Alert Notification Service (ANS)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
May 1, 2013

Louisville, KY www.stonestreetone.com

Table of Contents

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 2000
- Windows CE
- Linux
- QNX

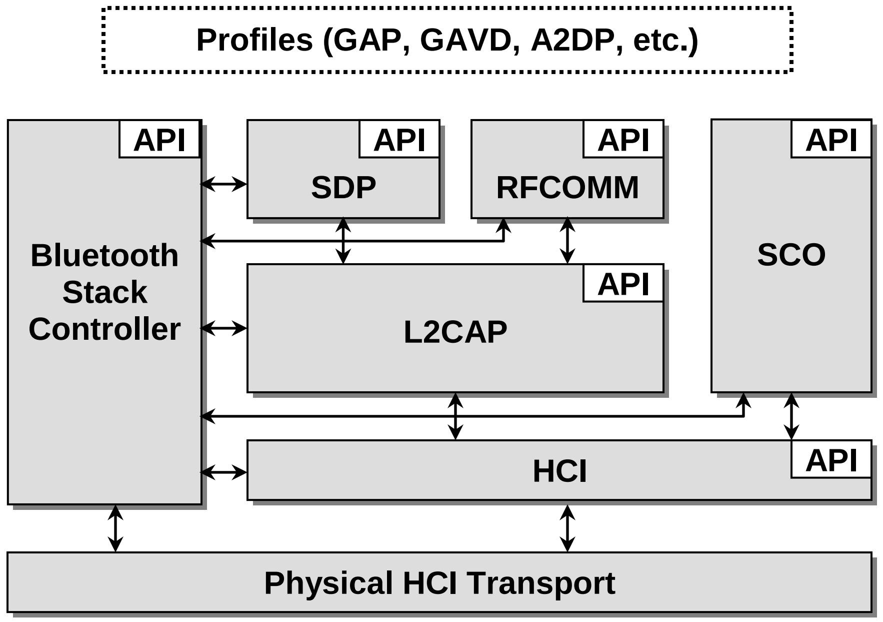

Figure 1-1 the Stonestreet One Bluetooth Protocol Stack

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 1, Architecture and Terminology Overview, version 4.0, June 30, 2010.*
2. *Specification of the Bluetooth System, Volume 6, Core System Package [Low Energy Controller Volume]*, version 4.0, June 30, 2010.
3. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual*, version 4.0.1, April 5, 2012.
4. *Bluetooth Doc Alert Notification Service Specification, version v10r00, September 15, 2011*.

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
ATT	Attribute Protocol
ANS	Alert Notification Service
BD_ADDR	Bluetooth Device Address
BT	Bluetooth
DIS	Device Information Service
GATT	Generic Attribute Protocol
GAPS	Generic Access Profile Service
HCI	Host Controller Interface
HS	High Speed
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy

2. Alert Notification Service Programming Interfaces

The Alert Notification Service programming interface defines the protocols and procedures to be used to implement Alert Notification Service capabilities. The Alert Notification Service commands are listed in section 2.1, the event callback prototypes are described in section 2.2, and the Alert Notification Service events are itemized in section 2.3. The actual prototypes and constants outlined in this section can be found in the **ANSAPI.H** header file in the Bluetopia distribution.

2.1 Alert Notification Service Commands

The available Alert Notification Service command functions are listed in the table below and are described in the text that follows.

Function	Description
ANS_Initialize_Service	Opens an ANS Server.
ANS_Cleanup_Service	Closes an opened ANS Server.
ANS_Set_Supported_Categories	Sets the Alert Notification Supported Categories for the specified Category Type.
ANS_Query_Supported_Categories	Gets the Alert Notification Supported Categories for the specified Category Type.
ANS_Read_Client_Configuration_Response	Responds to an ANS Read Client Configuration Request.
ANS_New_Alert_Notification	Sends New Alert notification to a specified remote device.
ANS_Un_Read_Alert_Status_Notification	Sends Unread Alert Status notification to a specified remote device.
ANS_Decode_New_Alert_Notification	Parses a value received from a remote ANS Server interpreting it as a New Alert Notification.
ANS_Free_New_Alert_Data	Frees the memory of New Alert Data.
ANS_Decode_Un_Read_Alert_Status_ Notification	Parses a value received from a remote ANS Server interpreting it as an UnRead Alert Notification.
ANS_Decode_Supported_Categories	Parses a value received from a remote ANS Server interpreting it as a Supported Categories Notification.
ANS_Format_Control_Point_Command	Formats an Alert Notification Control Point Command into a user specified buffer.

ANS_Initialize_Service

This function opens an ANS Server on a specified Bluetooth Stack.

Notes:

1. Only ONE ANS Server may be open at a time, per Bluetooth Stack ID.
2. All Client Requests will be dispatch to the EventCallback function that is specified by the second parameter to this function.

Prototype:

int BTPSAPI **ANS_Initialize_Service** (unsigned int BluetoothStackID,
ANS_Event_Callback_t EventCallback, unsigned long CallbackParameter,
unsigned int *ServiceID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
EventCallback	Callback function that is registered to receive events that are associated with the specified service.
CallbackParameter	A user-defined parameter that will be passed back to the user in the callback function.
ServiceID	Unique GATT Service ID of the registered ANS service returned from GATT_Register_Service API.

Return:

Positive non-zero if successful. The return value will be the Service Instance ID of ANS Server that was successfully opened on the specified Bluetooth Stack ID. This is the value that should be used in all subsequent function calls that require Instance ID.

Negative if an error occurred. Possible values are:

ANS_ERROR_INSUFFICIENT_RESOURCES
ANS_ERROR_SERVICE_ALREADY_REGISTERED
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_SERVICE_TABLE_FORMAT
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_NOT_INITIALIZED

Possible Events:

None.

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.

ANS_Cleanup_Service

This function is responsible for cleaning up and freeing all resources associated with an ANS Service Instance. After this function is called, no other ANS Service function can be called until after a successful call to the ANS_Initialize_Service() function is performed.

Prototype:

int BTPSAPI **ANS_Cleanup_Service**(unsigned int BluetoothStackID,

unsigned int InstanceID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to Cleanup ANS Service. This InstanceID was returned from the ANS_Initialize_Service() function.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INVALID_PARAMETER
ANS_ERROR_INVALID_INSTANCE_ID

Possible Events:

None.

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.

ANS_Set_Supported_Categories

This function is responsible for setting the Alert Notification Supported Categories for the specified Category Type on the specified ANS Instance.

Prototype:

int BTPSAPI **ANS_Set_Supported_Categories**(unsigned int BluetoothStackID, unsigned int InstanceID, ANS_Supported_Categories_Type_t SupportedCategoryType, Word_t SupportedCategoriesMask)

Parameters:

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

InstanceID	Specifies the unique Service Instance ID on which Supported Categories will be set. . This InstanceID was returned from the ANS_Initialize_Service() function.
SupportedCategoryType	Specifies the Category Type to set the Supported Categories for. This is defined to be one of the following values: scNewAlert, scUnreadAlertStatus
SupportedCategoriesMask	The Supported Categories bit mask is to set as the supported categories for the specified ANS Instance. The SupportedCategoriesMask is a bit mask that is made up of bit masks of the form ANS_SUPPORTED_CATEGORIES_XXX.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INVALID_INSTANCE_ID
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

None.

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.

ANS_Query_Supported_Categories

This function is responsible for querying the Alert Notification Supported Categories for the specified Category Type from the specified ANS Instance.

Prototype:

int BTPSAPI **ANS_Query_Supported_Categories**(unsigned int BluetoothStackID, unsigned int InstanceID, ANS_Supported_Categories_Type_t SupportedCategoryType, Word_t *SupportedCategoriesMask)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	Specifies the unique Service Instance ID to read from. . This InstanceID was returned from the ANS_Initialize_Service() function.

SupportedCategoryType	Specifies the Category Type to query the Supported Categories for. This is defined to be one of the following values: scNewAlert, scUnreadAlertStatus
SupportedCategoriesMask	The SupportedCategoriesMask is a pointer to a bit mask that will be made up of bit masks of the form ANS_SUPPORTED_CATEGORIES_XXX, if this function returns success.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INVALID_INSTANCE_ID
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER.

Possible Events:

None.

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.

ANS_Read_Client_Configuration_Response

The following function is responsible for responding to an ANS Read Client Configuration Request.

Prototype:

int BTPSAPI **ANS_Read_Client_Configuration_Response** (unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int TransactionID, Boolean_t NotificationsEnabled)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	Specifies the unique Service Instance ID to read Client Configuration Response. . This InstanceID was returned from the ANS_Initialize_Service() function.
TransactionID	The Transaction ID of the original read request. This value was received in the etANS_Read_Client_Configuration_Request event.

NotificationsEnabled	TRUE if The Client Configuration Characteristic Descriptor is enabled, FALSE otherwise.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INVALID_INSTANCE_ID
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER.

Possible Events:

etGATT_Client_Read_Response

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.

ANS_New_Alert_Notification

The following function is responsible for sending a New Alert notification to a specified remote device.

Prototype:

int BTPSAPI **ANS_New_Alert_Notification**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, ANS_New_Alert_Data_t *NewAlert)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to notify new alert. . This InstanceID was returned from the ANS_Initialize_Service() function.
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value notification.
NewAlert	New Alert Data structure contains all of the required and optional data for the notification. This structure is declared as follows:

```
typedef struct
{
   ANS_Category_Identification_t          CategoryID;
   Byte_t                                 NumberOfNewAlerts;
```

```
    char                                *LastAlertString;
}ANS_New_Alert_Data_t;
```

Where the CategoryID is defined to be one of the following values:

ciSimpleAlert,	ciEmail,	ciNews,
ciCall,	ciMissedCall,	ciSMS_MMS,
ciVoiceMail,	ciSchedule,	ciHighPriorityAlert,
ciInstantMessage,	ciAllCategories	

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INSUFFICIENT_RESOURCES
ANS_ERROR_INVALID_INSTANCE_ID
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER.

Possible Events:

etGATT_Connection_Server_Notification

Notes:

1. The BluetoothStackID parameter is not included in versionsof 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.

ANS_Un_Read_Alert_Status_Notification

The following function is responsible for sending an Unread Alert Status notification to a specified remote device.

Prototype:

int BTPSAPI **ANS_Un_Read_Alert_Status_Notification**(unsigned int BluetoothStackID,
unsigned int InstanceID, unsigned int ConnectionID,
ANS_Un_Read_Alert_Data_t *UnReadAlert)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to notify unread alert status. . This InstanceID was returned from the ANS_Initialize_Service() function.

ConnectionID	Connection ID of the currently connected remote client device to send the handle/value notification.
UnReadAlert	Un Read Alert Data structure contains all of the required and optional data for the notification. This structure is declared as follows:

```
typedef struct _tagANS_Un_Read_Alert_Data_t
{
   ANS_Category_Identification_t    CategoryID;
   Byte_t                           NumberOfUnreadAlerts;
}ANS_Un_Read_Alert_Data_t;
```

Where the CategoryID is defined to be one of the following values:

ciSimpleAlert,	ciEmail,	ciNews,
ciCall,	ciMissedCall,	ciSMS_MMS,
ciVoiceMail,	ciSchedule,	ciHighPriorityAlert,
ciInstantMessage,	ciAllCategories	

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INSUFFICIENT_RESOURCES
ANS_ERROR_INVALID_INSTANCE_ID
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER.

Possible Events:

etGATT_Connection_Server_Notification

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.

ANS_Decode_New_Alert_Notification

The following function is responsible for parsing a New Alert notification received from a remote ANS Server.

NOTES:

1. The return value from this function MUST be freed by calling ANS_Free_New_Alert_Data() when the decoded New Alert Notification is no longer needed.

Prototype:

ANS_New_Alert_Data_t *BTPSAPI **ANS_Decode_New_Alert_Notification**(
    unsigned int ValueLength, Byte_t *Value)

Parameters:

ValueLength	Specifies the length of the New Alert Notification value returned by the remote ANS Server.
Value	Value is a pointer to the New Alert Notification data returned by the remote ANS Server. The Value parameter must point to a buffer of at least ANS_NEW_ALERT_NOTIFICATION_DATA_SIZE bytes size.

Return:

A pointer to the decoded New Alert data if successful or NULL if an error occurred.

Possible Events:

None.

Notes:

1. None.

ANS_Free_New_Alert_Data

The following function is responsible for freeing the memory of decoded New Alert Data that was returned by a successful call to ANS_Decode_New_Alert_Notification.

Prototype:

void BTPSAPI **ANS_Free_New_Alert_Data**(ANS_New_Alert_Data_t NewAlertData)

Parameters:

NewAlertData — New Alert Data structure contains all of the required and optional data for the notification. This structure is declared as follows:

```
typedef struct
{
   ANS_Category_Identification_t    CategoryID;
   Byte_t                           NumberOfNewAlerts;
   char                             *LastAlertString;
}ANS_New_Alert_Data_t;
```

Where the CategoryID is defined to be one of the following values:

ciSimpleAlert,	ciEmail,	ciNews,
ciCall,	ciMissedCall,	ciSMS_MMS,
ciVoiceMail,	ciSchedule,	ciHighPriorityAlert,
ciInstantMessage,	ciAllCategories	

Return:

None.

Possible Events:

None.

Notes:

1. None.

ANS_Decode_Un_Read_Alert_Status_Notification

The following function is responsible for parsing an Unread Alert Status notification received from a remote ANS Server.

Prototype:

int BTPSAPI **ANS_Decode_Un_Read_Alert_Status_Notification**(unsigned int ValueLength, Byte_t *Value, ANS_Un_Read_Alert_Data_t *UnReadAlert);

Parameters:

ValueLength	Specifies the length of the Unread Alert Notification value returned by the remote ANS Server.
Value	Value is a pointer to the Unread Alert Notification data returned by the remote ANS Server. The Value parameter must point to a buffer of at least ANS_UNREAD_ALERT_STATUS_NOTIFICATION_DATA_SIZE bytes size.
UnReadAlert	A pointer to the location that will store the Unread Alert Data. This structure is defined as follows:

```
typedef struct _tagANS_Un_Read_Alert_Data_t
{
    ANS_Category_Identification_t    CategoryID;
    Byte_t                           NumberOfUnreadAlerts;
}ANS_Un_Read_Alert_Data_t;
```

Where the CategoryId is defined to be one of the following values:

```
ciSimpleAlert,       ciEmail,          ciNews,
ciCall,              ciMissedCall,     ciSMS_MMS,
ciVoiceMail,         ciSchedule,       ciHighPriorityAlert,
ciInstantMessage,    ciAllCategories
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_MALFORMATTED_DATA
ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED

BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

None.

Notes:

1. None.

ANS_Decode_Supported_Categories

The following function is responsible for parsing a Supported Categories value received from a remote ANS Server.

Prototype:

int BTPSAPI **ANS_Decode_Supported_Categories**(unsigned int ValueLength, Byte_t *Value, Word_t *SupportedCategoriesMask);

Parameters:

ValueLength	Specifies the length of the Supported Categories value returned by the remote ANS Server.
Value	Value is a pointer to the Supported Categories data returned by the remote ANS Server. The Value parameter must point to a buffer of at least size of 2 Bytes.
SupportedCategoriesMask	A pointer to the location that will store the Supported Categories bit mask. The SupportedCategoriesMask is a bit mask that is made up of bit masks of the form ANS_SUPPORTED_CATEGORIES_XXX.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

None.

Notes:

1. None

ANS_Format_Control_Point_Command

The following function is responsible for formatting an Alert Notification Control Point Command into a user specified buffer.

Prototype:

int BTPSAPI **ANS_Format_Control_Point_Command**
(ANS_Control_Point_Command_Value_t *CommandBuffer,
ANS_Control_Point_Command_t Command, ANS_Category_Identification_t
CommandCategory);

Parameters:

CommandBuffer	Specifies The Command Buffer to format the command into. This Structure is defined as follows:
	typedef struct
	{
	NonAlignedByte_t CommandID;
	NonAlignedByte_t CategoryID;
	} **ANS_Control_Point_Command_Value_t**;

Command	Specifies the Command to format into the buffer. This is defined to be one of the following values: pcEnable_New_Alert_Notifications, pcEnable_Unread_Category_Notifications, pcDisable_New_Alert_Notifications, pcDisable_Unread_Category_Notifications, pcNotify_New_Alert_Immediately, pcNotify_Unread_Category_Immediately
CommandCategory	Specifies the category that the command applies to. This is defined to be one of the following values: ciSimpleAlert, ciEmail, ciNews, ciCall, ciMissedCall, ciSMS_MMS, ciVoiceMail, ciSchedule, ciHighPriorityAlert, ciInstantMessage, ciAllCategories

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

ANS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

None.

Notes:

1. None.

2.2 Alert Notification Service Event Callback Prototypes

2.2.1 Server Event Callback

The event callback function mentioned in the ANS_Initialize_Service command accepts the callback function described by the following prototype.

ANS_Event_Callback_t

Prototype of callback function passed in the ANS_Initialize_Service command.

Prototype:

void (BTPSAPI ***ANS_Event_Callback_t**)(unsigned int BluetoothStackID, ANS_Event_Data_t *ANS_Event_Data, unsigned long CallbackParameter)

Parameters:

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

```
typedef struct
{
    ANS_Event_Type_t      Event_Data_Type;
    Word_t                Event_Data_Size;
    union
    {
        ANS_Read_Client_Configuration_Data_t
        *ANS_Read_Client_Configuration_Data;
        ANS_Client_Configuration_Update_Data_t
        *ANS_Client_Configuration_Update_Data;
        ANS_Control_Point_Command_Data_t
        *ANS_Control_Point_Command_Data;
    } Event_Data;
} ANS_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 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 Alert Notification Service Events

The Alert Notification Service contains events that are received by the server. The following sections detail those events:

Event	Description
etANS_Server_Read_Client_Configuration_Reques t	Dispatched when an ANS Client requests to read Client Configuration Descriptor from a registered ANS Server.
etANS_Server_Client_Configuration_Update	Dispatched when an ANS Client requests to update Client Configuration Descriptor on to a registered ANS Server.
etANS_Server_Control_Point_Command	Dispatched to a ANS Server in response to the reception of request from ANS Client to write to the Control Point command.

etANS_Server_Read_Client_Configuration_Request

Dispatched when an ANS Client requests to read Client Configuration Descriptor from a registered ANS Server.

Return Structure:

```
typedef struct
{
   unsigned int                InstanceID;
   unsigned int                ConnectionID;
   unsigned int                TransactionID;
   GATT_Connection_Type_t      ConnectionType;
   BD_ADDR_t                   RemoteDevice;
   ANS_Characteristic_Type_t   ClientConfigurationType;
}ANS_Read_Client_Configuration_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Connection ID of the currently connected remote ANS server device.
TransactionID	The TransactionID identifies the transaction between a client and server. This identifier should be used to respond to current request.

ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.
ClientConfigurationType	Identifies the Client Characteristic Configuration Type for which configuration data to be read from remote Device. The ClientConfigurationType is defined to be one of the following values: ctTemperatureMeasurement, ctIntermediateTemperature, ctMeasurementInterval

etANS_Server_Client_Configuration_Update

Dispatched when an ANS Client requests to update Client Configuration Descriptor to a registered ANS Server.

Return Structure:

```
typedef struct
{
   unsigned int                InstanceID;
   unsigned int                ConnectionID;
   GATT_Connection_Type_t      ConnectionType;
   BD_ADDR_t                   RemoteDevice;
   ANS_Characteristic_Type_t   ClientConfigurationType;
   Boolean_t                   NotificationsEnabled;
}ANS_Client_Configuration_Update_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Connection ID of the currently connected remote ANS server device.
ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.
ClientConfigurationType	Identifies the Client Characteristic Configuration Type for which configuration data to be set on remote Device. The ClientConfigurationType is defined to be one of the following values: scNewAlert, scUnreadAlertStatus.
NotificationEnabled	Specifies the Client Configuration descriptor to enable/disable notification. TRUE if CCCD needs to be enabled, FALSE otherwise.

etANS_Server_Control_Point_Command

Dispatched when an ANS Client requests Control Point Command to a registered ANS Server.

Return Structure:

```
typedef struct
{
   unsigned int                     InstanceID;
   unsigned int                     ConnectionID;
   GATT_Connection_Type_t           ConnectionType;
   BD_ADDR_t                        RemoteDevice;
   ANS_Control_Point_Command_t      Command;
   ANS_Category_Identification_t    Category;
} ANS_Server_Control_Point_Command_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Connection ID of the currently connected remote ANS server device
ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.
Command	Identifies Command for which ANS Control Point was configured. The Command is defined to be one of the following values: pcEnable_New_Alert_Notifications, pcEnable_Unread_Category_Notifications, pcDisable_New_Alert_Notifications, pcDisable_Unread_Category_Notifications, pcNotify_New_Alert_Immediately, pcNotify_Unread_Category_Immediately.
Category	Identifies Category for which ANS Control Point was configured. The category is defined to be one of the following values: ciSimpleAlert, ciEmail, ciNews, ciCall, ciMissedCall, ciSMS_MMS, ciVoiceMail, ciSchedule, ciHighPriorityAlert, ciInstantMessage, ciAllCategories.

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
ANSAPI.h	Bluetooth Alert Notification Service (GATT based) API Type Definitions, Constants, and Prototypes.
ANSTYPES.h	Bluetooth Alert Notification Service Types.
SS1BTANS.h	Bluetooth Alert Notification Service Include file.