



Message Access Profile (MAP)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
January 10, 2014



Bluetooth and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc., USA and licensed to Stonestreet One, LLC. Bluetopia[®], Stonestreet One[™], and the Stonestreet One logo are registered trademarks of Stonestreet One LLC, Louisville, Kentucky, USA. All other trademarks are property of their respective owners.
Copyright © 2000-2014 by Stonestreet One, LLC. All rights reserved.

Table of Contents

1. INTRODUCTION.....	4
1.1 Scope	4
1.2 Applicable Documents	5
1.3 Acronyms and Abbreviations	6
2. MESSAGE ACCESS PROFILE PROGRAMMING INTERFACE	8
2.1 Message Access Profile Commands.....	8
MAP_Open_Message_Access_Server.....	10
MAP_Open_Message_Notification_Server.....	10
MAP_Open_Request_Response	11
MAP_Register_Message_Access_Server_SDP_Record	12
MAP_Register_Message_Notification_Server_SDP_Record	13
MAP_Open_Remote_Message_Access_Server_Port.....	14
MAP_Open_Remote_Message_Notification_Server_Port.....	15
MAP_Close_Server	16
MAP_Close_Connection	17
MAP_Get_Server_Mode	17
MAP_Set_Server_Mode	18
MAP_Abort_Request.....	19
MAP_Set_Notification_Registration_Request	20
MAP_Set_Notification_Registration_Response.....	21
MAP_Send_Event_Request.....	21
MAP_Send_Event_Response	23
MAP_Get_Folder_Listing_Request	23
MAP_Get_Folder_Listing_Response	24
MAP_Get_Message_Listing_Request.....	26
MAP_Get_Message_Listing_Response.....	27
MAP_Get_Message_Request	28
MAP_Get_Message_Response.....	30
MAP_Set_Message_Status_Request	32
MAP_Set_Message_Status_Response.....	33
MAP_Push_Message_Request	34
MAP_Push_Message_Response.....	35
MAP_Update_Inbox_Request	36
MAP_Update_Inbox_Response.....	37
MAP_Set_Folder_Request	38
MAP_Set_Folder_Response	39
2.2 Message Access Profile Event Callback Prototypes.....	40
MAP_Event_Callback_t	40
2.3 Message Access Profile Events.....	42
etMAP_Open_Request_Indication	44
etMAP_Open_Port_Indication	44
etMAP_Open_Port_Confirmation	45

etMAP_Close_Port_Indication	45
etMAP_Notification_Registration_Indication	45
etMAP_Notification_Registration_Confirmation	46
etMAP_Send_Event_Indication	46
etMAP_Send_Event_Confirmation	47
etMAP_Get_Folder_Listing_Indication	47
etMAP_Get_Folder_Listing_Confirmation	48
etMAP_Get_Message_Listing_Indication	48
etMAP_Get_Message_Listing_Confirmation	49
etMAP_Get_Message_Indication	50
etMAP_Get_Message_Confirmation	50
etMAP_Set_Message_Status_Indication	51
etMAP_Set_Message_Status_Confirmation	51
etMAP_Push_Message_Indication	51
etMAP_Push_Message_Confirmation	52
etMAP_Update_Inbox_Indication	53
etMAP_Update_Inbox_Confirmation	53
etMAP_Set_Folder_Indication	53
etMAP_Set_Folder_Confirmation	54
etMAP_Abort_Indication	54
etMAP_Abort_Confirmation	55
3. FILE DISTRIBUTIONS.....	56

1. Introduction

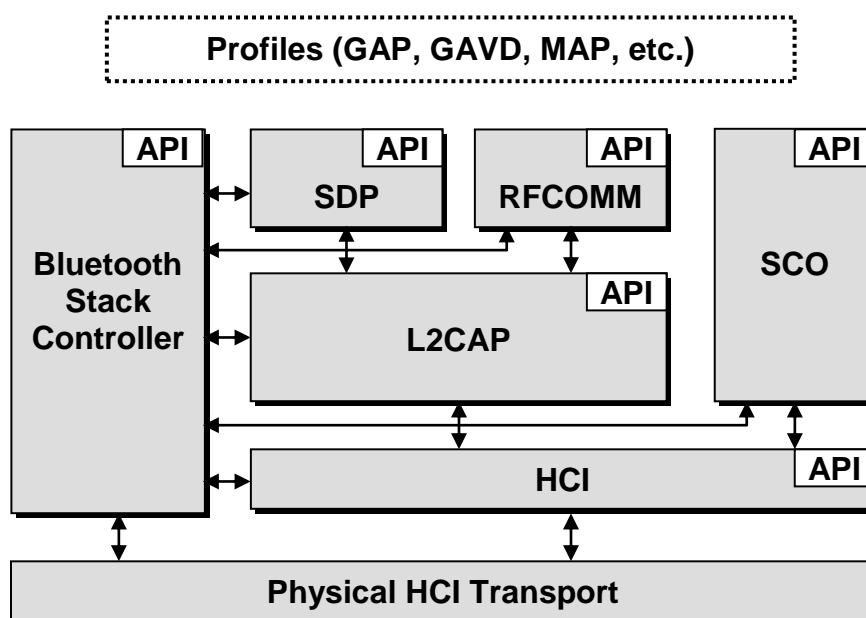
Bluetopia®, the Bluetooth Protocol Stack by Stonestreet One, provides a software architecture that encapsulates the upper functionality of the Bluetooth Protocol Stack. More specifically, this stack is a software solution that resides above the Physical HCI (Host Controller Interface) Transport Layer and extends through the L2CAP (Logical Link Control and Adaptation Protocol) and the SCO (Synchronous Connection-Oriented) Link layers. In addition to basic functionality at these layers, the Bluetooth Protocol Stack by Stonestreet One provides implementations of the Service Discovery Protocol (SDP), RFCOMM (the Radio Frequency serial COMMunications port emulator), and several of the Bluetooth Profiles. Program access to these layers, services, and profiles is handled via Application Programming Interface (API) calls.

This document focuses on the API reference that contains a description of all programming interfaces for the Bluetooth Message Access Profile provided by Bluetopia. Chapter 2 contains a description of the programming interface for this profile. And, Chapter 3 contains the header file name list for the Bluetooth Message Access Profile library.

1.1 Scope

This reference manual provides information on the Message Access Profile APIs identified in Figure 1-1 below. These APIs are available on the full range of platforms supported by Stonestreet One:

- Windows 95
- Windows NT 4.0
- Windows Millennium
- Windows 98
- Windows NT 4.0
- Windows Millennium
- Linux
- Windows 2000
- Windows CE



- QNX

Figure 1-1 the Stonestreet One Bluetooth Protocol Stack

1.2 Applicable Documents

The following documents may be used for additional background and technical depth regarding the Bluetooth technology.

1. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller volume]*, version 2.0 +EDR, November 2004.
2. *Specification of the Bluetooth System, Volume 3, Core System Package [Host volume]*, version 2.0 +EDR, November 2004.
3. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 2.1+EDR, July 26, 2007.
4. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 2.1+EDR, July 26, 2007.
5. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 2.1+EDR, July 26, 2007.
6. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 2.1+EDR, July 26, 2007.
7. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 2.1+EDR, July 26, 2007.
8. *Specification of the Bluetooth System, Bluetooth Core Specification Addendum 1*, June 26, 2008.
9. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 3.0+HS, April 21, 2009.
10. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 3.0+HS, April 21, 2009.
11. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 3.0+HS, April 21, 2009.
12. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 3.0+HS, April 21, 2009.
13. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 3.0+HS, April 21, 2009.
14. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 3.0+HS, April 21, 2009.
15. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 4.0, June 30, 2010.
16. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 4.0, June 30, 2010.

17. *Specification of the Bluetooth System, Volume 2, Core System Package [BR/EDR Controller Volume]*, version 4.0, June 30, 2010.
18. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 4.0, June 30, 2010.
19. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 4.0, June 30, 2010.
20. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 4.0, June 30, 2010.
21. *Specification of the Bluetooth System, Volume 6, Core System Package [Low Energy Controller Volume]*, version 4.0, June 30, 2010.
22. Adopted Bluetooth Profiles, Protocol and Transport specifications, various versions and dates, available from Bluetooth SIG.
23. *Bluetooth Assigned Numbers*, version 1.1, February 22, 2001.
24. *Digital cellular telecommunications system (Phase 2+); Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol (GSM 07.10)*, version 7.1.0, Release 1998; commonly referred to as: ETSI TS 07.10.
25. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual*, version 4.0.1, April 5, 2012.

Possible error returns are listed for each API function call. These are the *most likely* errors, but in fact programmers should allow for the possibility of any error listed in the BTErrors.h header file to occur as the value of a function return.

1.3 Acronyms and Abbreviations

Acronyms and abbreviations used in this document and other Bluetooth specifications are listed in the table below.

Term	Meaning
API	Application Programming Interface
BD_ADDR	Bluetooth Device Address
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Data Rate
HS	High Speed
LE	Low Energy
MAP	Message Access Profile
LSB	Least Significant Bit

Term	Meaning
MSB	Most Significant Bit
SDP	Service Discovery Protocol
SPP	Serial Port Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. Message Access Profile Programming Interface

The Message Access Profile (MAP) programming interface defines the protocols and procedures to be used to implement Message Access capabilities. The MAP commands are listed in section 2.1, the event callback prototype is described in section 2.2, and the MAP events are itemized in section 2.3. The actual prototypes and constants outlined in this section can be found in the **MAPAPI.H** header file in the Bluetopia distribution.

2.1 Message Access Profile Commands

The available MAP command functions are listed in the table below and are described in the text which follows.

Function	Description
MAP_Open_Message_Access_Server	Opens a Message Access Server on the specified RFCOMM Port.
MAP_Open_Message_Notification_Server	Opens a Message Notification Server on the specified RFCOMM Port.
MAP_Open_Request_Response	Supplies the user response to an Open Request from a remote device.
MAP_Register_Message_Access_Server_SDP_Record	Adds MAP Service Records to the SDP database for the Message Access Server.
MAP_Register_Message_Notification_Server_SDP_Record	Adds MAP Service Records to the SDP database for the Message Notification Server.
MAP_Un_Register_SDP_Record	Removes the SDP Information for the specified record handle.
MAP_Open_Remote_Message_Access_Server_Port	Establishes a connection to the specified Message Access Server.
MAP_Open_Remote_Message_Notification_Server_Port	Establishes a connection to the specified Message Notification Server.
MAP_Close_Server	Closes any connection to a remote Client and removes support for the specified server.
MAP_Close_Connection	Closes the connection to the specified server.
MAP_Get_Server_Mode	Provides a mechanism to query the current Message Access Profile Server Mode (Auto-Accept or Manual-Accept).
MAP_Set_Server_Mode	Controls the MAP operation for Auto-Accept or Manual-Accept of Client connections.
MAP_Abort_Request	Sends a request to a remote device to abort the

	current operation.
MAP_Set_Notification_Registration_Request	Sends a Notification Registration Request to a remote Message Access Server.
MAP_Set_Notification_Registration_Response	Sends a response to Notification Registration Request.
MAP_Send_Event_Request	Send an Event object to a remote Message Notification Server.
MAP_Send_Event_Response	Sends a response to a remote Notification Client after the reception of an Event Object.
MAP_Get_Folder_Listing_Request	Sends a Folder Listing Request to the remote Message Access server.
MAP_Get_Folder_Listing_Response	Sends a Folder Listing Object to a remote Message Access Client.
MAP_Get_Message_Listing_Request	Sends a Message Listing Request to the remote Message Access server.
MAP_Get_Message_Listing_Response	Sends a Message Listing Object to a remote Message Access Client.
MAP_Get_Message_Request	Sends a Message Request to the remote Message Access server.
MAP_Get_Message_Response	Sends a Message Object to a remote Message Access Client.
MAP_Set_Message_Status_Request	Sends a Set Message Status request to a remote Message Access Server.
MAP_Set_Message_Status_Response	Send a Set Message Status Response to a remote Message Access Client.
MAP_Push_Message_Request	Sends a Message Object to a remote Message Access Server to be delivered over the specified network.
MAP_Push_Message_Response	Sends a response to a remote Message Access Client.
MAP_Update_Inbox_Request	Generates a MAP Update Inbox Request to the specified remote MAP server.
MAP_Update_Inbox_Response	Sends a MAP Update Inbox Response to the specified remote MAP client.
MAP_Set_Folder_Request	Sends a request to a remote Message Access Server to change the current directory.
MAP_Set_Folder_Response	Sends a response to a remote Message Access Client.

MAP_Open_Message_Access_Server

This function opens a Message Access Server on the specified RFCOMM port.

Prototype:

```
int BTPSAPI MAP_Open_Message_Access_Server(unsigned int BluetoothStackID ,
      unsigned int MessageAccessServiceServerPort, MAP_Event_Callback_t EventCallback,
      unsigned long CallbackParameter);
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MessageAccessServerPort	Local RFCOMM Channel Number to use. This must fall in the range defined by the following constants: SPP_PORT_NUMBER_MINIMUM SPP_PORT_NUMBER_MAXIMUM
EventCallback	Function to call when events occur on this port.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each packet.

Return:

Positive, non-zero if successful. A successful return code will be a MAPID that can be used to reference the Opened MAP Port in ALL other functions in this module (except the MAP_Open_Remote_Message_Access_Server_Port and MAP_Open_Remote_Message_Notification_Server_Port).

An error code if negative; one of the following values may be returned:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INSUFFICIENT_RESOURCES
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
```

Possible Events:

etMAP_Open_Port_Indication

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Open_Message_Notification_Server

This function opens a Message Notification Server on the specified RFCOMM port.

Prototype:

```
int BTPSAPI MAP_Open_Message_Notification_Server(unsigned int BluetoothStackID,  
    unsigned int MessageNotificationServiceServerPort, unsigned int MAS_MAPID,  
    MAP_Event_Callback_t EventCallback, unsigned long CallbackParameter);
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MessageNotificationServerPort	Local RFCOMM Channel Number to use. This must fall in the range defined by the following constants: SPP_PORT_NUMBER_MINIMUM SPP_PORT_NUMBER_MAXIMUM
MAS_MAPID	The MAPID of the local Message Access Client that this server will be associated with. The Message Access Client must be connected to open this server.
EventCallback	Function to call when events occur on this port.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each packet.

Return:

Positive, non-zero if successful. A successful return code will be a MAPID that can be used to reference the Opened Port in the function MAP_Send_Event_x.

An error code if negative; one of the following values may be returned:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID  
BTMAP_ERROR_INSUFFICIENT_RESOURCES  
BTMAP_ERROR_NOT_INITIALIZED  
BTMAP_ERROR_INVALID_PARAMETER
```

Possible Events:

etMAP_Open_Port_Indication

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Open_Request_Response

This function is provided for Servers to send a response for an evMAP_Open_Port_Indication event. This event is generated when the Server Mode is set for Manual Accept.

Prototype:

int BTPSAPI **MAP_Open_Request_Response**(unsigned int BluetoothStackID, unsigned int MAPID, Boolean_t AcceptConnection);

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The MAPID that identifies the server that received the event. This is the value that was returned from one of the Open Server functions.

Return:

Zero if successful.

An error code if negative; one of the following values may be returned:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Register_Message_Access_Server_SDP_Record

This function provides a means to add an SDP Service Record to the SDP Database.

Notes:

1. This function should only be called with the MAPID that was returned from the MAP_Open_Message_Access_Server_Port() function. This function should **never** be used with the MAPID of MAP Clients.
2. The Service Record Handle that is returned from this function will remain in the SDP Record Database until it is deleted by calling the SDP_Delete_Service_Record() function. A Macro is provided to delete the Service Record from the SDP Database. This Macro maps MAP_Un_Register_SDP_Record() to SDP_Delete_Service_Record(), and is defined as follows:

MAP_Un_Register_SDP_Record(__BluetoothStackID, __ MAPID, _SDPRecordHandle)

3. The Service Name is always added at Attribute ID 0x0100. A Language Base Attribute ID List is created that specifies that 0x0100 is UTF-8 Encoded, English Language.

Prototype:

int BTPSAPI MAP_Register_Message_Access_Server_SDP_Record (unsigned int BluetoothStackID, unsigned int MAPID, char *ServiceName, Byte_t MASInstance, Byte_t SupportedMessageTypes, DWord_t *SDPServiceRecordHandle))

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The MAPID this command applies to. This is the value that was returned from the MAP_Open_Message_Access_Server_Port() function.
ServiceName	Name to appear in the SDP Database for this service.
MASInstance	A byte value that identifies an instance of the Server.
SDPServiceRecordHandle	Returned handle to the SDP Database entry that may be used to remove the entry at a later time.

Return:

Zero if successful.

An error code if negative; one of the following values may be returned:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Register_Message_Notification_Server_SDP_Record

This function provides a means to add an SDP Service Record to the SDP Database.

Notes:

4. This function should only be called with the MAPID that was returned from the MAP_Open_Message_Notification_Server_Port() function. This function should **never** be used with the Client MAPID.
5. The Service Record Handle that is returned from this function will remain in the SDP Record Database until it is deleted by calling the SDP_Delete_Service_Record() function. A Macro is provided to delete the Service Record from the SDP Database. This Macro maps MAP_Un_Register_SDP_Record() to SDP_Delete_Service_Record(), and is defined as follows:

MAP_Un_Register_SDP_Record(__BluetoothStackID, __MAPID, _SDPRecordHandle)

6. The Service Name is always added at Attribute ID 0x0100. A Language Base Attribute ID List is created that specifies that 0x0100 is UTF-8 Encoded, English Language.

Prototype:

```
int BTPSAPI MAP_Register_Message_Notification_SDP_Record(unsigned int
    BluetoothStackID, unsigned int MAPID, char *ServiceName, DWord_t
    *SDPServiceRecordHandle)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The MAPID this command applies to. This is the value that was returned from the MAP_Open_Message_Notification_Server_Port() function.
ServiceName	Name to appear in the SDP Database for this service.
SDPServiceRecordHandle	Returned handle to the SDP Database entry that may be used to remove the entry at a later time.

Return:

Zero if successful.

An error code if negative; one of the following values may be returned:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
```

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Open_Remote_Message_Access_Server_Port

This function is responsible for establishing a connection to a remote Message Access Server.

Prototype:

```
int BTPSAPI MAP_Open_Remote_Message_Access_Server_Port (unsigned int
    BluetoothStackID, BD_ADDR_t BD_ADDR, unsigned int ServerPort,
    MAP_Event_Callback_t EventCallback, unsigned long CallbackParameter)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
-------------------------------	---

BD_ADDR	The Bluetooth Address of the remote Message Access Server.
ServerPort	The RFCOMM Channel number where the server resides.
EventCallback	Function to call when events occur on this port.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each packet.

Return:

Positive, non-zero if successful. If this function is successful, the return value will represent the MAPID that can be passed to all other functions that require it.

An error code if negative; one of the following values may be returned:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Open_Remote_Message_Notification_Server_Port

This function is used to open a remote Message Notification Server Port on the specified Remote Device.

Prototype:

int BTPSAPI **MAP_Open_Remote_Message_Notification_Server_Port** (unsigned int BluetoothStackID, unsigned int LocalMAPID, unsigned int ServerPort, MAP_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
LocalMAPID	MAPID of the Message Access Client that is associated with this Server.
ServerPort	The RFCOMM Channel to connect with. This must fall in the range defined by the following constants: SPP_PORT_NUMBER_MINIMUM SPP_PORT_NUMBER_MAXIMUM
EventCallback	Function to call when events occur on this port.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each packet.

Return:

Positive, non-zero if successful. If this function is successful, the return value will represent the MAPID that can be passed to all other functions that require it.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INSUFFICIENT_RESOURCES
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Possible Events:

etMAP_Port_Open_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Close_Server

This function is used to close and remove a MAP Server that was previously opened with the MAP_Open_Message_Access_Server_Port() function OR the MAP_Open_Message_Notification_Server_Port() function. This function Un-Registers the MAP Server Port and SDP information from the system.

Prototype:

int BTPSAPI MAP_Close_Server(unsigned int BluetoothStackID, unsigned int MAPID)

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The MAP server port to close. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Close_Connection

This function is used to close a connection between a Client and Server. The function can be issued on either the Client or Server side. When used by a server, the connection is closed, but the server is still installed and can then receive another connection.

Prototype:

int BTPSAPI **MAP_Close_Connection** (unsigned int BluetoothStackID, unsigned int MAPID)

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from either the MAP_Open_Message_Access_Server(), MAP_Open_Message_Notification_Server(), Open_Remote_Message_Access_Server_Port() or Open_Remote_Message_Notification_Server_Port() function.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Get_Server_Mode

This function is responsible for retrieving the current Message Access Profile Server Mode.

Prototype:

```
int BTPSAPI MAP_Get_Server_Mode(unsigned int BluetoothStackID, unsigned int
    MAPID, unsigned long *ServerModeMask)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function or MAP_Open_Message_Notification_Server() function.
ServerModeMask	Pointer to a variable to receive the current Server Mode Mask. The following bits are defined for the returned value: MAP_SERVER_MODE_AUTOMATIC_ACCEPT_CONNECTION MAP_SERVER_MODE_MANUAL_ACCEPT_CONNECTION

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
```

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Set_Server_Mode

This function is responsible for setting the current Message Access Profile Server Mode.

Prototype:

```
int BTPSAPI MAP_Set_Server_Mode(unsigned int BluetoothStackID, unsigned int
    MAPID, unsigned long ServerModeMask)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the

MAP_Open_Message_Access_Server() function or
MAP_Open_Message_Notification_Server() function.

ServerModeMask The Server Mode Mask to set. The following bits are defined:
MAP_SERVER_MODE_AUTOMATIC_ACCEPT_CONNECTION
MAP_SERVER_MODE_MANUAL_ACCEPT_CONNECTION

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Abort_Request

This function is used to send an Abort Request to the Remote Device.

Prototype:

int BTPSAPI MAP_Abort_Request(unsigned int BluetoothStackID, unsigned int MAPID)

Parameters:

BluetoothStackID¹ Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.

MAPID The unique identifier of the connection this command is to be performed with. This command can be used by any Client or Server.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING

Possible Events:

etMAP_Abort_Confirmation

etMAP_Close_Port_Indication

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Set_Notification_Registration_Request

This function is used to send a request to a remote Message Access Server to establish a connection or disconnect a connection to a local Message Notification Server.

Prototype:

```
int BTPSAPI MAP_Set_Notification_Registration_Request(unsigned int  
    BluetoothStackID, unsigned int MAPID, Boolean_t Enabled)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.
Enabled	If TRUE, then connection is to be established. If FALSE, then the connection is to be disconnected.

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID  
BTMAP_ERROR_NOT_INITIALIZED  
BTMAP_ERROR_INVALID_PARAMETER
```

Possible Events:

etMAP_Notification_Registration_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Set_Notification_Registration_Response

This function is used to send a Set Notification Registration Response to the Remote Message Access Service Client. It is recommended that this function should be called after the establishment/disconnection to the Message Notification Server is complete so that a reasonable result code can be returned.

Prototype:

```
int BTPSAPI MAP_Set_Notification_Registration_Response(unsigned int  
    BluetoothStackID, unsigned int MAPID, Byte_t ResponseCode)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Accesss_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_PRECONDITION_FAILED MAP_OBEX_RESPONSE_SERVICE_UNAVAILABLE

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID  
BTMAP_ERROR_INVALID_PARAMETER  
BTMAP_ERROR_NOT_INITIALIZED  
BTMAP_ERROR_ACTION_NOT_ALLOWED
```

Possible Events:

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Send_Event_Request

This function is used to send an Event Object to a remote Message Notification Server.

Prototype:

```
int BTPSAPI MAP_Send_Event_Request(unsigned int BluetoothStackID, unsigned int
    MAPID, unsigned int DataLength, Byte_t *DataBuffer, unsigned int *AmountSent,
    Boolean_t Final)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Notification_Server_Port() function.
DataLength	The number of bytes in the object segment specified by DataBuffer argument below.
DataBuffer	Pointer to the segment of the event object data.
AmountSent	Pointer to a variable used to return to the caller the amount of data pointed to by the ObjectSegment parameter actually successfully sent in the request.
Final	Boolean flag indicating if this request is the last request segment of this operation.

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH
```

Possible Events:

etMAP_Send_Event_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Send_Event_Response

This function is used to send a Response for a Send Event Request to the Remote Message Notification Service Client.

Prototype:

```
int BTPSAPI MAP_Send_Event_Response(unsigned int BluetoothStackID, unsigned int  
MAPID, Byte_t ResponseCode)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Notification_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_PRECONDITION_FAILED MAP_OBEX_RESPONSE_SERVICE_UNAVAILABLE

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID  
BTMAP_ERROR_INVALID_PARAMETER  
BTMAP_ERROR_NOT_INITIALIZED  
BTMAP_ERROR_ACTION_NOT_ALLOWED
```

Possible Events:

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Get_Folder_Listing_Request

This function is used to send a request for a folder listing to a remote Message Access Server.

Prototype:

```
int BTPSAPI MAP_Get_Folder_Listing_Request(unsigned int BluetoothStackID, unsigned  
int MAPID, Word_t MaxListCount, Word_t ListStartOffset)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.
MaxListCount	Specifies the maximum number of folder entries to return for this request. If this parameter is 0, then the response to this will request will contain only the number of folders that are present in the current directory.
ListStartOffset	Identifies the List Entry that should be used to indicate the offset of the first entry of the return Folder Listing Object. The offset is Zero relative.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH

Possible Events:

etMAP_Get_Folder_Listing_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Get_Folder_Listing_Response

This function is used to send a Response for a Get Folder Listing Request to the Remote Message Access Client.

Prototype:

```
int BTPSAPI MAP_Get_Folder_Listing_Response(unsigned int BluetoothStackID,  
      unsigned int MAPID, Byte_t ResponseCode, Word_t *FolderCount, unsigned int  
      DataLength, Byte_t *DataBuffer, unsigned int *AmountSent)
```


Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_FOUND MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN
FolderCount	Pointer to a parameter that contains the number of folders that exist in the current directory. This parameter is used when the request contained a 0 for the MaxListCount parameter. If this parameter is non-NULL then all remaining parameters are ignored.
Data Length ²	Should be set to the number of bytes that are contained in the object segment pointed to by DataBuffer.
DataBuffer ²	Pointer to a byte buffer containing a segment of the Folder Listing Objects.
AmountSent ³	Pointer to a variable used to return to the caller the amount of data pointed to by the ObjectSegment parameter actually successfully sent in the request.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

2. If `DataBuffer` is non-NULL and `DataLength` greater than 0 will cause a Body or End-of-Body header to be added to the packet, either on the first or subsequent packets. See `MAPAPI.h` for the possible cases when the two different headers will be used.
3. If `AmountSent` returns a smaller amount than `DataLength`, then not all the data was sent in the call. The function should be called again with `DataLength` and `DataBuffer` modified to take into account the data that was actually sent (`AmountSent`).

MAP_Get_Message_Listing_Request

This function is used to send a request for a message listing to a remote Message Access Server.

Prototype:

```
int BTPSAPI MAP_Get_Message_Listing_Request(unsigned int BluetoothStackID,  
      unsigned int MAPID, Word_t *FolderName, Word_t MaxListCount, Word_t  
      ListStartOffset, MAP_Message_Listing_Info_t *ListingInfo)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to <code>BSC_Initialize</code> .
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the <code>MAP_Open_Remote_Message_Access_Server_Port()</code> function.
FolderName	Pointer to a UNICODE string that indicates the folder from which the listing should be generated. If the parameter is NULL, then current directory is used.
MaxListCount	Specifies the maximum number of message entries to return for this request. If this parameter is 0, then the response to this will request will contain only the number of messages that are present in the specified directory and an indication as to whether new messages exist.
ListStartOffset	Identifies the List Entry that should be used to indicate the offset of the first entry of the return Message Listing Object. The offset is Zero relative.
ListingInfo	This is a structure that contains information that can be used to create a filter for the resulting object. Refer to the MAP specification for information about the filtering that is supported.

Return:

Zero if successful.

An error code if negative; one of the following values:

`BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID`

BTMAP_ERROR_NOT_INITIALIZED
 BTMAP_ERROR_INVALID_PARAMETER
 BTMAP_ERROR_ACTION_NOT_ALLOWED
 BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
 BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH

Possible Events:

etMAP_Get_Message_Listing_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Get_Message_Listing_Response

This function is used to send a Response for a Get Message Listing Request to the Remote Message Access Client. It used used to respond to a MAP Get Message Listing Indication.

Prototype:

```
int BTPSAPI MAP_Get_Message_Listing_Response(unsigned int BluetoothStackID, unsigned
int MAPID, Byte_t ResponseCode, Word_t *MessageCount, Boolean_t NewMessage,
MAP_TimeDate_t *CurrentTime, unsigned int DataLength, Byte_t *DataBuffer, unsigned
int *AmountSent)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_FOUND MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN
MessageCount	Pointer to a parameter that contains the number of messages that exist in the specified directory. This parameter is used when the request contained a 0 for the MaxListCount

	parameter. If this parameter is non-NULL then all parameters other than then NewMessage parameter are ignored.
NewMessage	Boolean to indicate if any new message exists in the specified directory.
CurrentTime	Indicates the time at which the response is being sent and used by the receiver of this response to correlate the timestamps in the listing with the current time of the server.
DataLength ²	The number of bytes that are included in the object segment pointed to by DataBuffer
DataBuffer ²	Pointer to a byte buffer containing a segment of the Message Listing object
AmountSent ³	Pointer to a variable used to return to the caller the amount of data pointed to by the ObjectSegment parameter actually successfully sent in the request.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.
2. If DataBuffer is non-NULL and DataLength greater than 0 will cause a Body or End-of-Body header to be added to the packet, either on the first or subsequent packets. See MAPAPI.h for the possible cases when the two different headers will be used.
3. If AmountSent returns a smaller amount than DataLength, then not all the data was sent in the call. The function should be called again with DataLength and DataBuffer modified to take into account the data that was actually sent (AmountSent).

MAP_Get_Message_Request

This function is used to send a request for a specified message to a remote Message Access Server.

Prototype:

```
int BTPSAPI MAP_Get_Message_Request(unsigned int BluetoothStackID, unsigned int
    MAPID, char *MessageHandle, Boolean_t Attachment, MAP_CharSet_t CharSet,
    MAP_Fractional_Type_t FractionalType)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.
MessageHandle	Pointer to a NULL Terminated character string that indicates the handle of the message to be retrieved. The handle is obtained from the message listing object or event object and consists of 16 hexadecimal digits.
Attachment	Boolean Flag to indicate whether any existing attachments should be included in the Message Object.
CharSet	Specifies the character set that should be used by the server when generating the message object. The following character sets may be used with this command: csNative csUTF8
FractionalType	This is used to allow the reception of fragmented email. The following values can be used with this command. ftUnfragmented ftFirst ftNext The value ftFirst and ftNext shall not be used unless fractioned messages are supported by the message service.

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH
```

Possible Events:

etMAP_Get_Message_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Get_Message_Response

This function is used to send a Message Object to a Remote Message Access Client and respond to a MAP Get Message Indication.

Prototype:

```
int BTPSAPI MAP_Get_Message_Response(unsigned int BluetoothStackID, unsigned int
MAPID, Byte_t ResponseCode, MAP_Fractional_Type_t FractionalType, unsigned int
DataLength, Byte_t *DataBuffer, unsigned int *AmountSent)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_FOUND MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN
FractionalType	This is used to allow the reception of fragmented email. The following values can be used with this command. ftUnfragmented ftMore ftLast The value ftMore and ftLast shall not be used unless fractioned messages are supported by the message service and requested in the request.
DataLength	The number of bytes that are included in the object segment pointed to by DataBuffer

DataBuffer	Pointer to a byte buffer containing the Message Listing Object segment.
AmountSent	Pointer to a variable used to return to the caller the amount of data pointed to by the ObjectSegment parameter actually successfully sent in the request.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.
2. If DataBuffer is non-NULL and DataLength greater than 0 will cause a Body or End-of-Body header to be added to the packet, either on the first or subsequent packets. See MAPAPI.h for the possible cases when the two different headers will be used.
3. If AmountSent returns a smaller amount than DataLength, then not all the data was sent in the call. The function should be called again with DataLength and DataBuffer modified to take into account the data that was actually sent (AmountSent).

MAP_Set_Message_Status_Request

This function is used to set the status state of a message managed by the Message Access Server.

Prototype:

```
int BTPSAPI MAP_Set_Message_Status_Request(unsigned int BluetoothStackID, unsigned
int MAPID, char *MessageHandle, MAP_Status_Indicator_t StatusIndicator, Boolean_t
StatusValue)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.
MessageHandle	Pointer to a NULL Terminated character string that indicates the handle of the message to be retrieved. The handle is obtained from the message listing object or event object and consists of 16 hexadecimal digits.
StatusIndicator	Specifies which indicator is to be modified. The following values can be supplied for this parameter. siReadStatus siDeletedStatus
StatusValue	The following specified the new state of the specified indicator. If TRUE the indicator should be set and if FALSE the indicator should the cleared.

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH
```

Possible Events:

etMAP_Set_Message_Status_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Set_Message_Status_Response

This function is used to send a response for a Set Message Status request to a Remote Message Access Client.

Prototype:

```
int BTPSAPI MAP_Set_Message_Status_Response(unsigned int BluetoothStackID, unsigned int MAPID, Byte_t ResponseCode)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_FOUND MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Possible Events:

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Push_Message_Request

This function is used to send a message object to a Message Access Server for delivery over a supported message service.

Prototype:

```
int BTPSAPI MAP_Push_Message_Request(unsigned int BluetoothStackID, unsigned int
    MAPID, Word_t *FolderName, Boolean_t Transparent, Boolean_t Retry, MAP_CharSet_t
    CharSet, unsigned int DataLength, Byte_t *DataBuffer, unsigned int *AmountSent,
    Boolean_t Final)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.
FolderName	Pointer to a UNICODE string that indicates the folder in which the message shall be placed. If the parameter is NULL, then current directory is used.
Transparent	Boolean value to indicate whether a copy of the message should be placed in the SENT folder upon successful delivery. If TRUE, no copy is retained.
Retry	Boolean value to indicate whether a message should be retrieved after an unsuccessful attempt to deliver the message. If FALSE, no retry attempt is made.
CharSet	Specifies the character set that was used when during the creation of the message object. The following character sets may be used with this command: csNative csUTF8
DataLength	The number of bytes that are included in the object segment pointed to by DataBuffer.
DataBuffer	Pointer to byte buffer containing the Message Listing Object segment.
AmountSent	Pointer to a variable used to return to the caller the amount of data pointed to by the Buffer parameter actually successfully sent in the request.
Final	Boolean flag indicating if this request is the last request segment of this operation.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH

Possible Events:

etMAP_Push_Message_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Push_Message_Response

This function is used to send a response for a Push Message request to a Remote Message Access Client.

Prototype:

```
int BTPSAPI MAP_Push_Message_Response(unsigned int BluetoothStackID, unsigned int  
MAPID, Byte_t ResponseCode, char *MessageHandle)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_FOUND MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN

MessageHandle Pointer to a NULL Terminated character string that indicates the handle that was assigned to the message by the server. The handle consists of 16 hexadecimal digits.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Update_Inbox_Request

This function is used to instruct the Message Access Server to scan for new messages.

Prototype:

int BTPSAPI **MAP_Update_Inbox_Request**(unsigned int BluetoothStackID, unsigned int MAPID)

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH

Possible Events:

etMAP_Update_Inbox_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Update_Inbox_Response

This function is used to send a response for an Update Inbox request to a Remote Message Access Client.

Prototype:

```
int BTPSAPI MAP_Update_Inbox_Response(unsigned int BluetoothStackID, unsigned int  
MAPID, Byte_t ResponseCode)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Set_Folder_Request

This function is used to instruct the Message Access Server to move from the current directory.

Prototype:

```
int BTPSAPI MAP_Set_Folder_Request(unsigned int BluetoothStackID, unsigned int MAPID,
    MAP_Set_Folder_Option_t PathOption, Word_t *FolderName)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique client identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Remote_Message_Access_Server_Port() function.
PathOption	This parameter specifies the direction relative to the current directory for the move. The following values may be used for this parameter. sfRoot sfDown sfUp
FolderName	Pointer to a UNICODE string that indicates the destination folder from move. When the PathOption is soDown, this parameter is mandatory and specifies a child folder. When the PathOption is soUp, then this parameter is optional and is used to specify a child directory to move to after moving to the parent directory. When the PathOption is soRoot, this parameter is ignored.

Return:

Zero if successful.

An error code if negative; one of the following values:

```
BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_ACTION_NOT_ALLOWED
BTMAP_ERROR_REQUEST_ALREADY_OUTSTANDING
BTMAP_ERROR_INSUFFICIENT_PACKET_LENGTH
```

Possible Events:

etMAP_Set_Folder_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

MAP_Set_Folder_Response

This function is used to send a response for a Set Folder request to a Remote Message Access Client.

Prototype:

```
int BTPSAPI MAP_Set_Folder_Response(unsigned int BluetoothStackID, unsigned int  
MAPID, Byte_t ResponseCode)
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MAPID	The unique identifier of the connection this command is to be performed with. This is the value that was returned from the MAP_Open_Message_Access_Server() function.
ResponseCode	The Response Code to be associated with this response. The following are suggested values that are valid for the Response Code parameter: MAP_OBEX_RESPONSE_OK MAP_OBEX_RESPONSE_NOT_ACCEPTABLE MAP_OBEX_RESPONSE_UNAUTHORIZED MAP_OBEX_RESPONSE_FORBIDDEN

Return:

Zero if successful.

An error code if negative; one of the following values:

BTMAP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTMAP_ERROR_INVALID_PARAMETER
BTMAP_ERROR_NOT_INITIALIZED
BTMAP_ERROR_ACTION_NOT_ALLOWED

Possible Events:**Notes:**

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

2.2 Message Access Profile Event Callback Prototypes

The event callback functions, mentioned in the Message Access Profile Registration or Connection commands, all accept the callback function described by the following prototype.

MAP_Event_Callback_t

Prototype of callback function passed in one of the MAP Open commands.

Prototype:

```
void (BTPSAPI *MAP_Event_Callback_t)(unsigned int BluetoothStackID,  
    MAP_Event_Data_t *MAP_Event_Data, unsigned long CallbackParameter);
```

Parameters:

BluetoothStackID ¹	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize
MAP_Event_Data	Data describing the event for which the callback function is called. This is defined by the following structure:


```
typedef struct
{
    MAP_Event_Type_t Event_Data_Type;
    Word_t          Event_Data_Size;
    union
    {
        MAP_Open_Request_Indication_Data_t
            *MAP_Open_Request_Indication_Data;
        MAP_Open_Port_Indication_Data_t
            *MAP_Open_Port_Indication_Data;
        MAP_Open_Port_Confirmation_Data_t
            *MAP_Open_Port_Confirmation_Data;
        MAP_Close_Port_Indication_Data_t
            *MAP_Close_Port_Indication_Data;
        MAP_Notification_Registration_Indication_Data_t
            *MAP_Notification_Registration_Indication_Data;
        MAP_Notification_Registration_Confirmation_Data_t
            *MAP_Notification_Registration_Confirmation_Data;
        MAP_Send_Event_Indication_Data_t
            *MAP_Send_Event_Indication_Data;
        MAP_Send_Event_Confirmation_Data_t
            *MAP_Send_Event_Confirmation_Data;
        MAP_Get_Folder_Listing_Indication_Data_t
            *MAP_Get_Folder_Listing_Indication_Data;
        MAP_Get_Folder_Listing_Confirmation_Data_t
            *MAP_Get_Folder_Listing_Confirmation_Data;
        MAP_Get_Message_Listing_Indication_Data_t
            *MAP_Get_Message_Listing_Indication_Data;
        MAP_Get_Message_Listing_Confirmation_Data_t
            *MAP_Get_Message_Listing_Confirmation_Data;
        MAP_Get_Message_Indication_Data_t
            *MAP_Get_Message_Indication_Data;
        MAP_Get_Message_Confirmation_Data_t
            *MAP_Get_Message_Confirmation_Data;
        MAP_Set_Message_Status_Indication_Data_t
            *MAP_Set_Message_Status_Indication_Data;
        MAP_Set_Message_Status_Confirmation_Data_t
            *MAP_Set_Message_Status_Confirmation_Data;
        MAP_Push_Message_Indication_Data_t
            *MAP_Push_Message_Indication_Data;
        MAP_Push_Message_Confirmation_Data_t
            *MAP_Push_Message_Confirmation_Data;
        MAP_Update_Inbox_Indication_Data_t
            *MAP_Update_Inbox_Indication_Data;
        MAP_Update_Inbox_Confirmation_Data_t
            *MAP_Update_Inbox_Confirmation_Data;
    }
};
```

```

MAP_Set_Folder_Indication_Data_t
    *MAP_Set_Folder_Indication_Data;
MAP_Set_Folder_Confirmation_Data_t
    *MAP_Set_Folder_Confirmation_Data;
MAP_Abort_Indication_Data_t
    *MAP_Abort_Indication_Data;
MAP_Abort_Confirmation_Data_t
    *MAP_Abort_Confirmation_Data;
} Event_Data;
} MAP_Event_Data_t;

```

where, Event_Data_Type is one of the enumerations of the event types listed in the table in section 2.3, and each data structure in the union is described with its event in that section as well.

CallbackParameter User-defined parameter (e.g., tag value) that was defined in the callback registration.

Return:

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

2.3 Message Access Profile Events

The possible MAP Profile events from the Bluetooth stack is listed in the table below and are described in the text that follows:

Event	Description
etMAP_Open_Request_Indication	Dispatched when a remote client requests to connect to a local server.
etMAP_Open_Port_Indication	Dispatched when a remote Client connects to a Message Access Server or a Message Notification Server.
etMAP_Open_Port_Confirmation	Dispatched to the Initiator to indicate the success or failure of a previously submitted Connection Attempt to a remote Server Port.
etMAP_Close_Port_Indication	Dispatched when the Servers and Clients when a service connection is disconnected.
etMAP_Notification_Registration_Indication	Dispatched to the Message Access Server when the Message Access Client send a Notification Registration Request.

etMAP_Notification_Registration_Confirmation	Dispatched to the Message Access Client when the Message Access Server send a Notification Registration Response.
etMAP_Send_Event_Indication	Dispatched to the Message Notification Server when a Message Notification Client send a Send Event Request.
etMAP_Send_Event_Confirmation	Dispatched to the Message Notification Client when a Message Notification Server sends a Send Event Response.
etMAP_Get_Folder_Listing_Indication	Dispatched to the Message Access Server when the Client sends a Get Folder Listing Request.
etMAP_Get_Folder_Listing_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends a Get Folder Listing Response.
etMAP_Get_Message_Listing_Indication	Dispatched to the Message Access Server when the Client sends a Get Message Listing Request.
etMAP_Get_Message_Listing_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends a Get Message Listing Response.
etMAP_Get_Message_Indication	Dispatched to the Message Access Server when the Client sends a Get Message Request.
etMAP_Get_Message_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends a Get Message Response.
etMAP_Set_Message_Status_Indication	Dispatched to the Message Access Server when the Client sends a Set Message Status Request.
etMAP_Set_Message_Status_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends a Set Message Status Response.
etMAP_Push_Message_Indication	Dispatched to the Message Access Server when the Client sends a Push Message Request.
etMAP_Push_Message_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends a Push Message Response.
etMAP_Update_Inbox_Indication	Dispatched to the Message Access Server

	when the Client sends an Update Inbox Request.
etMAP_Update_Inbox_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends an Update Inbox Response.
etMAP_Set_Folder_Indication	Dispatched to the Message Access Server when the Client sends a Set Folder Request.
etMAP_Set_Folder_Confirmation	Dispatched to the Message Access Client when the Message Access Server sends a Set Folder Response.
etMAP_Abort_Indication	Dispatched to the Client/Server when the Server/Client sends an Abort Request.
etMAP_Abort_Confirmation	Dispatched to the Client/Server when the Server/Client sends an Abort Response.

etMAP_Open_Request_Indication

Indicate that an Open Port request has been received.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    BD_ADDR_t      BD_ADDR;
} MAP_Open_Request_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server on which the connect request was received.
BD_ADDR	Address of the Bluetooth Device making the request.

etMAP_Open_Port_Indication

Indicates a Client connection to a Server port.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    BD_ADDR_t      BD_ADDR;
} MAP_Open_Port_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server on which the connect request was made.
-------	---

BD_ADDR Address of the Bluetooth Device making the connection.

etMAP_Open_Port_Confirmation

Indicates the status of a connection request.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
    unsigned int  OpenStatus;
} MAP_Open_Port_Confirmation_Data_t;
```

Event Parameters:

MAPID Identifier of the MAP server on which the connect request was made.

OpenStatus One of the following possible status values:

MAP_OPEN_STATUS_SUCCESS
MAP_OPEN_STATUS_CONNECTION_TIMEOUT
MAP_OPEN_STATUS_CONNECTION_REFUSED
MAP_OPEN_STATUS_UNKNOWN_ERROR

etMAP_Close_Port_Indication

Indicate that a port has been closed by the remote Bluetooth Device.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
} MAP_Close_Port_Indication_Data_t;
```

Event Parameters:

MAPID Identifier of the MAP connection that was closed.

etMAP_Notification_Registration_Indication

Dispatched to the Message Access Server when the Message Access Client sends a Notification Registration Request.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    BD_ADDR_t      BD_ADDR;
    Boolean_t       NotificationStatus;
} MAP_Notification_Registration_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
BD_ADDR	Address of the Bluetooth Device making the request.
NotificationStatus	Indicates if a connection should be made or disconnected.

etMAP_Notification_Registration_Confirmation

Dispatched to the Message Access Client when the Message Access Server send a Notification Registration Response.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Byte_t          ResponseCode;
} MAP_Notification_Registration_Confirmation_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

etMAP_Send_Event_Indication

Dispatched to the Message Notification Server when a Message Notification Client send a Send Event Request.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Byte_t          MASInstanceID;
    unsigned int    DataLength;
    Byte_t          *DataBuffer;
    Boolean_t       Final;
} MAP_Send_Event_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
MASInstanceID	Identifies the instance of the Server sending the event.

DataLength	The number of bytes in the Event Object pointed to by DataBuffer.
DataBuffer	Pointer to byte buffer containing the Event Object segment.
Final	Indicates whether this is the last segment of the object.

etMAP_Send_Event_Confirmation

Dispatched to the Message Notification Client when a Message Notification Server sends a Send Event Response.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Byte_t          ResponseCode;
} MAP_Send_Event_Confirmation_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

etMAP_Get_Folder_Listing_Indication

Dispatched to the Message Access Server when the Client sends a Get Folder Listing Request.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Word_t          MaxListCount;
    Word_t          ListStartOffset;
} MAP_Get_Folder_Listing_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
MaxListCount	Identifies the maximum number of list entries that can be contained in the listing object. If Zero, then this indicates that the response should only contain the actual number of entries in the current folder.
ListStartOffset	Identifies the List Entry that should be used as the offset of the first entry of the returned Folder Listing Object. The offset is Zero relative.

etMAP_Get_Folder_Listing_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends a Get Folder Listing Response

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Byte_t          ResponseCode;
    Word_t          NumberOfFolders;
    unsigned int    DataLength;
    Byte_t          *DataBuffer;
} MAP_Get_Folder_Listing_Confirmation_Data_t;
```

EventParameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.
NumberOfFolder	Indicates the number of folders in the current directory. This parameter is meaningful ONLY if the request was made with MaxListCount equal to zero.
DataLength	Number of bytes in the Folder Listing Object pointed to by DataBuffer.
DataBuffer	Pointer to a byte buffer containing the Folder Listing Object segment

etMAP_Get_Message_Listing_Indication

Dispatched to the Message Access Server when the Client sends a Get Message Listing Request.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Word_t          *FolderName;
    Word_t          MaxListCount;
    Word_t          ListStartOffset;
    MAP_Message_Listing_Info_t ListingInfo;
} MAP_Get_Message_Listing_Indication_Data_t;
```

EventParameters:

MAPID	Identifier of the MAP server connection.
FolderName	Pointer to a UNICODE string that indicates the folder from which the listing should be generated. If the parameter is NULL, then current directory is used.

MaxListCount	Specifies the maximum number of message entries to return for this request. If this parameter is 0, then the response to this request must contain only the number of messages that are present in the specified directory and an indication as to whether new messages exist.
ListStartOffset	Identifies the List Entry that should be used to indicate the offset of the first entry of the return Message Listing Object. The offset is Zero relative.
ListingInfo	This is a structure that contains information that can be used to create a filter for the resulting object. Refer to the MAP specification for information about the filtering that is supported.

etMAP_Get_Message_Listing_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends a Get Message Listing Response.

Return Structure:

```
typedef struct
{
    unsigned int      MAPID;
    Byte_t            ResponseCode;
    Boolean_t          NewMessage;
    Word_t             NumberOfMessages;
    unsigned int       DataLength;
    Byte_t             *DataBuffer;
    MAP_TimeDate_t     MSETime;
} MAP_Get_Message_Listing_Confirmation_Data_t;
```

EventParameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.
NewMessage	Boolean to indicate if any new messages exist in the specified directory.
MSETime	Indicates the time on the server at which the response was sent and can be used by the receiver of this response to correlate the timestamps in the listing with the current time of the server.
NumberOfMessages	Indicates the number of messages in the current directory. This parameter is meaningful ONLY if the request was made with MaxListCount equal to zero.
DataLength	Number of bytes contained in the Message Listing Object pointed to by DataBuffer.
DataBuffer	Pointer to byte buffer that contains the Message Listing Object segment.

etMAP_Get_Message_Indication

Dispatched to the Message Access Server when the Client sends a Get Message Request.

Return Structure:

```
typedef struct
{
    unsigned int      MAPID;
    Boolean_t         Attachment;
    MAP_CharSet_t     CharSet;
    MAP_Fractional_Type_t FractionalType;
    char              *MessageHandle;
} MAP_Get_Message_Indication_Data_t;
```

EventParameters:

MAPID	Identifier of the MAP server connection.
Attachment	Indicates whether any existing attachments should be included in the Message Object.
CharSet	Specifies the character set that should be used when constructing the Message Object.
FractionalType	Indicates if the request is for a fractional message. If the message is fractional, this indicates the fraction that is being requested.
MessageHandle	Pointer to a NULL terminated string that specifies the handle to the message that is being requested.

etMAP_Get_Message_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends a Get Message Response.

Return Structure:

```
typedef struct
{
    unsigned int      MAPID;
    Byte_t            ResponseCode;
    MAP_Fractional_Type_t FractionalType;
    unsigned int      DataLength;
    Byte_t            *DataBuffer;
} MAP_Get_Message_Confirmation_Data_t;
```

EventParameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

FractionalType	Indicates whether this is a fractional message. If the message is fractional, this indicated the fraction that is being returned.
DataLength	Number of bytes contained in the Message Object pointed to by DataBuffer.
DataBuffer	Pointer to byte buffer that contains the Message Object segment.

etMAP_Set_Message_Status_Indication

Dispatched to the Message Access Server when the Client sends a Set Message Status Request.

Return Structure:

```
typedef struct
{
    unsigned int      MAPID;
    char              *MessageHandle;
    MAP_Status_Indicator_t  StatusIndicator;
    Boolean_t         StatusValue;
} MAP_Set_Message_Status_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
MessageHandle	Pointer to a NULL terminated string that specifies the handle to the message that is being requested.
StatusIndicator	Identifies that status parameter that is to be modified
StatusValue	Indicates the new state of the status indicator.

etMAP_Set_Message_Status_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends a Set Message Status Response.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
    Byte_t        ResponseCode;
} MAP_Set_Message_Status_Confirmation_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

etMAP_Push_Message_Indication

Dispatched to the Message Access Server when the Client sends a Push Message Request.

Return Structure:

```
typedef struct
{
    unsigned int      MAPID;
    Word_t            *FolderName;
    Boolean_t          Transparent;
    Boolean_t          Retry;
    MAP_CharSet_t      CharSet;
    unsigned int       DataLength;
    Byte_t             *DataBuffer;
    Boolean_t          Final;
} MAP_Push_Message_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
FolderName	Pointer to a UNICODE string that indicates the folder where then message should be placed. If the parameter is NULL, then current directory is used.
Transparent	Boolean value to indicate whether a copy of the message should be placed in the SENT folder upon successful delivery. If TRUE, no copy is retained.
Retry	Boolean value to indicate whether a message should be retried after an unsuccessful attempt to deliver the message. If FALSE, no retry attempt is made.
CharSet	Specifies the character set that was used during the creation of the message object.
DataLength	Number of bytes contained in the Message Object pointed to by DataBuffer.
DataBuffer	Pointer to byte buffer that contains the Message Object segment.
Final	Indicates if this is the final segment of the message.

etMAP_Push_Message_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends a Push Message Response.

Return Structure:

```
typedef struct
{
    unsigned int    MAPID;
    Byte_t          ResponseCode;
    char            *MessageHandle;
} MAP_Push_Message_Confirmation_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

etMAP_Update_Inbox_Indication

Dispatched to the Message Access Server when the Client sends an Update Inbox Request.

Return Structure:

```
typedef struct
{
    unsigned int MAPID;
} MAP_Update_Inbox_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
-------	--

etMAP_Update_Inbox_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends an Update Inbox Response.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
    Byte_t        ResponseCode;
} MAP_Update_Inbox_Confirmation_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

etMAP_Set_Folder_Indication

Dispatched to the Message Access Server when the Client sends a Set Folder Request.

Return Structure:

```
typedef struct
{
    unsigned int      MAPID;
    MAP_Set_Folder_Option_t PathOption;
    Word_t            *FolderName;
} MAP_Set_Folder_Indication_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
PathOption	This parameter specifies the direction relative to the current directory for the move. The following values may be used for this parameter. sfRoot sfDown sfUp
FolderName	Pointer to a UNICODE string that indicates the destination folder. When the PathOption is soDown, this parameter is mandatory and specifies a child folder. When the PathOption is soUp, then this parameter is optional and is used to specify a child directory to move to after moving to the parent directory. When the PathOption is soRoot, this parameter is ignored.

etMAP_Set_Folder_Confirmation

Dispatched to the Message Access Client when the Message Access Server sends a Set Folder Response.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
    Byte_t        ResponseCode;
} MAP_Set_Folder_Confirmation_Data_t;
```

Event Parameters:

MAPID	Identifier of the MAP server connection.
ResponseCode	The response code associated with the generated event.

etMAP_Abort_Indication

Dispatched to the Client/Server when the Server/Client sends an Abort Request.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
} MAP_Abort_Indication_Data_t;
```

Event Parameters:

MAPID Identifier of the MAP connection.

etMAP_Abort_Confirmation

Dispatched to the Client/Server when the Server/Client sends an Abort Response.

Return Structure:

```
typedef struct
{
    unsigned int  MAPID;
} MAP_Abort_Confirmation_Data_t;
```

Event Parameters:

MAPID Identifier of the MAP connection.

3. File Distributions

The header files that are distributed with the Bluetooth Message Access Profile Library are listed in the table below.

File	Contents/Description
MAPAPI.h	Bluetooth Message Access Profile API definitions
SS1BTMAP.h	Bluetooth Message Access Profile Include file