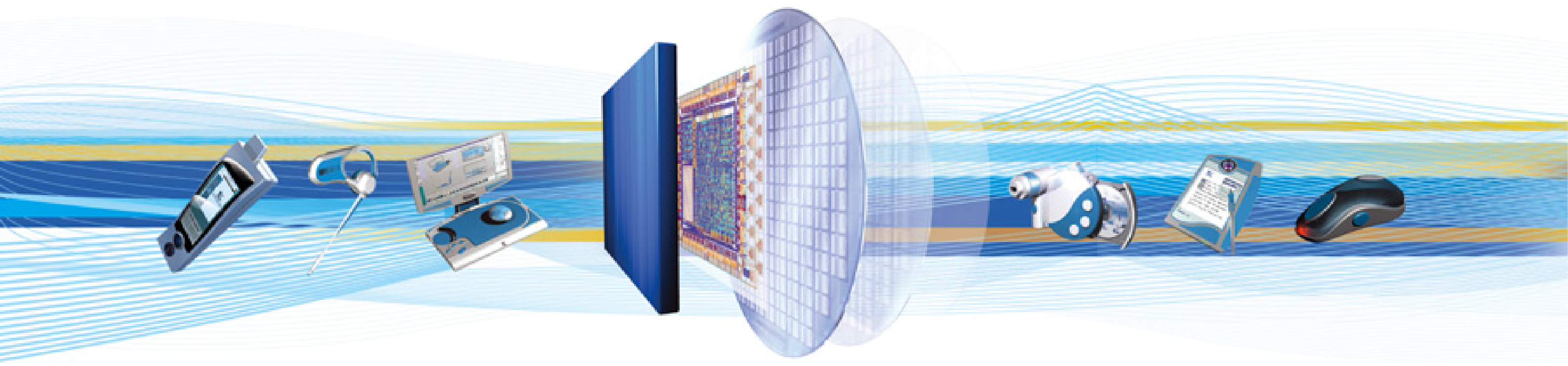
CSR Synergy Bluetooth 18.2.0

MAPS – Message Access Profile Server

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 functionality and message interface provided by CSR Synergy Bluetooth for using the server part of the Message Access Profile (MAP) specified by Bluetooth Special Interest Group (SIG).

2 Description

2.1 Introduction

Mobile Messaging becomes increasingly important. Across the globe, record traffic growth in established messaging markets combined with a number of newer emerging markets caused a volume growth of mobile messaging far beyond anticipated levels. The rapid development of the Smartphone market underlines the increasing importance of mobile messaging and therefore the provision of a profile to address messaging use cases.

The Message Access Profile (MAP) defines the features and procedures that shall be used by devices that exchange message objects. It is based on a Client-Server interaction model where the Client initiates the transactions.

In general MAP will be used for combining the messaging capabilities of a messaging server device and the user interface capabilities of a client device for notifying, browsing, reading, deleting, generation and sending of messages.

For instance the Message Access Profile can be used

- to access in a car to a mobile phone, using the car's display or audio system to avoid cumbersome handling of the mobile phone while driving, providing acoustical announcements of email or SMS reception Text2Speech output of messages etc.,
- to provide message access to a mobile messaging device using any available PC or notebook to leverage the superior display and input capabilities of the computer
- to enable messaging to any stationary or mobile devices with IO capability, e.g. TVs, message panels, digital picture frames, eBooks, pagers, portable navigation systems or wearable Bluetooth devices.

Figure 1 and Figure 2 show typical scenarios involving the MAP profile.

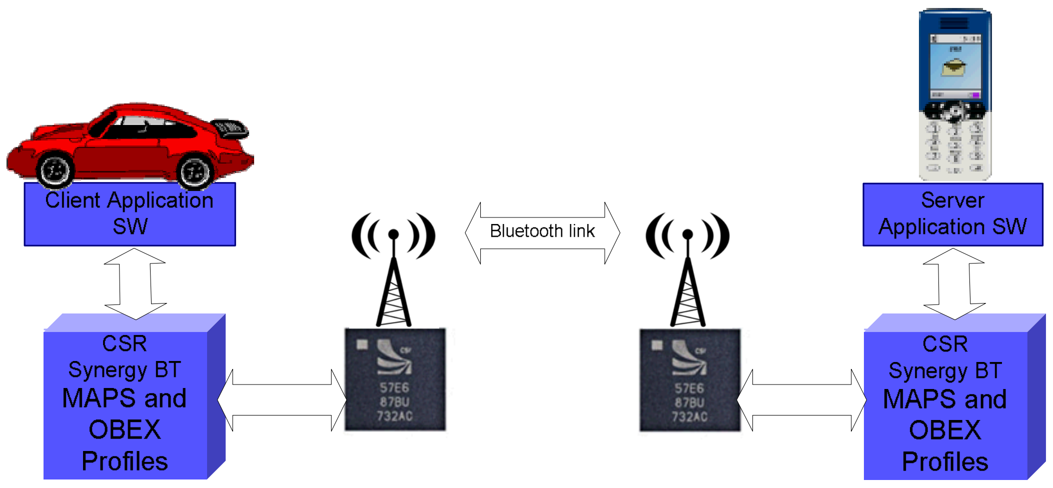

Client Application SW
CSR Synergy BT MAPS and OBEX Profiles
Bluetooth link
Server Application SW
CSR Synergy BT MAPS and OBEX Profiles

Figure 1: Typical scenario

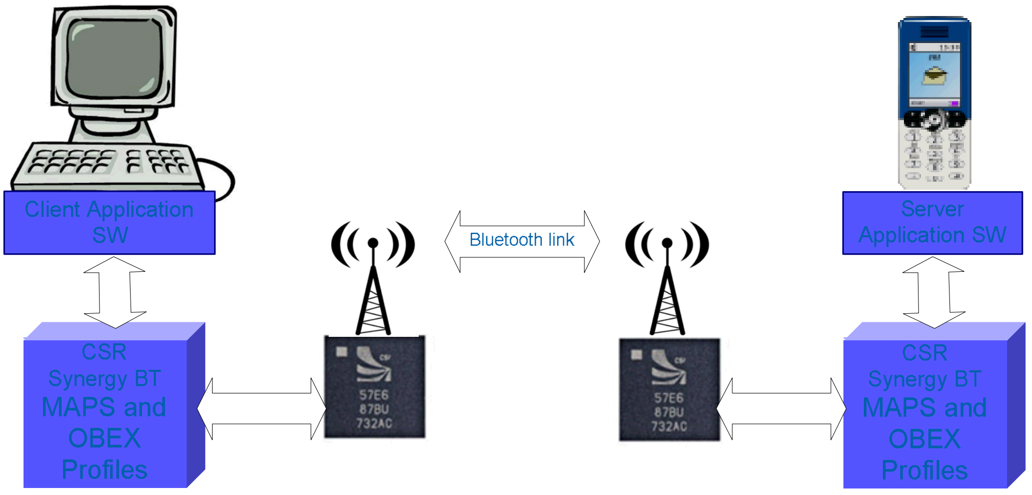


Figure 2: Typical scenario

In Figure 1 the left hand side of the Bluetooth link represent a Bluetooth enabled car kit, while the left hand side of Figure 2 represents a standard PC. These devices connect – via a Bluetooth link - to the mobile phone on the right hand side of the figures. In these typical cases the mobile phone will act as a Message Access Service (MAS) server, and the other devices act as Message Access Service clients.

In standard OBEX connections, the client connects to – and controls – the server. This is also the case for devices supporting the MAP profile. However, in scenarios where both the devices support Message Notification Service (MNS), it is possible for the MAS client (e.g. car kit or PC) to request the MNS service to be started. This will make the MAS server (e.g. mobile phone) setup a MNS connection and act as a client on this connection. The MAS client will then act as MNS server. When the MNS channel is successfully setup, it is possible for the MAS server to notify the MAS client about events occurring on the server, e.g. new massages arriving or messages being moved or deleted.

An MSC of a typical scenario where the server device connects to the client device, browses it and receives a particular message is found in Figure 3. In Figure 4 a MSC showing a scenario where the MAS client is notified about a new message is found. In this figure it is also shown that the MAS client receives the new message from the MAS server.

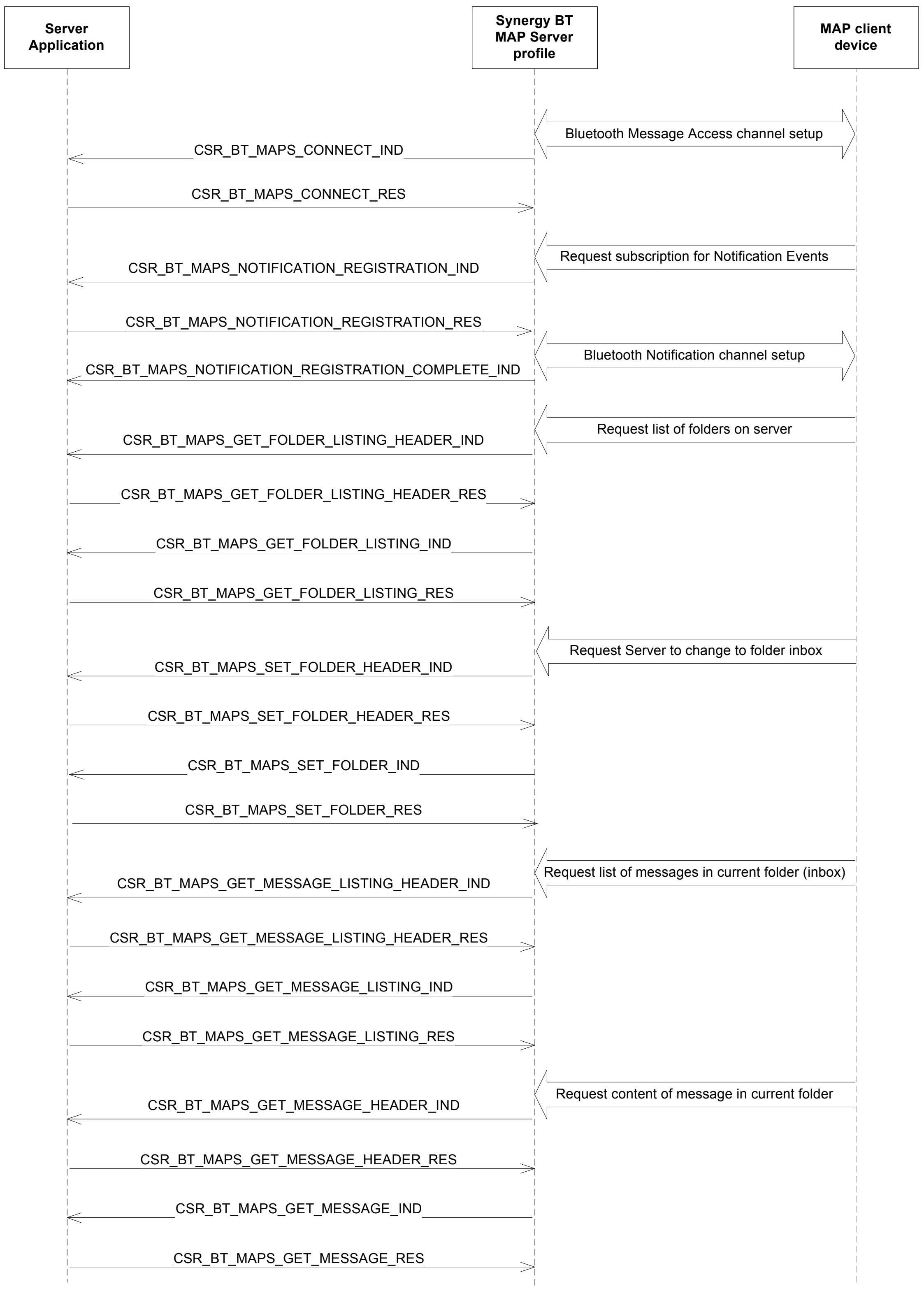

Server Application
Synergy BT MAP Server profile
MAP client device
Bluetooth Message Access channel setup
CSR_BT_MAPS_CONNECT_IND
CSR_BT_MAPS_CONNECT_RES
Request subscription for Notification Events
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_RES
Bluetooth Notification channel setup
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_COMPLETE_IND
Request list of folders on server
CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_IND
CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_RES
CSR_BT_MAPS_GET_FOLDER_LISTING_IND
CSR_BT_MAPS_GET_FOLDER_LISTING_RES
Request Server to change to folder inbox
CSR_BT_MAPS_SET_FOLDER_HEADER_IND
CSR_BT_MAPS_SET_FOLDER_HEADER_RES
CSR_BT_MAPS_SET_FOLDER_IND
CSR_BT_MAPS_SET_FOLDER_RES
Request list of messages in current folder (inbox)
CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_RES
CSR_BT_MAPS_GET_MESSAGE_LISTING_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_RES
Request content of message in current folder
CSR_BT_MAPS_GET_MESSAGE_HEADER_IND
CSR_BT_MAPS_GET_MESSAGE_HEADER_RES
CSR_BT_MAPS_GET_MESSAGE_IND
CSR_BT_MAPS_GET_MESSAGE_RES

Figure 3: Example of how the server device connects to the client device browses it and receives a particular message

In Figure 3 the client device connects to the server device after the server device has been activated, i.e. made ready to accept an MAP connection. The server is activated by sending a CSR_BT_MAP_ACTIVATE_REQ to the CSR Synergy Bluetooth MAP server profile. When the corresponding CSR_BT_MAP_ACTIVATE_CFM is received, the server device is activated.

After the connection is established, the client requests to subscribe to notification events to be able to receive notifications about e.g. new message arriving. This request makes the server setup a notification channel in which it is client.

After the notification channel is setup, the client chooses to request a folder listing from the server. This is to find out which folders are available on the server. Hereafter the client chooses to look for messages in the inbox-folder. To do this, the client must first browse into the inbox-folder and then request a message listing for this folder. The message listing request will supply the client with a list of message in the inbox-folder and the handles for these.

To receive the content of a particular message, the client must send a request to get a particular message.

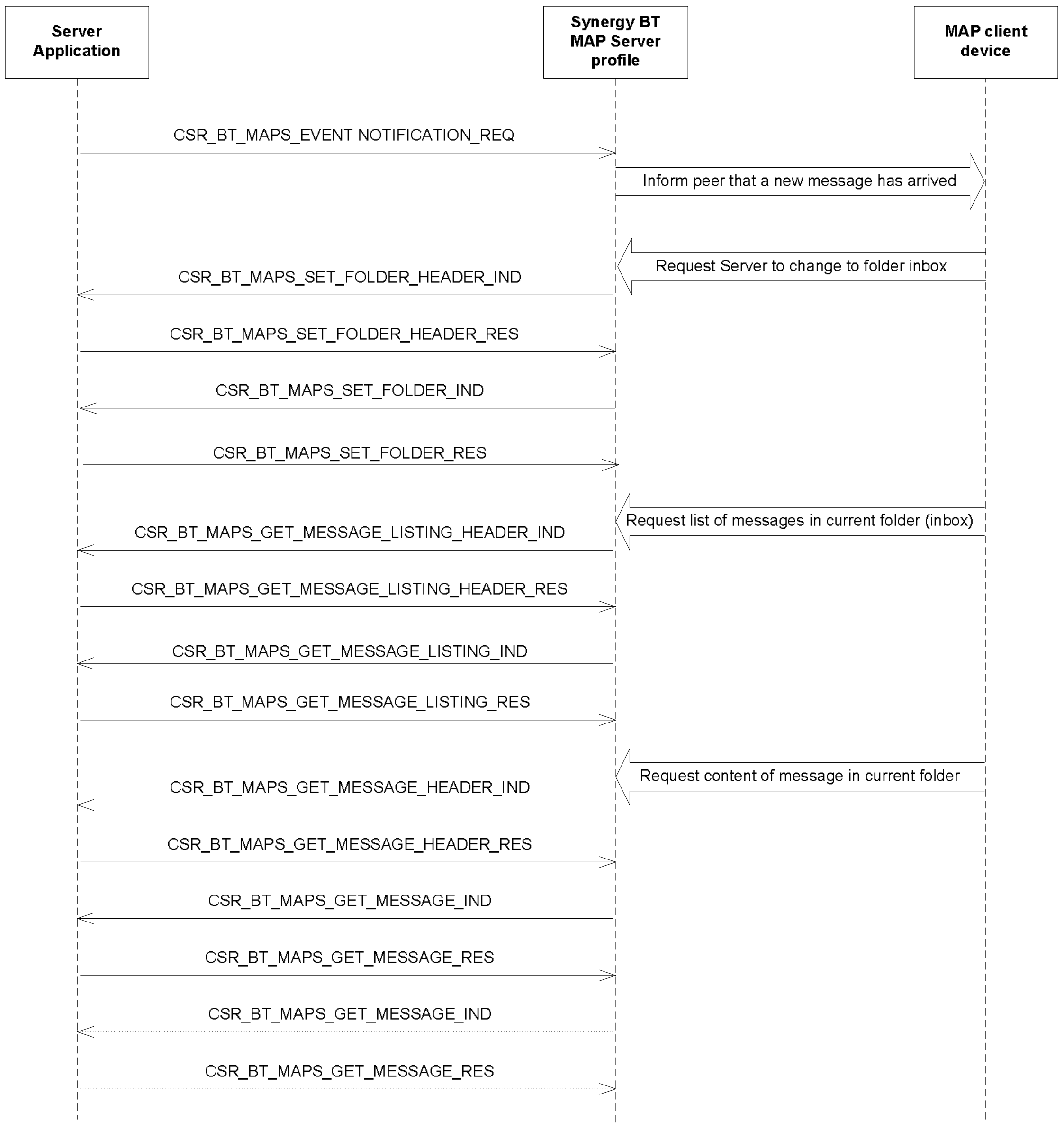

Server Application
Synergy BT MAP Server profile
MAP client device
CSR_BT_MAPS_EVENT NOTIFICATION_REQ
Inform peer that a new message has arrived
Request Server to change to folder inbox
CSR_BT_MAPS_SET_FOLDER_HEADER_IND
CSR_BT_MAPS_SET_FOLDER_HEADER_RES
CSR_BT_MAPS_SET_FOLDER_IND
CSR_BT_MAPS_SET_FOLDER_RES
Request list of messages in current folder (inbox)
CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_RES
CSR_BT_MAPS_GET_MESSAGE_LISTING_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_RES
Request content of message in current folder
CSR_BT_MAPS_GET_MESSAGE_HEADER_IND
CSR_BT_MAPS_GET_MESSAGE_HEADER_RES
CSR_BT_MAPS_GET_MESSAGE_IND
CSR_BT_MAPS_GET_MESSAGE_RES
CSR_BT_MAPS_GET_MESSAGE_IND
CSR_BT_MAPS_GET_MESSAGE_RES

Figure 4: Example of how the MAS client is notified about a new message has arrived and how it receives the content of it

In Figure 4 the client receives a notification event from the server saying that a new message has arrived. To be able to receive these events, the client must have requested the server to receive event notifications beforehand. See Figure 3 for an example on how to send this request. In Figure 4 the client chooses to receive the new message after being notified that is has arrived. The client does that by first browsing into the inbox-folder, then requesting a message listing and finally be requesting to receive the content of the new message. If the message is too large to fit into a single OBEX packet, it is sent in more packets. The server application may therefore have to send more than one CSR_BT_MAPS_GET_MESSAGE_RESs. Note that server application must wait for a CSR_BT_MAPS_GET_MESSAGE_IND before sending the response. This is to make sure that the client is ready for receiving new data.

3 Interface Description

In this section a series of MSCs will be shown to explain the usage of the MAP Server profile. The primitives and the functions available to the application are also described in the subsections of this chapter.

3.1 Activation

For an application to be ready to accept an incoming MAP connection request, the MAP server profile must be activated. This is done by sending a CSR_BT_MAPS_ACTIVATE_REQ to the MAP server profile. This is done by calling `CsrBtMapsActivateReqExtSend` function. The activation of the MAP server profile triggers the registration of a MAS instance, which remote devices will be able to find. The instance needs an ID and a name. The main MAS instance shall take the ID value 0. The arguments for the function are described in Table 1.

Type	Argument	Description
CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_MAPS_ACTIVATE_CFM is sent

CsrBtMapMesSupport	supportedMessages	Bit pattern defining the supported message types of the server. Available options are:
		
		CSR_BT_MAP_NO_TYPE_SUPPORT = 0x00
		CSR_BT_MAP_EMAIL_SUPPORT = 0x01
		CSR_BT_MAP_SMS_GSM_SUPPORT = 0x02
		CSR_BT_MAP_SMS_CDMA_SUPPORT = 0x04
		CSR_BT_MAP_MMS_SUPPORT = 0x08
		CSR_BT_MAP_ANY_TYPE_SUPPORT = 0xFF

CsrUint16	obexMaxPacketSize	To control the maximum allowed obex packet size the application can receive. There is a define CSR_BT_MAX_OBEX_SIGNAL_LENGHT (in csr_bt_obex.h) to be used for this value, the max allowed value is 64K bytes – 1.
CsrUint16	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.
CsrBool	srmEnable	Enable local support for Single Response Mode.
CsrUint8	instanceId	Unique identifier of the instance to activate
CsrUint16	nameLen	Length of the name that the MAS instance shall have. Shall be 0 if no name is given.
CsrUint8	*name	The name given to the MAS instance. If no name is given, this parameter shall be NULL. The MAP server profile will provide a sensible name for the MAS instance in that case.

Table 1: Arguments for `CsrBtMapsActivateExtReqSend` function

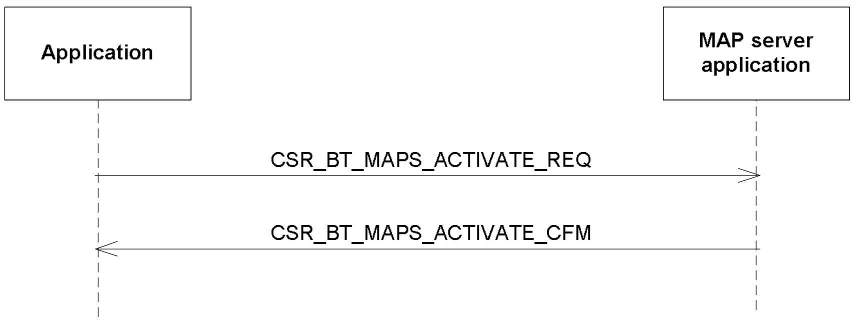


Figure 5: MAP server Activation

When the MAP server profile is ready to accept a connection, a CSR_BT_MAPS_ACTIVATE_CFM primitive, which contains the parameters found in Table 2, is sent to the application.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_ACTIVATE_CFM
CsrSchedQid	instanceId	ID of the MAPS instance that generated the event
CsrBtResultCode	resultCode	The result code of the operation. Possible values depends on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_OBEX_PROFILES then the possible result codes can be found in csr_bt_obex_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 2: Parameters in a CSR_BT_MAPS_ACTIVATE_CFM primitive

NOTE: The application is not allowed to send other primitives to the MAP server profile before it has received a CSR_BT_MAPS_ACTIVATE_CFM primitive with resultCode = CSR_BT_RESULT_CODE_OBEX_SUCCESS and resultSupplier = CSR_BT_SUPPLIER_OBEX_PROFILES.

3.2 Deactivation

When the application running on top of the CSR Synergy Bluetooth MAP server profile does not want to use the MAP server profile anymore, the MAP server profile can be deactivated by sending a CSR_BT_MAPS_DEACTIVATE_REQ. This is done using the `CsrBtMapsDeactivateReqSend` function. This function takes no arguments.

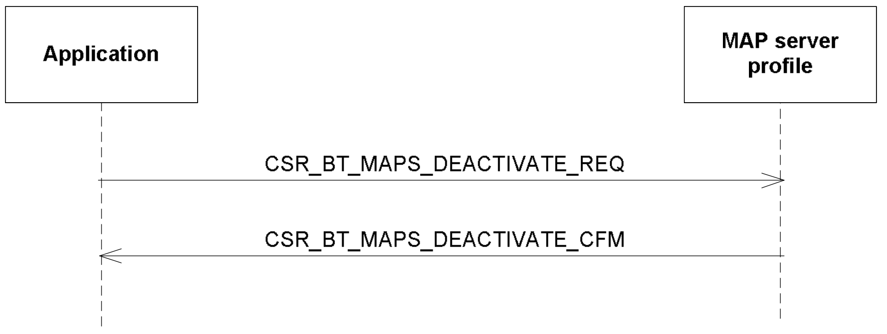


Figure 6: MAP server deactivation

After completing the deactivation process, the MAP server profile will send a CSR_BT_MAPS_DEACTIVATE_CFM to the application. The parameters for this primitive are found in Table 3.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_DEACTIVATE_CFM

Table 3: Parameters in a CSR_BT_MCAP_DEACTIVATE_CFM primitive

3.3 Cancelling Activation

If for any reason an application wants to cancel the activation process after sending a CSR_BT_MAPS_ACTIVATE_REQ but before receiving a CSR_BT_MAPS_ACTIVATE_CFM, this can be done by sending a CSR_BT_MAPS_DEACTIVATE_REQ as described in section 3.2. See Figure 7.

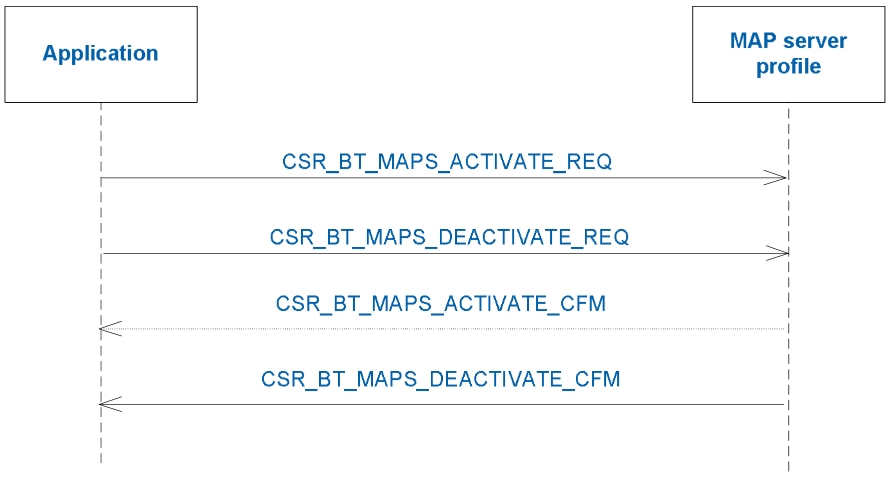


Figure 7: Cancelling MCAP activation

After cancelling the activation procedure a CSR_BT_MAPS_DEACTIVATE_CFM (i.e. not a CSR_BT_MAPS_ACTIVATE_CFM) will be received by the application. See section 3.2 for details.

In case of crossing signals, a CSR_BT_MAPS_ACTIVATE_CFM *may* be received by the application, but in that case the CSR_BT_MAPS_ACTIVATE_CFM can be ignored.

3.4 Setting Security Level

To set the security level of the MAP server application, it must send a CSR_BT_MAPS_SET_SECURITY_LEVEL_REQ to the MAPS profile by calling the `CsrBtMapsSetSecurityLevelReqSend` function. The arguments for the `CsrBtMapsSetSecurityLevelReqSend` function are found in Table 4.

Type	Argument	Description
CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_MAPS_SET_SECURITY_LEVEL_CFM is sent
dm_security_level_t	secLevel	Security level of the MAP server. The application must specify one of the following values: • CSR_BT_SEC_DEFAULT: Use default security settings • CSR_BT_SEC_MANDATORY: Use mandatory security settings • CSR_BT_SEC_SPECIFY: Specify new security settings If CSR_BT_SEC_SPECIFY is set the following values can be OR'ed additionally: • CSR_BT_SEC_AUTHORISATION: Require authorisation • CSR_BT_SEC_AUTHENTICATION: Require authentication • CSR_BT_SEC_ CSR_BT_SEC_ENCRYPTION: Require encryption (implies authentication) • CSR_BT_SEC_MITM: Require MITM protection (implies encryption)

Table 4: Arguments for `CsrBtMapsSetSecurityLevelReqSend` function

As depicted in Figure 8, the server application will, after requesting the security level to be set, receive a CSR_BT_MAPS_SET_SECURITY_LEVEL_CFM primitive. The parameters for this primitive is found in are found in Table 5.

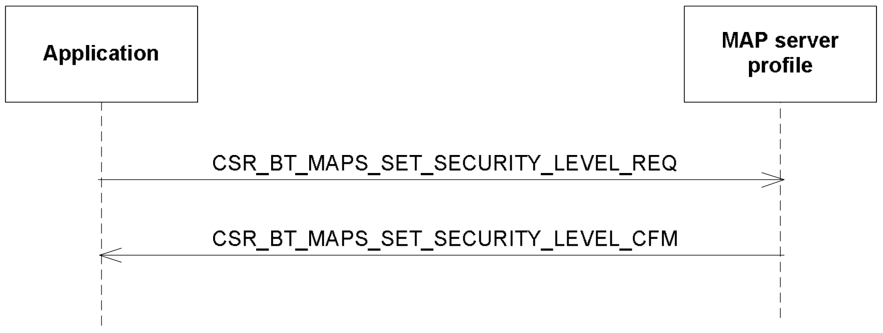

Application
MAP server profile
CSR_BT_MAPS_SET_SECURITY_LEVEL_REQ
CSR_BT_MAPS_SET_SECURITY_LEVEL_CFM

Figure 8: Setting security level of the MAP server

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_SET_SECURITY_LEVEL_CFM
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrBtResultCode	resultCode	The result code of the operation. Possible values depends on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_OBEX_PROFILES then the possible result codes can be found in csr_bt_obex_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 5: Parameters in a CSR_BT_MAPS_SET_SECURITY_LEVEL_CFM primitive

3.5 Connecting a MAS Channel

The MAP server application is not allowed to initiate an MAS connection. The MAS client is responsible for this. When an MAS client initiates a connection, this will be indicated to the MAS server application by sending a CSR_BT_MAPS_CONNECT_IND to it. See Figure 9.

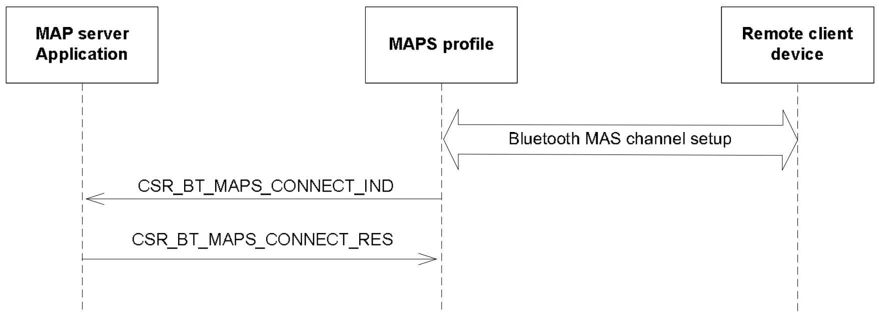


Figure 9: Connecting a MAS channel

The parameters for the CSR_BT_MAPS_CONNECT_IND primitive are found in Table 6.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_CONNECT_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrBtDeviceAddr	deviceAddr	Bluetooth address of the device sending the connect request
CsrUint32	length	The length parameter contains the length in bytes of the bodies of all the objects that the sender plans to send. Note this length cannot be guaranteed correct, so while the value may be useful for status indicators and resource reservations, the application should not die if the length is not correct.
CsrUint32	count	Count is used for indicating the number of objects that will be sent by the sender during this connection
CsrBtConnId	btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend`-function.

Table 6: Parameters in a CSR_BT_MAPS_CONNECT_IND primitive

After receiving a CSR_BT_MAPS_CONNECT_IND the application must respond by sending a CSR_BT_MAPS_CONNECT_RES. This is done by calling the `CsrBtMapsConnectResSend` function. The arguments for the `CsrBtMapsConnectResSend` function are found in Table 7.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the connection indication. Available error codes are found in the csr_bt_obex.h file. If this argument does not equal CSR_BT_RESULT_CODE_OBEX_SUCCESS, the connection will not be set up.

Table 7: Arguments for `CsrBtMapsConnectResSend` function

3.6 Disconnecting a MAS Channel

Since the MAP server application is not the initiator of the MAS channel, it is also not allowed to initiate the disconnection of the channel (except when the link between the client and the server is lost, but this scenario is handled by the lower CSR Synergy BT layers and does not require any actions by the MAP server application).

When the MAS client disconnects the MAS channel – or if the connection is lost due to radio link loss – it will be indicated to the MAP server application by sending a CSR_BT_MAPS_DISCONNECT_IND primitive to it. See Figure 10.

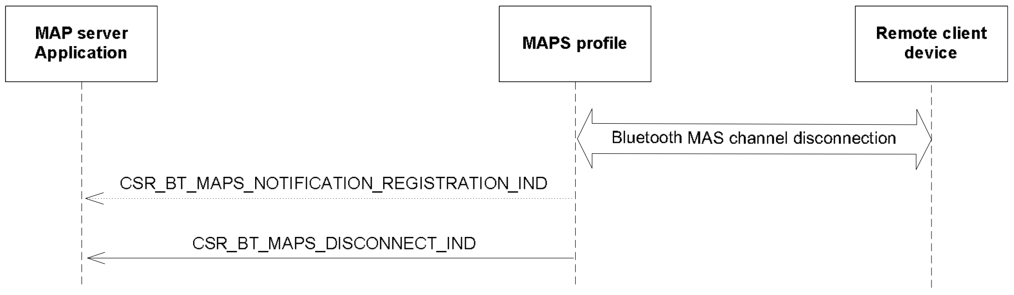

MAP server Application
MAPS profile
Remote client device
Bluetooth MAS channel disconnection
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND
CSR_BT_MAPS_DISCONNECT_IND

Figure 10: Disconnecting a MAS channel

The parameters for the CSR_BT_MAPS_DISCONNECT_IND primitive are found in Table 8.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_CONNECT_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrBtReasonCode	reasonCode	The reason code of the operation. Possible values depends on the value of reasonSupplier. If eg. 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 are the respective prim.h files or csr_bt_obex.h is regarded as reserved and the application should consider them as errors.
CsrBtSupplier	reasonSupplier	This parameter specifies the supplier of the reason given in reasonCode. Possible values can be found in csr_bt_result.h

Table 8: Parameters in a CSR_BT_MAPS_DISCONNECT_IND primitive

If by the time of disconnection occurring the MAS server subscribes to notification events, the MAP server application will receive a CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND primitive indicating that the subscription had ended. The parameters for this primitive are found in section 3.17 on page 28.

3.7 Connecting and disconnecting a MNS Channel

When the MAS client requests a subscription to notification events, the MAS server is responsible for initiating the Message Notification Service (MNS) channel. This is taken care of by the MAP server profile, i.e. no action is required by the server application to do this.

Disconnection of the MNS channel is also handled entirely by the MAP server profile.

3.8 Browsing: Set back folder

The MAS client may request the MAS server application to go back one folder level (in some operating systems this will equal to the command `cd..`) This is indicated to the MAS server application by sending a CSR_BT_MAPS_SET_BACK_FOLDER_IND to it. See Figure 11.

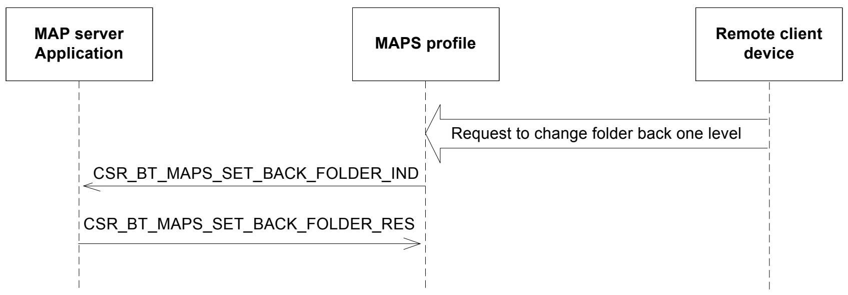

MAP server Application
MAPS profile
Remote client device
Request to change folder back one level
CSR_BT_MAPS_SET_BACK_FOLDER_IND
CSR_BT_MAPS_SET_BACK_FOLDER_RES

Figure 11: Browsing back one folder

The parameters for this primitive are found in Table 9.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_SET_BACK_FOLDER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUcs2ByteString	*folderName	Pointer to a NULL-terminated string, which contains the folder name. If the string length is 0, the folder should be changed to the parent folder only, i.e. no further folder changing should take place.

Table 9: Parameters in a CSR_BT_MAPS_SET_BACK_FOLDER_IND primitive

After receiving a CSR_BT_MAPS_SET_BACK_FOLDER_IND the application must respond by sending a CSR_BT_MAPS_SET_BACK_FOLDER_RES. This is done by calling the `CsrBtMapsSetBackFolderResSend` function. The arguments for the `CsrBtMapsSetBackFolderResSend` function are found in Table 10.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the set back folder indication. Available error codes are found in the csr_bt_obex.h file.

Table 10: Arguments for `CsrBtMapsSetBackFolderResSend` function

3.9 Browsing: Set root folder

The MAS client may request the MAS server application to go the root folder. This is indicated to the MAS server application by sending a CSR_BT_MAPS_SET_ROOT_FOLDER_IND to it. See Figure 12.

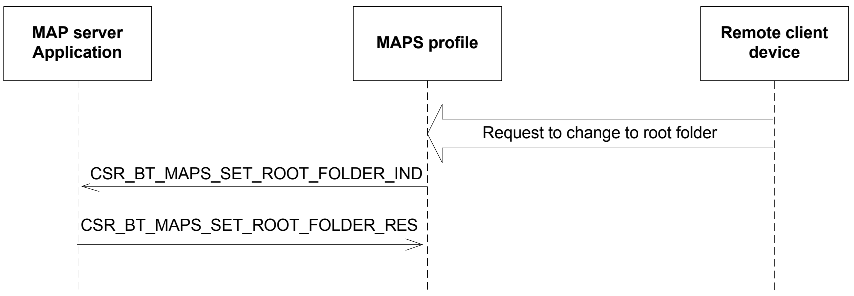

MAP server Application
MAPS profile
Remote client device
Request to change to root folder
CSR_BT_MAPS_SET_ROOT_FOLDER_IND
CSR_BT_MAPS_SET_ROOT_FOLDER_RES

Figure 12: Browsing to root folder

The parameters for this primitive are found in Table 11.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_SET_ROOT_FOLDER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event

Table 11: Parameters in a CSR_BT_MAPS_SET_ROOT_FOLDER_IND primitive

After receiving a CSR_BT_MAPS_SET_ROOT_FOLDER_IND the application must respond by sending a CSR_BT_MAPS_SET_ROOT_FOLDER_RES. This is done by calling the `CsrBtMapsSetRootFolderResSend` function. The arguments for the `CsrBtMapsSetRootFolderResSend` function are found in Table 12.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the set root folder indication. Available error codes are found in the csr_bt_obex.h file.

Table 12: Arguments for `CsrBtMapsSetRootFolderResSend` function

3.10 Browsing: Set folder

The MAS client may request the MAS server application to go to a specific folder. This is indicated to the MAS server application by sending a CSR_BT_MAPS_SET_FOLDER_IND to it. See Figure 13.

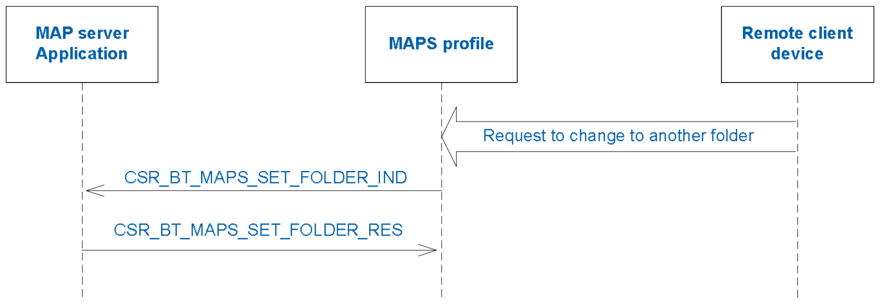

MAP server Application
MAPS profile
Remote client device
Request to change to another folder
CSR_BT_MAPS_SET_FOLDER_IND
CSR_BT_MAPS_SET_FOLDER_RES

Figure 13: Browsing: Set folder

The parameters for this primitive are found in Table 13.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_SET_FOLDER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUcs2ByteString	*folderName	Pointer to a NULL-terminated string, which contains the folder name

Table 13: Parameters in a CSR_BT_MAPS_SET_FOLDER_IND primitive

After receiving a CSR_BT_MAPS_SET_FOLDER_IND the application must respond by sending a CSR_BT_MAPS_SET_FOLDER_RES. This is done by calling the `CsrBtMapsSetFolderResSend` function. The arguments for the `CsrBtMapsSetFolderResSend` function are found in Table 14.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the set folder indication. Available error codes are found in the csr_bt_obex.h file.

Table 14: Arguments for `CsrBtMapsSetFolderResSend` function

3.11 Get: Folder Listing

If the MAS client requests a list of folders within the current folder it is indicated to the MAS server application by first sending a CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_IND primitive to it. See Figure 14.

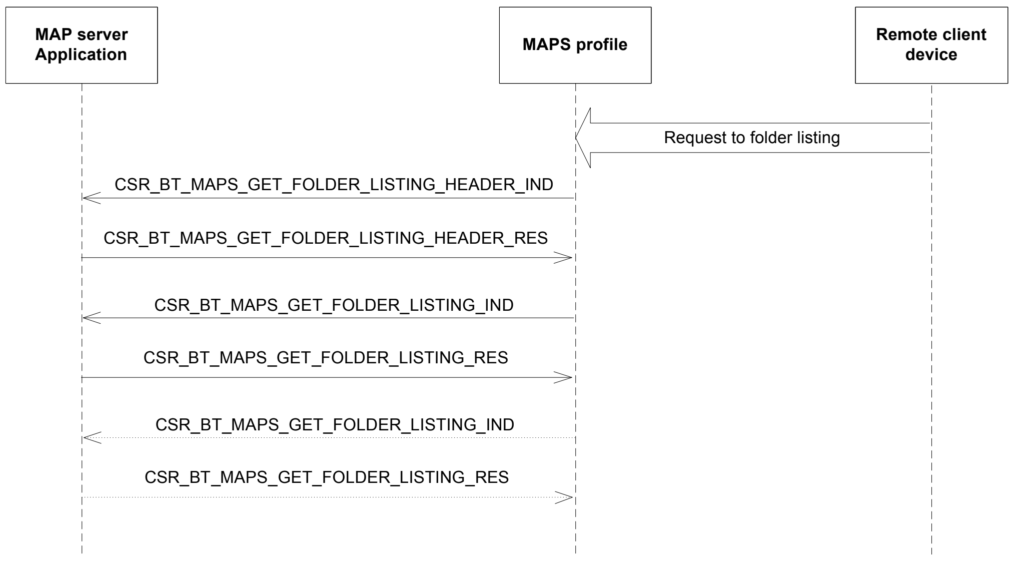

MAP server Application
MAPS profile
Remote client device
Request to folder listing
CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_IND
CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_RES
CSR_BT_MAPS_GET_FOLDER_LISTING_IND
CSR_BT_MAPS_GET_FOLDER_LISTING_RES
CSR_BT_MAPS_GET_FOLDER_LISTING_IND
CSR_BT_MAPS_GET_FOLDER_LISTING_RES

Figure 14: Get folder listing

The parameters for this primitive are found in Table 15.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUint16	maxListCount	Maximum number of folders in the listing. If this parameter is not included in the received OBEX packet, the value is set to 0
CsrUint16	listStartOffset	Offset of where to start the listing. If this parameter is not included in the received OBEX packet, the value is set to 0

Table 15: Parameters in a CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_IND primitive

After receiving a CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_IND the application must respond by sending a CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_RES. This is done by calling the `CsrBtMapsGetFolderListingHeaderResSend` function. The arguments for the `CsrBtMapsGetFolderListingHeaderResSend` function are found in Table 16.

Type	Argument	Description
CsrUint16	fullFolderListingSize	Number of folders in the complete folder listing
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 16: Arguments for `CsrBtMapsGetFolderListingHeaderResSend` function

After sending the CSR_BT_MAPS_GET_FOLDER_LISTING_HEADER_RES the application will receive a CSR_BT_MAPS_GET_FOLDER_LISTING_IND primitive, which will allow the application send the actual listing. The parameters for this primitive are found in Table 16.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_GET_FOLDER_LISTING_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUint16	obexResponsePacketLength	Maximum allowed number of bytes in the corresponding response

Table 17: Parameters in a CSR_BT_MAPS_GET_FOLDER_IND primitive

After receiving a CSR_BT_MAPS_GET_FOLDER_LISTING_IND the application must respond by sending a CSR_BT_MAPS_GET_FOLDER_LISTING_RES containing the actual listing. This is done by calling the `CsrBtMapsGetFolderListingResSend` function. The arguments for the `CsrBtMapsGetFolderListingResSend` function are found in Table 18.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the set folder indication. Available error codes are found in the csr_bt_obex.h file.
CsrUint16	bodyLength	Length of the body being sent
CsrUint8	*body	Pointer to the body data
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 18: Arguments for `CsrBtMapsGetFolderListingResSend` function

If the listing is too large to fit into one OBEX packet the application will receive more than one CSR_BT_MAPS_GET_FOLDER_LISTING_IND primitive. These “extra” primitives must also be responded to by calling the `CsrBtMapsGetFolderListingResSend` function.

3.12 Get: Message Listing

If the MAS client requests a list of messages within the current folder it is indicated to the MAS server application by first sending a CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND primitive to it. See Figure 15.

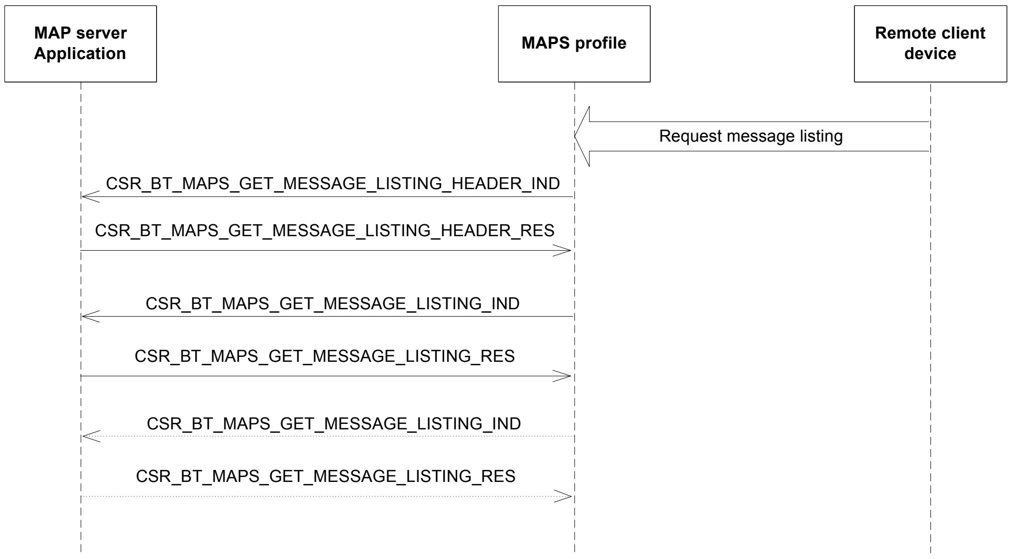

MAP server Application
MAPS profile
Remote client device
Request message listing
CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_RES
CSR_BT_MAPS_GET_MESSAGE_LISTING_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_RES
CSR_BT_MAPS_GET_MESSAGE_LISTING_IND
CSR_BT_MAPS_GET_MESSAGE_LISTING_RES

Figure 15: Get message listing

The parameters for this primitive are found in Table 19.

Type	Argument	Description
CsrBtMapsPrim	Type	Signal identity – always set to CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUcs2ByteString	*folderName	Pointer to a NULL-terminated string, which contains the folder name. If the string length is 0, the listing for the current folder is desired.
CsrUint16	maxListCount	Maximum number of folders in the listing. If this parameter is not included in the received OBEX packet, the value is set to 0
CsrUint16	listStartOffset	Offset of where to start the listing. If this parameter is not included in the received OBEX packet, the value is set to 0
CsrUint8	maxSubjectLength	Maximum string length of the subject field. If this parameter is not included in the received OBEX packet, the value is set to 0
CsrBtMapMesParms	parameterMask	Bitmask of the parameters relevant for the message listing. A bit value of 1 means that the parameter should be present, and a bit value of 0 means that the parameter must be filtered out. If this parameter is not included in the received OBEX packet, all bits are set to 0. For details of the different parameters, refer to section 5.5.4 of [1]
CsrBtMapMesTypes	filterMessageType	Bitmask specifying which message type(s) to include in the listing. A bit value of 1 means that the message type should be present, and a bit value of 0 means that the message type must be filtered out. If this parameter is not included in the received OBEX packet, all bits are set to 0 Valid types are: CSR_BT_MAP_TYPE_NO_SMS_GSM, CSR_BT_MAP_TYPE_NO_SMS_CDMA, CSR_BT_MAP_TYPE_NO_EMAIL and CSR_BT_MAP_TYPE_NO_MMS

Type	Argument	Description
CsrUtf8String	*filterPeriodBegin	NULL-terminated time string. If this parameter is not included in the received OBEX packet, the value is set to NULL The parameter shall be formatted to "YYYYMMDDTHHMMSS"-format
CsrUtf8String	*filterPeriodEnd	NULL-terminated time string. If this parameter is not included in the received OBEX packet, the value is set to NULL The parameter shall be formatted to "YYYYMMDDTHHMMSS"-format
CsrBtMapReadStatus	filterReadStatus	Bitmask specifying which message type(s) to include in the listing should be done on behalf of the Read status. If this parameter is not included in the received OBEX packet, all bits are set to 0. Valid values are: CSR_BT_MAP_STATUS_NO_FILTERING, CSR_BT_MAP_STATUS_UNREAD and CSR_BT_MAP_STATUS_READ.
CsrUtf8String	*filterRecipient	NULL-terminated recipient string. If this parameter is not included in the received OBEX packet, the value is set to NULL For details of the different parameters, refer to section 5.5.4 of [1]
CsrUtf8String	*filterOriginator	NULL-terminated originator string. If this parameter is not included in the received OBEX packet, the value is set to NULL For details of the different parameters, refer to section 5.5.4 of [1]
CsrBtMapPriority	filterPriority	Bitmask specifying which priority type(s) to include in the listing. If this parameter is not included in the received OBEX packet, all bits are set to 0. Valid values are: CSR_BT_MAP_PRIORITY_NO_FILTERING, CSR_BT_MAP_PRIORITY_HIGH and CSR_BT_MAP_PRIORITY_NON_HIGH.

Table 19: Parameters in a CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND primitive

After receiving a CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND the application must respond by sending a CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_RES. This is done by calling the `CsrBtMapsGetMessageListingHeaderResSend` function. The arguments for the `CsrBtMapsGetMessageListingHeaderResSend` function are found in Table 20.

Type	Argument	Description
CsrBtMapNewMessage	newMessages	Specifying if there are unread messages on the MAS server or not. Valid values are: CSR_BT_MAP_NEW_MESSAGE_OFF and CSR_BT_MAP_NEW_MESSAGE_ON.
CsrUtf8String	*mseTime	NULL-terminated time string. For details of the different parameters, refer to section 5.5.4 of [1]
CsrUint16	fullFolderListingSize	Number of folders in the complete folder listing
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 20: Arguments for `CsrBtMapsGetMessageListingHeaderResSend` function

After sending the CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_RES the application will receive a CSR_BT_MAPS_GET_MESSAGE_LISTING_IND primitive, which will allow the application send the actual listing. The parameters for this primitive are found in Table 21.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_GET_MESSAGE_LISTING_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUint16	obexResponsePacketLength	Maximum allowed number of bytes in the corresponding response

Table 21: Parameters in a CSR_BT_MAPS_GET_MESSAGE_IND primitive

After receiving a CSR_BT_MAPS_GET_MESSAGE_LISTING_IND the application must respond by sending a CSR_BT_MAPS_GET_MESSAGE_LISTING_RES containing the actual listing. This is done by calling the `CsrBtMapsGetMessageListingResSend` function. The arguments for the `CsrBtMapsGetMessageListingResSend` function are found in Table 22.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.
CsrUint16	bodyLength	Length of the body being sent
CsrUint8	*body	Pointer to the body data
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 22: Arguments for `CsrBtMapsGetMessageListingResSend` function

If the listing is too large to fit into one OBEX packet the application will receive more than one CSR_BT_MAPS_GET_MESSAGE_LISTING_IND primitive. These “extra” primitives must also be responded to by calling the `CsrBtMapsGetMessageListingResSend` function.

3.13 Get: Message

If the MAS client requests a message within the current folder it is indicated to the MAS server application by first sending a CSR_BT_MAPS_GET_MESSAGE_HEADER_IND primitive to it. See Figure 16.

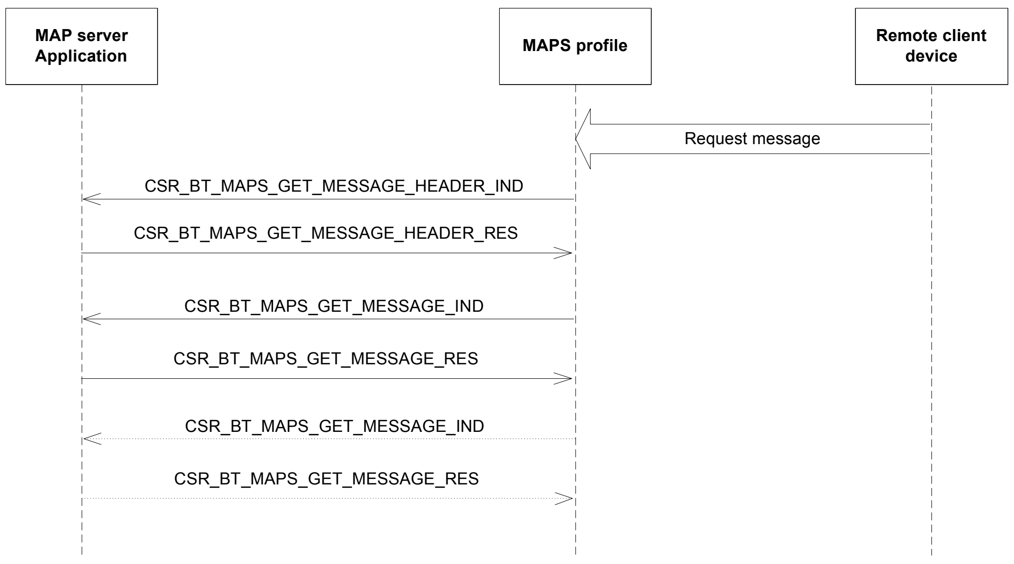

MAP server Application
MAPS profile
Remote client device
Request message
CSR_BT_MAPS_GET_MESSAGE_HEADER_IND
CSR_BT_MAPS_GET_MESSAGE_HEADER_RES
CSR_BT_MAPS_GET_MESSAGE_IND
CSR_BT_MAPS_GET_MESSAGE_RES
CSR_BT_MAPS_GET_MESSAGE_IND
CSR_BT_MAPS_GET_MESSAGE_RES

Figure 16: Get message

The parameters for this primitive are found in Table 23.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_GET_MESSAGE_HEADER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrCharString	*messageHandle	Pointer to a NULL-terminated string, which contains the message handle
CsrBtMapAttachment	attachment	Bitmap specifying if attachments should be included or not. Valid values are: CSR_BT_MAP_ATTACHMENT_OFF and CSR_BT_MAP_ATTACHMENT_ON.
CsrBtMapCharset	charset	Bitmask specifying which charset to use for the message. Valid values are: CSR_BT_MAP_CHARSET_NATIVE and CSR_BT_MAP_CHARSET_UTF8.
CsrBtMapFracReq	fractionRequest	Bitmask specifying which fragment of the message to get (if any). If this parameter is not included in the received OBEX packet, all bits are set to 0. Valid values are: CSR_BT_MAP_FRACTION_REQ_NOT_USED, CSR_BT_MAP_FRACTION_REQ_FIRST and CSR_BT_MAP_FRACTION_REQ_NEXT.

Table 23: Parameters in a CSR_BT_MAPS_GET_MESSAGE_HEADER_IND primitive

After receiving a CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_IND the application must respond by sending a CSR_BT_MAPS_GET_MESSAGE_LISTING_HEADER_RES. This is done by calling the `CsrBtMapsGetMessageHeaderResSend` function. The arguments for the `CsrBtMapsGetMessageHeaderResSend` function are found in Table 24.

Type	Argument	Description
CsrBtMapFracReq	fractionRequest	Bitmask specifying which fragment of the message is sent (if any)
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 24: Arguments for `CsrBtMapsGetMessageHeaderResSend` function

After sending the CSR_BT_MAPS_GET_MESSAGE_HEADER_RES the application will receive a CSR_BT_MAPS_GET_MESSAGE_IND primitive, which will allow the application send the actual message. The parameters for this primitive are found in Table 25.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_GET_MESSAGE_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUint16	obexResponsePacketLength	Maximum allowed number of bytes in the corresponding response

Table 25: Parameters in a CSR_BT_MAPS_GET_MESSAGE_IND primitive

After receiving a CSR_BT_MAPS_GET_MESSAGE_LISTING_IND the application must respond by sending a CSR_BT_MAPS_GET_MESSAGE_LISTING_RES containing the actual listing. This is done by calling the `CsrBtMapsGetMessageResSend` function. The arguments for the `CsrBtMapsGetMessageResSend` function are found in Table 26.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.
CsrUint16	bodyLength	Length of the body being sent
CsrUint8	*body	Pointer to the body data
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 26: Arguments for `CsrBtMapsGetMessageResSend` function

If the listing is too large to fit into one OBEX packet the application will receive more than one CSR_BT_MAPS_GET_MESSAGE_IND primitive. These "extra" primitives must also be responded to by calling the `CsrBtMapsGetMessageResSend` function.

3.14 Set Message Status

If the MAS client requests to modify a status indicator for a message within the current folder it is indicated to the MAS server application by sending a CSR_BT_MAPS_SET_MESSAGE_STATUS_IND primitive to it. See Figure 17.

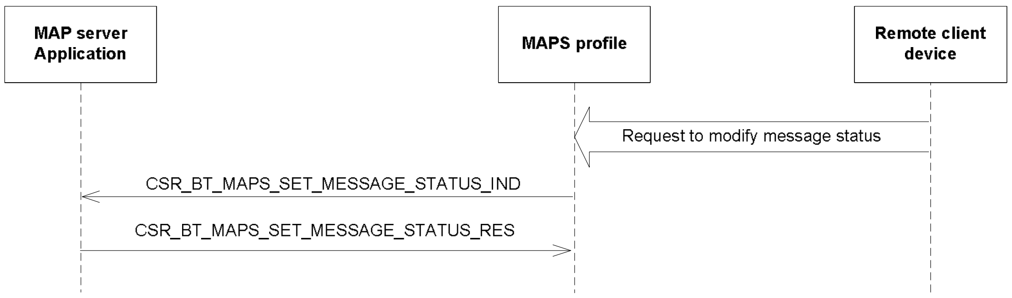

MAP server Application
MAPS profile
Remote client device
Request to modify message status
CSR_BT_MAPS_SET_MESSAGE_STATUS_IND
CSR_BT_MAPS_SET_MESSAGE_STATUS_RES

Figure 17: Set message status

The parameters for this primitive are found in Table 27.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_SET_MESSAGE_STATUS_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUtf8String	*messageHandle	Pointer to a NULL-terminated string, which contains the message handle
CsrBtMapStatusInd	statusIndicator	Bitmap specifying which status indicator to modify. Valid values are: CSR_BT_MAP_STATUS_IND_READ and CSR_BT_MAP_STATUS_IND_DELETE.
CsrBtMapStatusVal	statusValue	Specifies new value of the status indicator selected. Valid values are: CSR_BT_MAP_STATUS_VAL_NO and CSR_BT_MAP_STATUS_VAL_YES.

Table 27: Parameters in a CSR_BT_MAPS_SET_MESSAGE_STATUS_IND primitive

After receiving a CSR_BT_MAPS_SET_MESSAGE_STATUS_IND the application must respond by sending a CSR_BT_MAPS_SET_MESSAGE_STATUS_RES. This is done by calling the `CsrBtMapsSetMessageStatusResSend` function. The arguments for the `CsrBtMapsSetMessageStatusResSend` function are found in Table 28.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 28: Arguments for `CsrBtMapsSetMessageStatusResSend` function

3.15 Push Message

If the MAS client requests to push a message to the current folder it is indicated to the MAS server application by first sending a CSR_BT_MAPS_PUSH_MESSAGE_HEADER_IND primitive to it. See Figure 18.

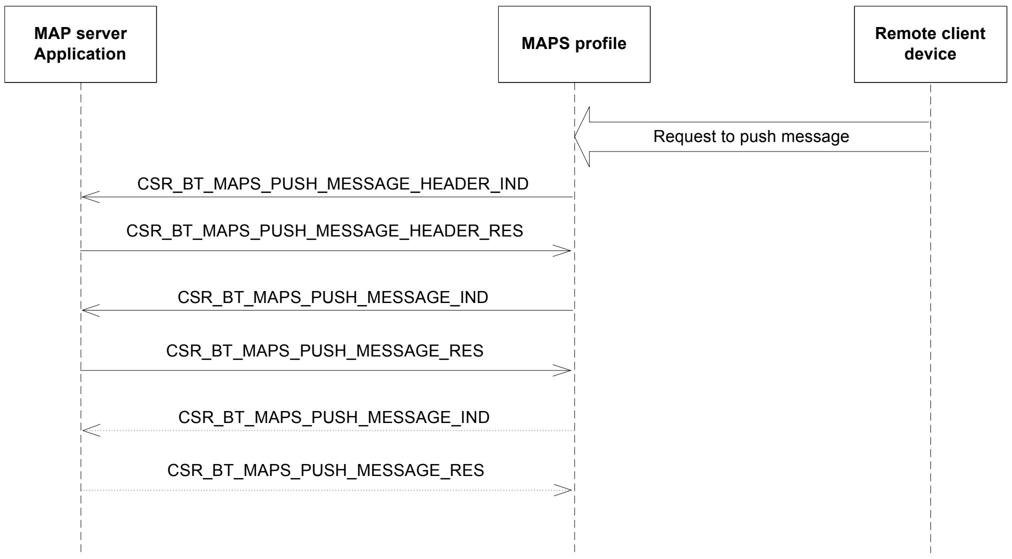

MAP server Application
MAPS profile
Remote client device
Request to push message
CSR_BT_MAPS_PUSH_MESSAGE_HEADER_IND
CSR_BT_MAPS_PUSH_MESSAGE_HEADER_RES
CSR_BT_MAPS_PUSH_MESSAGE_IND
CSR_BT_MAPS_PUSH_MESSAGE_RES
CSR_BT_MAPS_PUSH_MESSAGE_IND
CSR_BT_MAPS_PUSH_MESSAGE_RES

Figure 18: Push message

The parameters for this primitive are found in Table 29.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_PUSH_MESSAGE_HEADER_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUcs2String	*folderName	Pointer to a NULL-terminated string, which contains the folder name. If the string length is 0, the message should be pushed to the current folder.
CsrBtMapTrans	transparent	Indicating if a copy of the message should be kept in the "Sent"-folder. If the value is 1, no copies should be kept. If this parameter is not included in the received OBEX packet, the value is set to 0. Valid values are: CSR_BT_MAP_TRANSPARENT_NOT_SPECIFIED, CSR_BT_MAP_TRANSPARENT_OFF and CSR_BT_MAP_TRANSPARENT_ON
CsrBtMapRetry	retry	Specifies if the MSE should try to resent the message, if the first delivery to the network fails. Valid values are: CSR_BT_MAP_RETRY_NOT_SPECIFIED, CSR_BT_MAP_RETRY_OFF and CSR_BT_MAP_RETRY_ON.

Type	Argument	Description
CsrBtMapCharset	charset	Bitmask specifying which charset to use for the message Valid values are: CSR_BT_MAP_CHARSET_NATIVE and CSR_BT_MAP_CHARSET_UTF8.

Table 29: Parameters in a CSR_BT_MAPS_PUSH_MESSAGE_HEADER_IND primitive

After receiving a CSR_BT_MAPS_PUSH_MESSAGE_LISTING_HEADER_IND the application must respond by sending a CSR_BT_MAPS_PUSH_MESSAGE_LISTING_HEADER_RES. This is done by calling the `CsrBtMapsPushMessageHeaderResSend` function. The arguments for the `CsrBtMapsPushMessageHeaderResSend` function are found in Table 30.

Type	Argument	Description
CsrCharString	*messageHandle	Pointer to a NULL-terminated ASCII string, with16 hexadecimal digits which representing the assigned handle to the receiving message. Note if the application cannot assign a message handle until the whole message is receive, it must set the 'messageHandle' parameter to NULL and used the function CsrBtMapsPushMessageFinalResSend when responding the last packet, seeTable 33
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 30: Arguments for `CsrBtMapsPushMessageHeaderResSend` function

After sending the CSR_BT_MAPS_PUSH_MESSAGE_HEADER_RES the application will receive a CSR_BT_MAPS_PUSH_MESSAGE_IND primitive, which will contain the actual message body. The parameters for this primitive are found in Table 31.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_PUSH_MESSAGE_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrBool	finalFlag	Indicates if this is the last packet. If set to TRUE the current packet is the last packet
CsrUint16	bodyLength	Length of the body being sent
CsrUint8	*body	Pointer to the body data

Table 31: Parameters in a CSR_BT_MAPS_PUSH_MESSAGE_IND primitive

After receiving a CSR_BT_MAPS_PUSH_MESSAGE_LISTING_IND the application must respond by sending a CSR_BT_MAPS_PUSH_MESSAGE_LISTING_RES containing the actual listing. This is done by calling either `CsrBtMapsPushMessageResSend or CsrBtMapsPushMessageFinalResSend`. The arguments for the `CsrBtMapsPushMessageResSend` function are found in Table 32 and the arguments for `CsrBtMapsPushMessageFinalResSend are found in` Table 33.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 32: Arguments for `CsrBtMapsPushMessageResSend` function

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.
CsrBool	srmpOn	Reserved for future use – Set to FALSE
CsrCharString	*messageHandle	Pointer to a NULL-terminated ASCII string, with16 hexadecimal digits which representing the assigned handle to the receiving message. This function may be use by an application, which cannot assign a message handle until the whole message is receive. E.g. if the 'messageHandle' parameter in the function `CsrBtMapsPushMessageHeaderResSend` is set to NULL then the application must use the `CsrBtMapsPushMessageFinalResSend when responding the last packet i.e. when the` 'responseCode' parameter is set to `CSR_BT_OBEX_SUCCESS_RESPONSE_CODE`

Table 33: Arguments for `CsrBtMapsPushMessageFinalResSend` function

If the listing is too large to fit into one OBEX packet the application will receive more than one CSR_BT_MAPS_PUSH_MESSAGE_IND primitive. These "extra" primitives must also be responded to by calling the `CsrBtMapsPushMessageResSend or the CsrBtMapsPushMessageFinalResSend` function.

3.16 Update Inbox

If the MAS client requests the MAS server to update its inbox, it is indicated to the MAS server application by sending a CSR_BT_MAPS_UPDATE_INBOX_IND primitive to it. See Figure 19

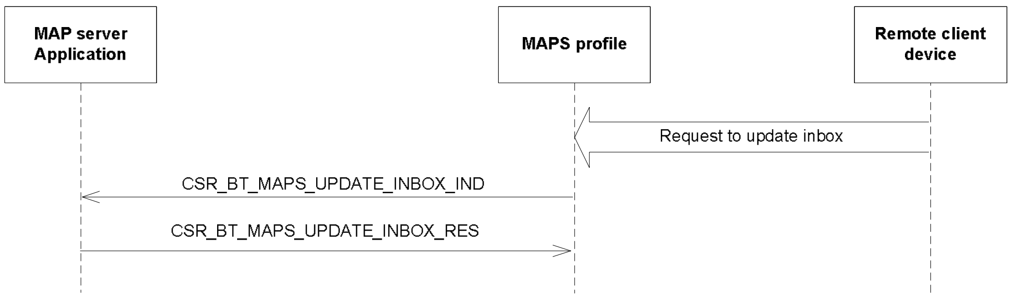


Figure 19: Update inbox

The parameters for this primitive are found in Table 34.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_UPDATE_INBOX_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event

Table 34: Parameters in a CSR_BT_MAPS_UPDATE_INBOX_IND primitive

After receiving a CSR_BT_MAPS_UPDATE_INBOX_IND the application must respond by sending a CSR_BT_MAPS_UPDATE_INBOX_RES. This is done by calling the `CsrBtMapsUpdateInboxResSend` function. The arguments for the `CsrBtMapsUpdateInboxResSend` function are found in Table 35.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.

Type	Argument	Description
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 35: Arguments for `CsrBtMapsUpdateInboxResSend` function

3.17 Notification Registration

If the MAS client requests to subscribe to notification events from the MAS server/MNS client, it is indicated to the MAS server application by sending a CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND primitive to it. See Figure 20.

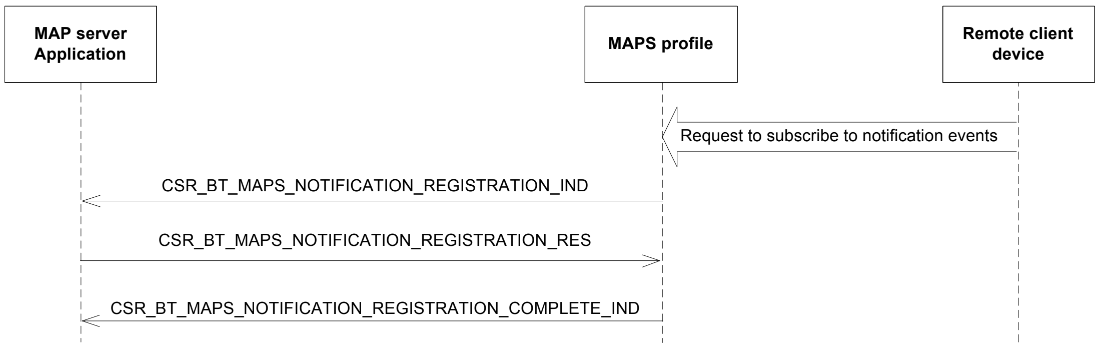

MAP server Application
MAPS profile
Remote client device
Request to subscribe to notification events
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_RES
CSR_BT_MAPS_NOTIFICATION_REGISTRATION_COMPLETE_IND

Figure 20: Notification registration

The parameters for this primitive are found in Table 36.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_NOTIFICATION_REGISTRATION _IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrBool	notificationStatus	Boolean specifying of notification events should be switched on or off. If set to TRUE, subscription to notification event is on. If set to FALSE, subscription is off. NOTE: The MAS server/MNS client is not allowed to send notification events to the MAS client/MNS server before a CSR_BT_NOTIFICATION_REGISTRATION_COMPLETE_IND primitive is received.

Table 36: Parameters in a CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND primitive

After receiving a CSR_BT_MAPS_NOTIFICATION_REGISTRATION_IND the application must respond by sending a CSR_BT_MAPS_NOTIFICATION_REGISTRATION_RES. This is done by calling the `CsrBtMapsNotificationRegistrationResSend` function. The arguments for the `CsrBtMapsNotificationRegistrationResSend` function are found in Table 37.

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file.
CsrBool	srmpOn	Reserved for future use – Set to FALSE

Table 37: Arguments for `CsrBtMapsNotificationRegistrationResSend` function

If the responseCode of the CSR_BT_NOTIFICATION_REGISTRATION_RES primitive equals CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, the MNS channel is setup by the MAP server profile without any further actions required by the MAP server application. When this process has finished, the MAP server profile will send a CSR_BT_NOTIFICATION_REGISTRATION_COMPLETE_IND primitive to the MAP server application. If the resultCode of this primitive equals CSR_BT_RESULT_CODE_OBEX_SUCCESS and the resultSupplier equals CSR_BT_SUPPLIER_OBEX_PROFILES, the MAP server application is allowed to send notification events to the peer device.

If the responseCode of the CSR_BT_NOTIFICATION_REGISTRATION_COMPLETE_IND primitive does not equal CSR_BT_RESULT_CODE_OBEX_SUCCESS or the resultSupplier does not equal CSR_BT_SUPPLIER_OBEX_PROFILES, the MAP server application is not allowed to send notification events. Note that in this case it is not possible for the MAP server application to try to re-initiate a MNS connection due to MAP specification restrictions.

The parameters for the CSR_BT_NOTIFICATION_REGISTRATION_COMPLETE_IND primitive are found in Table 38.

Type	Argument	Description
CsrBtMapsPrim	Type	Signal identity – always set to CSR_BT_MAPS_NOTIFICATION_REGISTRATION_COMPLETE_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrBtResultCode	resultCode	The result code of the operation. Possible values depends on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_OBEX_PROFILES then the possible result codes can be found in csr_bt_obex_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 38: Parameters in a CSR_BT_MAPS_NOTIFICATION_REGISTRATION_COMPLETE_IND primitive

3.18 Abort

If the MAS client requests the MAS server to abort the current operation, it is indicated to the MAS server application by sending a CSR_BT_MAPS_ABORT_IND primitive to it. See Figure 21.

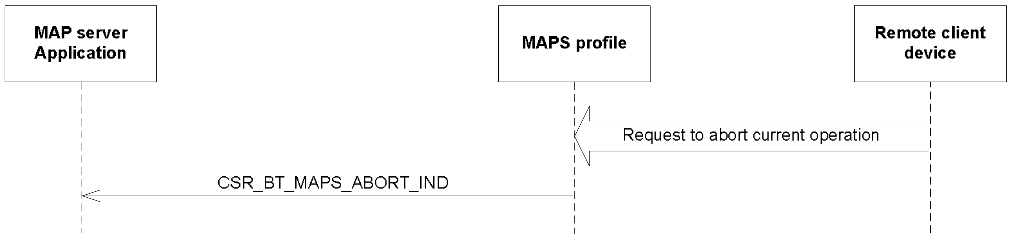


Figure 21: Abort

The parameters for this primitive are found in Table 39.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_ABORT_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUint16	descriptionOffset	Description offset relative to the payload pointer
CsrUint16	descriptionLength	Length of description
data_ptr_t	*payload	Pointer to payload data
CsrUint16	payloadLength	Length of payload

Table 39: Parameters in a CSR_BT_MAPS_ABORT_IND primitive

3.19 Event Notification

If the MAS server/MNS client needs to inform the MAS client/MNS server about an event occurring, it can do so by sending a CSR_BT_MAPS_EVENT_NOTIFICATION_REQ primitive to the MAPS server profile. This is done by calling the `CsrBtMapsEventNotificationReqSend` function.

NOTE: The application is not allowed to send a MAPS_EVENT_NOTIFICATION_REQ before it has received a MAPS_EVENT_NOTIFICATION_CFM for the previous one.

The arguments for the `CsrBtMapsEventNotificationReqSend` function are found in Table 40.

Type	Argument	Description
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event

Table 40: Arguments for `CsrBtMapsEventNotificationReqSend` function

After sending a CSR_BT_EVENT_NOTIFICATION_REQ the MAP server application will receive a CSR_BT_EVENT_NOTIFICATION_IND. See Figure 22. The parameters for the CSR_BT_EVENT_NOTIFICATION_IND primitive are found in Table 41.

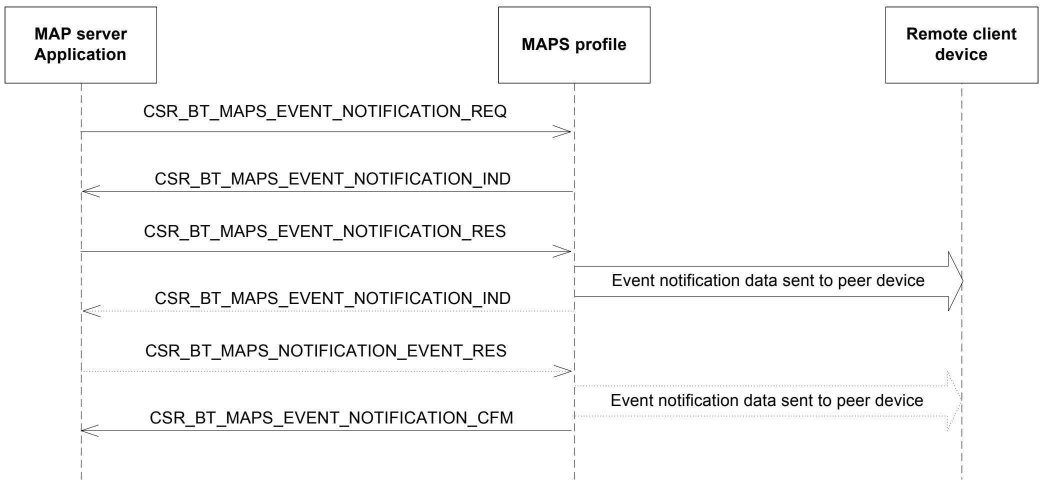

MAP server Application
MAPS profile
Remote client device
CSR_BT_MAPS_EVENT_NOTIFICATION_REQ
CSR_BT_MAPS_EVENT_NOTIFICATION_IND
CSR_BT_MAPS_EVENT_NOTIFICATION_RES
Event notification data sent to peer device
CSR_BT_MAPS_EVENT_NOTIFICATION_IND
CSR_BT_MAPS_NOTIFICATION_EVENT_RES
Event notification data sent to peer device
CSR_BT_MAPS_EVENT_NOTIFICATION_CFM

Figure 22: Event notification

Type	Argument	Description
CsrBtMapsPrim	Type	Signal identity – always set to CSR_BT_MAPS_EVENT_NOTIFICATION_IND
CsrSchedQid	instanceId	Instance identification for the MAS instance which created this event
CsrUint16	obexResponsePacketLength	Maximum allowed number of bytes in the corresponding response

Table 41: Parameters in a CSR_BT_MAPS_EVENT_NOTIFICATION_IND primitive

The CSR_BT_EVENT_NOTIFICATION_IND primitive must be responded to by sending a CSR_BT_MAPS_EVENT_NOTIFICATION_RES. This is done by calling the `CsrBtMapsEventNotificationResSend` function. The arguments for this function are found in Table 42.

Type	Argument	Description

Type	Argument	Description
CsrBtObexResponseCode	responseCode	OBEX response to the get message listing indication. Available error codes are found in the csr_bt_obex.h file. If this is the last data packet, this argument must be set to CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. If more packets are coming, this parameter must be set to CSR_BT_OBEX_CONTINUE_RESPONSE_CODE. If another responseCode is used, it will be interpreted as an error by the MAP server profile, and the current event notification operation will be aborted.
CsrUint16	bodyLength	Length of the body being sent
CsrUint8	*body	Pointer to the body data

Table 42: Arguments for `CsrBtMapsEventNotificationResSend` function

If the listing is too large to fit into one OBEX packet, i.e. the responseCode of the CSR_BT_EVENT_NOTIFICATION_RES is set to CSR_BT_OBEX_CONTINUE_RESPONSE_CODE, the application will receive more than one CSR_BT_MAPS_EVENT_NOTIFICATION_IND primitive. These “extra” primitives must also be responded to by calling the `CsrBtMapsEventNotificationResSend` function.

After sending a CSR_BT_MAPS_EVENT_NOTIFICATION_RES with a responseCode equalling CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, the application will receive a CSR_BT_MAPS_EVENT_NOTIFICATION_CFM primitive. The parameters for this primitive are found in Table 43.

Type	Argument	Description
CsrBtMapsPrim	type	Signal identity – always set to CSR_BT_MAPS_EVENT_NOTIFICATION_CFM
CsrBtResultCode	resultCode	The result code of the operation. Possible values depends on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_OBEX_PROFILES then the possible result codes can be found in csr_bt_obex_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 43: Parameters in a CSR_BT_MAPS_EVENT_NOTIFICATION_CFM primitive

Note that if the event notification operation was aborted by the MAP server application, i.e. the responseCode of the CSR_BT_EVENT_NOTIFICATION_RES was not set to CSR_BT_OBEX_SUCCESS_RESPONSE_CODE or CSR_BT_OBEX_CONTINUE_RESPONSE_CODE, the resultSupplier of the CSR_BT_EVENT_NOTIFICATION_CFM primitive is set to CSR_BT_SUPPLIER_OBEX_PROFILES and the resultCode is set to CSR_BT_RESULT_CODE_OBEX_ABORTED_BY_LOCAL_DEVICE.

4 Document References

Document	Reference
MESSAGE ACCESS PROFILE specification v. 10r00, dated 04 June. 2009. Available for download at www.bluetooth.org	[1]

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
HCI	Host Controller Interface
L2CAP	Logical Link Control and Adaption Protocol
MAP	Message Access Profile
MAPS	Message Access Profile Server
MAS	Message Access Service
MNS	Message Notification Service
OBEX	OBject EXchange
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

TradeMarks, Patents and Licences

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

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