description

This signal is used for setting the current folder on the MAP server back to the parent folder. The result of the operation is specified in the CSR_BT_MAPC_SET_BACK_FOLDER_CFM signal. If the current folder is the root folder the confirm will have the CSR_BT_OBEX_NOT_FOUND_RESPONSE_CODE result code.

Parameters

type	Signal identity, CSR_BT_MAPC_SET_BACK_FOLDER_REQ/CFM.
instanceId	Id of the MAPC instance that generated this event
result	The valid result codes are defined (in csr_bt_obex.h). In case of successful operation, the result is set to CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure of the operation.

4.9 CSR_BT_MAPC_SET_ROOT_FOLDER

Primitives \ Parameters	type	instanceId	result
CSR_BT_MAPC_SET_ROOT_FOLDER_REQ	✓		
CSR_BT_MAPC_SET_ROOT_FOLDER_CFM	✓	✓	✓

Table 9: CSR_BT_MAPC_SET_ROOT_FOLDER Primitives

Description

This signal is used for setting the current folder on the MAP server back to the root folder. The result in the CSR_BT_MAPC_SET_ROOT_FOLDER_CFM message can contain error codes corresponding to the reason for the failure on the server.

Parameters

type	Signal identity, CSR_BT_MAPC_SET_ROOT_FOLDER_REQ/CFM.
instanceId	Identifier of the MAPC instance that generated this event
result	The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.

4.10 CSR_BT_MAPC_GET_FOLDER_LISTING

Primitives \ Parameters	type	maxListCount	listStartOffset	instanceId	fullFolderListingSize	folderListingRetrieved	bodyOffset	bodyLength	*payload	payloadLength	result	smpOn
CSR_BT_MAPC_GET_FOLDER_LISTING_REQ	✓	✓	✓									✓
CSR_BT_MAPC_GET_FOLDER_LISTING_RES	✓											✓
CSR_BT_MAPC_GET_FOLDER_LISTING_IND	✓			✓	✓	✓	✓	✓	✓	✓		
CSR_BT_MAPC_GET_FOLDER_LISTING_CFM	✓			✓	✓		✓	✓	✓	✓	✓	

Table 10: CSR_BT_MAPC_GET_FOLDER_LISTING Primitives

Description

The signal is used for retrieving the folder-listing object from the current folder of MSE.

Parameters

Type	Signal identity, CSR_BT_MAPC_GET_FOLDER_LISTING_REQ/RES/IND/CFM.
maxListCount	Maximum number of folders to be listed in the folder listing.
listStartOffset	Offset from where the listing needs to be started.
instanceId	Identifier of the MAPC instance that generated this event.
fullFolderListingSize	Number of bytes in the entire folder listing.
folderListingRetrieved	Number of bytes of the folder listing received so far
bodyOffset	Payload relative offset to where the body part starts. NB: Only valid if bodyLength is greater than zero.
bodyLength	Length of the object body carried with this payload.
*payload	Pointer to the complete OBEX payload received from the server.
payloadLength	Total length of the entire payload.
result	OBEX result of the CSR_BT_MAPC_GET_FOLDER_LISTING_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.
smpOn	Reserved for future use. Set to FALSE.

4.11 CSR_BT_MAPC_GET_MESSAGE_LISTING

Primitives \ Parameters	type	*folderName	maxListCount	listStartOffset	maxSubjectLength	parameterMask	filterMessageType	*filterPeriodBegin	*filterPeriodEnd	filterReadStatus	*filterRecipient	*filterOriginator	filterPriority	instanceId	newMessages	mseTime	fullMessageListingSize	messageListingRetrieved	bodyOffset	bodyLength	*payload	payloadLength	result	smpOn
CSR_BT_MAPC_GET_MESSAGE_LISTING_REQ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓											✓
CSR_BT_MAPC_GET_MESSAGE_LISTING_RES	✓																							✓
CSR_BT_MAPC_GET_MESSAGE_LISTING_IND	✓													✓	✓	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_MAPC_GET_MESSAGE_LISTING_CFM	✓													✓	✓	✓	✓		✓	✓	✓	✓	✓	

Table 11: CSR_BT_MAPC_GET_MESSAGE_LISTING Primitives

Description

This signal is used the retrieve Messages-listing from the MSE of the folder specified by folderName.

Parameters

type	Signal identity, CSR_BT_MAPC_GET_MESSAGE_LISTING_REQ/RES/IND/CFM
*folderName	Null terminated name string of the folder from where the message listing is to be retrieved. The folderName specified shall be one of the child folder of the current folder of MSE. If folderName is set to NULL, the message listing of the current folder will be retrieved.
maxListCount	Maximum number of folders in the listing. To retrieve the number of messages in the folder, maxListCount shall be set to zero
listStartOffset	Offset from where to the listing should start.
maxSubjectLength	Maximum string length allowed on the subject field on each messages.
parameterMask	Bitmask of relevant parameters for the message listing. NB: a bit value of 1 means that the parameter should be present and a value of 0 means it should be filtered out. The MSE is expected to adhere to parameterMask. The details of parameterMask is defined in MAP specification.
filterMessageType	Bitmask specifying which message types should be filtered in the listing. NB: a bit value of 1 means that the message type should be filtered and a value of 0 means that it should be present. The MSE is expected to adhere to filterMessageType. Details of filterMessageType is defined in MAP specification.
*filterPeriodBegin	Null terminated time string that may be used for filtering the messages by delivery date newer than specified. The MSE is expected to adhere to filterPeriodBegin.
*filterPeriodEnd	Null terminated time string that may be used for filtering the messages by delivery date older than specified. The MSE is expected to adhere to filterPeriodEnd.
filterReadStatus	Bitmask specifying if filtering should be done based on read status. The details of the bit is defined in MAP specification. The MSE is expected to adhere to

	filterReadStatus.
*filterRecipient	Null terminated recipient string. The MSE filters the message listing based on the filterRecipient in respective vCard attributes. The details of filterRecipient is defined in MAP specification.
*filterOriginator	Null terminated originator string. The MSE filters the message listing based on the filterOriginator in respective vCard attributes. The details of filterOriginator is defined in MAP specification.
filterPriority	Bitmask specifying which priority type to be included in the message listing. The details of filterPriority is defined in MAP specification.
instanceId	Identifier of the MAPC instance that generated this event.
newMessages	Set to TRUE if there is/are unread messages on MSE.
*mseTime	Current time basis of MSE and UTC-offset. Null terminated time string
fullMessageListingSize	Number of bytes in the entire message listing.
messageListingRetrieved	Number of bytes of the message listing received so far.
bodyOffset	Payload relative offset to where the body part starts. NB: Only valid if bodyLength is greater than zero
bodyLength	Length of the object body carried with this payload.
*payload	Pointer to the entire OBEX payload received from the server.
payloadLength	Total length of the payload.
result	OBEX result of CSR_BT_MAPC_GET_MESSAGE_LISTING_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.
smpOn	Reserved for future use. Set to FALSE.

4.12 CSR_BT_MAPC_GET_MESSAGE

Primitives \ Parameters	type	*messageHandle	attachment	charset	fractionRequest	instanceId	bodyOffset	bodyLength	*payload	payloadLength	result	fractionDeliver	smpOn
CSR_BT_MAPC_GET_MESSAGE_REQ	✓	✓	✓	✓	✓								✓
CSR_BT_MAPC_GET_MESSAGE_RES	✓												✓
CSR_BT_MAPC_GET_MESSAGE_IND	✓					✓	✓	✓	✓	✓			
CSR_BT_MAPC_GET_MESSAGE_CFM	✓					✓	✓	✓	✓	✓	✓	✓	

Table 12: CSR_BT_MAPC_GET_MESSAGE Primitives

Description

This signal is used for retrieving a message specified by messageHandle form MSE.

Parameters

type	Signal identity, CSR_BT_MAPC_GET_MESSAGE_REQ/RES/IND/CFM.
*messageHandle	Null terminated message handle string that the application is intended to retrieve from MSE
attachment	Bitmask specifying whether to include attachment of the message or not. The MSE is expected to adhere to attachment parameter.
charset	Bitmask used for specifying the desired trans-coding of the message requested. The supported charset are defined in MAP specification.
fractionRequest	Bitmask to request which fragment of the message to retrieve. This parameter is application only if the message in MSE is fractioned. The details of fractionRequest is defined in MAP specification.
instanceId	Identifier of the MAPC instance that generated this event.
bodyOffset	Payload relative to offset from where the body begins. NB: Only valid if bodyLength is greater than zero
bodyLength	Length of the OBEX body object specified in this payload.
*payload	Pointer to the entire OBEX payload received from the server.
payloadLength	Total length of the payload.
result	OBEX result of CSR_BT_MAPC_GET_MESSAGE_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.
fractionDeliver	Bitmask specifying the fragment status of the message retrieved if available in MSE.
smpOn	Reserved for future use. Set to FALSE.

4.13 CSR_BT_MAPC_SET_MESSAGE_STATUS

Parameters / Primitives	type	*messageHandle	statusIndicator	statusValue	instanceId	result
CSR_BT_MAPC_SET_MESSAGE_STATUS_REQ	✓	✓	✓	✓		
CSR_BT_MAPC_SET_MESSAGE_STATUS_CFM	✓				✓	✓

Table 13: CSR_BT_MAPC_SET_MESSAGE_STATUS Primitives

Description

This signal allows the application to modify the status of a message on MSE e.g. changing the message status from unread to read.

Parameters

Type	Signal identity, CSR_BT_MAPC_SET_MESSAGE_STATUS_REQ/CFM.
*messageHandle	Null terminated message handle string of the message in MSE that the application is intended to set the message status
statusIndicator	Specifies which status indicator needs to be set. The detail of statusIndicator is defined in MAP specification.
statusValue	Specifies the value of the status indicator that needs to be set. The detail if statusValue parameter is defined in MAP specification.
instanceId	Identifier of the MAPC instance that generated this event.
result	OBEX result of CSR_BT_MAPC_SET_MESSAGE_STATUS_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.

4.14 CSR_BT_MAPC_PUSH_MESSAGE

Primitives \ Parameters	type	*folderName	lengthOfObject	transparent	retry	charset	instanceId	maxAllowedPayloadSize	finalFlag	*payload	payloadLength	result	*messageHandle
CSR_BT_MAPC_PUSH_MESSAGE_REQ	✓	✓	✓	✓	✓	✓							
CSR_BT_MAPC_PUSH_MESSAGE_IND	✓						✓	✓					
CSR_BT_MAPC_PUSH_MESSAGE_RES	✓								✓	✓	✓		
CSR_BT_MAPC_PUSH_MESSAGE_CFM	✓						✓					✓	✓

Table 14: CSR_BT_MAPC_PUSH_MESSAGE Primitives

Description

This signal allows the application to push a message to a folder on the MSE.

Parameters

type	Signal identity, CSR_BT_MAPC_PUSH_MESSAGE_REQ/IND/RES/CFM.
*folderName	Null terminated Utf8 string specifying the name of the folder in MSE where the message should be pushed. The folderName specified shall be one of the child folder of the current folder of MSE. If folderName is set to NULL, the message will be pushed to the current folder. Please note that the MSE may not allow the MAPC application to all folder.
lengthOfObject	Total length of the message to send, NB: if set to zero this informative OBEX header field will not be included in the PUT request.
transparent	Specifies if the MSE should keep a copy of the message in the sent folder. The parameter transparent is defined in MAP specification.
retry	Specifies if the MSE should try to resent if first delivery to the network fails. The parameter retry is defined in MAP specification.
charset	Specify the format of the message pushed. The details of the charset parameter and the format supported are defined in MAP specification.
instanceId	Identifier of the MAPC instance that generated this event.
maxAllowedPayloadSize	The maximum allowed payload size that can be included in the next CSR_BT_MAPC_PUSH_MESSAGE_RES.
finalFlag	finalFlag shall be set to TRUE if CSR_BT_MAPC_PUSH_MESSAGE_RES carries the final fragment of the message.
*payload	Payload to be sent.
payloadLength	Length of the payload to be sent.
result	OBEX result of CSR_BT_MAPC_PUSH_MESSAGE_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a

	failure in the operation.
*messageHandle	Null terminated string specifying the message handle assigned by the MSE. The messageHandle parameter is needed for the application correlate the notification event if CSR_BT_MAPC_PUSH_MESSAGE_REQ is made to outbox folder. The messageHandle parameter is also required if the application decided to change the statusIndicator of the message later.

4.15 CSR_BT_MAPC_UPDATE_INBOX

Primitives \ Parameters	type	instanceId	result
CSR_BT_MAPC_UPDATE_INBOX_REQ	✓		
CSR_BT_MAPC_UPDATE_INBOX_CFM	✓	✓	✓

Table 15: CSR_BT_MAPC_UPDATE_INBOX Primitives

Description

This signal requests the MSE to perform an inbox update. The MSE is expected to communicate with the network to update the inbox of message server.

Parameters

type	Signal identity, CSR_BT_MAPC_UPDATE_INBOX_REQ/CFM.
instanceId	Identifier of the MAPC instance that generated this event.
result	OBEX result of CSR_BT_MAPC_UPDATE_INBOX_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.

4.16 CSR_BT_MAPC_ABORT

Primitives	type	instanceId
CSR_BT_MAPC_ABORT_REQ	✓	
CSR_BT_MAPC_ABORT_CFM	✓	✓

Table 16: CSR_BT_MAPC_ABORT Primitives

Description

The CSR_BT_MAPC_ABORT_REQ is used when the application decides to terminate a multi-packet operation (such as GET/PUT) before it normally ends. The CSR_BT_MAPC_ABORT_CFM indicates that the abort is received and the MAP server is now resynchronized with the MAP client (MAPC). If the operation is not successful, the MAPC will disconnect the MAP session and send a CSR_BT_MAPC_DISCONNECT_IND to the application.

Parameters

type	Signal identity, CSR_BT_MAPC_ABORT_REQ/CFM.
instanceId	Identifier of the MAPC instance that generated this event.

4.17 CSR_BT_MAPC_NOTIFICATION_REGISTRATION

Primitives	type	enableNotifications	mnsController	instanceId	result
CSR_BT_MAPC_NOTIFICATION_REGISTRATION_REQ	✓	✓	✓		
CSR_BT_MAPC_NOTIFICATION_REGISTRATION_CFM	✓			✓	✓
CSR_BT_MAPC_NOTIFICATION_REGISTRATION_OFF_IND	✓			✓	

Table 17: CSR_BT_MAPC_NOTIFICATION_REGISTRATION Primitives

Description

The application can use this signal to register itself for being notified of the arrival of new message in the MSE or to be indicated by the MSE when the message pushed to outbox was sent successfully to the network. If the MSE disconnects the notification service without MAPC requesting to disable the notification service, the MAPC will notify the application through CSR_BT_MAPC_NOTIFICATION_REGISTRATION_OFF_IND.

Parameters

Type	Signal identity, CSR_BT_MAPC_NOTIFICATION_REGISTRATION_REQ/CFM.
enableNotifications	Specifies whether to enable or disable notifications from the MSE. Setting enableNotifications to TRUE will enable the notification
mnsController	This parameter shall be used with care. If there are more than one MSE instances in the peer MSE this flag should only be set as TRUE for the first Notification registration (i.e. where "enableNotifications == TRUE") made to an MSE instance in the peer MSE. This is because the specification only allows for one Message Notification Session [MNS] channel even though there are multiple Message Access Session [MAS] channels to the same device. This MNS registration will be valid until the MNS session is discontinued by either the peer MSE (in which case the task which sent this signal mnsController==TRUE will receive a CSR_BT_MAPC_REGISTRATION_NOTIFICATION_OFF_IND) or when the task which sent this signal with mnsController==TRUE disconnects its MAS session. If either of the above happens, the registration notification for all other tasks that sent CSR_BT_MAPC_NOTIFICATION_REQ with mnsController==FALSE and which are connected to the same MSE device will become void and the task would have to re-register if the tasks continue to wish to receive notification. In addition, the task, which sends CSR_BT_MAPC_NOTIFICATION_REQ with mnsController == TRUE and enableNofitications == TRUE, shall be capable of handling continuing notifications even though it uses this signal to deregister for notifications. The latter can only happen in case other tasks connected other MSE instances on the same MSE device has also registered for notifications with mnsController==FALSE and enableNotifications==TRUE.
instanceId	Identifier of the MAPC instance that generated this event.
Result	OBEX result of CSR_BT_MAPC_NOTIFICATION_REGISTRATION_REQ. The valid result codes are defined (in csr_bt_obex.h). On success, the result code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other result indicates a failure in the operation.

4.18 CSR_BT_MAPC_EVENT_NOTIFICATION

Primitives \ Parameters	type	instanceId	finalFlag	bodyOffset	bodyLength	*payload	payloadLength	response	smpOn
CSR_BT_MAPC_EVENT_NOTIFICATION_IND	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_MAPC_EVENT_NOTIFICATION_RES	✓							✓	✓

Table 18: CSR_BT_MAPC_EVENT_NOTIFICATION Primitives

Description

This signal is indicated to the application when there is notification from MSE. if an e.g. a new message arrives on the MSE. This signal is active only if the application has registered the notification service through CSR_BT_MAPC_NOTIFICATION_REGISTRATION_REQ.

Parameters

type	Signal identity, CSR_BT_MAPC_EVENT_NOTIFICATION_IND/RES.
instanceId	Identifier of the MAPC instance that generated this event.
finalFlag	If TRUE, the CSR_BT_MAPC_EVENT_NOTIFICATION_IND carries the entire obex packet from the MSE. If FALSE, the remaining OBEX packet can be retrieved by setting OBEX_CONTINUE_RESPONSE_CODE as response in CSR_BT_MAPC_EVENT_NOTIFICATION_RES.
bodyOffset	Payload relative offset to where the OBEX body header begins. NB: Only valid if bodyLength is greater than zero
bodyLength	Length of the object body specified in this payload.
*payload	Pointer to the entire OBEX payload received from the MSE.
payloadLength	Total length of the payload.
response	OBEX response to CSR_BT_MAPC_EVENT_NOTIFICATION_IND. The valid result codes are defined (in csr_bt_obex.h).
smpOn	Reserved for future use. Set to FALSE.

4.19 CSR_BT_MAPC_EVENT_NOTIFICATION_ABORT

Primitives \ Parameters	type	instanceId	descriptionOffset	descriptionLength	*payload	payloadLength
CSR_BT_MAPC_EVENT_NOTIFICATION_ABORT_IND	✓	✓	✓	✓	✓	✓

Table 19: CSR_BT_MAPC_EVENT_NOTIFICATION_ABORT_IND Primitives

Description

This signal indicates to the application that a multi-packet notification has been aborted by the MSE.

Parameters

type	Signal identity, CSR_BT_MAPC_EVENT_NOTIFICATION_ABORT_IND.
instanceId	Identifier of the MAPC instance that generated this event.
descriptionOffset	Payload relative offset to where the description body begins. NB: Only valid if descriptionLength is greater than zero
descriptionLength	Length of the description specified in this payload.
*payload	Pointer to the entire OBEX payload received from the MSE.
payloadLength	Total length of the payload.

4.20 CSR_BT_MAPC_SECURITY_IN

Primitives \ Parameters	type	appHandle	secLevel	instanceId	resultCode	resultSupplier
CSR_BT_MAPC_SECURITY_IN_REQ	✓	✓	✓			
CSR_BT_MAPC_SECURITY_IN_CFM	✓			✓	✓	✓

Table 20: CSR_BT_MAPC_SECURITY_IN Primitives

Description

Applications that wish to change the enforcement to a specific profile security level, i.e. authentication, encryption and/or authorisation, can use this signal to set up the security level for *new* connections. Note that this signal is for the local device only and can be used from within any state.

The CSR_BT_SECURITY_IN_REQ signal sets up the security level for new incoming connections. Already established and pending connections are not altered.

Note, that any attempts to set security to a less secure level than the mandatory security level will be rejected. See csr_bt_profiles.h for mandatory security settings. The default settings used by CSR Synergy Bluetooth are set to require authentication and encryption.

Note that if MITM protection is requested and the remote device does not have the required IO capabilities, pairing/bonding will fail and connections to the remote device *cannot* be made. See [SC] for further details.

Parameters

Type	Signal identity CSR_BT_MAPC_SECURITY_IN_REQ/CFM.
appHandle	Application handle.
secLevel	Security level for incoming connection, see csr_bt_profiles.h.
instanceId	Identifier of the MAPC instance that generated this event.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.21 CSR_BT_MAPC_SECURITY_OUT

Parameters / Primitives	type	appHandle	secLevel	instanceId	resultCode	resultSupplier
CSR_BT_MAPC_SECURITY_OUT_REQ	✓	✓	✓			
CSR_BT_MAPC_SECURITY_OUT_CFM	✓			✓	✓	✓

Table 21: CSR_BT_MAPC_SECURITY_OUT Primitives

Description

Applications that wish to change the enforcement to a specific profile security level, i.e. authentication, encryption and/or authorisation, can use this API to set up the security level for *new* connections. Note that this API is for the local device only and can be used from within any state.

The CSR_BT_SECURITY_OUT_REQ signal sets up the security level for new outgoing connections. Already established and pending connections are not altered.

Note, that any attempts to set security to a less secure level than the mandatory security level will be rejected. See csr_bt_profiles.h for mandatory security settings. The default settings used by CSR Synergy Bluetooth are set to require authentication and encryption.

Note that if MITM protection is requested and the remote device does not have the required IO capabilities, pairing/bonding will fail and connections to the remote device *cannot* be made. See [SC] for further details.

Parameters

type	Signal identity CSR_BT_MAPC_SECURITY_OUT_REQ/CFM.
appHandle	Application handle.
secLevel	Security level for outgoing connection, see csr_bt_profiles.h
instanceId	Identifier of the MAPC instance that generated this event.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

5 Document References

Document	Reference
MESSAGE ACCESS PROFILE Version 1.0 04 June 2009	[MAP]
IrDA Object Exchange Protocol - IrOBEX Version 1.2 18 March 1999	[OBEX]
Specifications for Ir Mobile Communications (IrMC) Version 1.1 01 March 1999	[IRMC]
CSR Synergy Bluetooth, SC – Security Controller API Description, Document no. api-0102-sc	[SC]

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
MAPC	OBEX Message Access Profile Client
MSE	Message Access Server Equipment (MAP Server)
SDS	Service Discovery Server
SIG	Special Interest Group
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards
OBEX	IrMC Object Exchange Protocol

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.