Battery Service (BAS)

Application Programming Interface Reference Manual

Release: 4.0.1
January 10, 2014

Louisville, KY www.stonestreetone.com

Table of Contents

1. Introduction

Bluetopia®+LE is Stonestreet One's Bluetooth protocol stack that supports the adopted Bluetooth low energy specification. Stonestreet One's upper level protocol stack that supports Single Mode devices is Bluetopia®+LE Single. 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), ATT (Attribute Protocol) Link Layers, the GAP (Generic Attribute Profile) Layer and the Genetic Attribute Protocol (GATT) Layer. In addition to basic functionality of these layers, the Bluetooth Protocol Stack by Stonestreet One provides implementations of the Device Information Service (DIS), BAS (Battery Service), and several of the Bluetooth Profiles. Program access to these layers, services, and profiles is handled via Application Programming Interface (API) calls.

The remainder of this chapter has sections on the scope of this document, other documents applicable to this document, and a listing of acronyms and abbreviations. Chapter 2 is the API reference that contains a description of all programming interfaces for the Battery Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Battery Servicer Profile library.

1.1 Scope

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

- Windows
- Windows Mobile
- Windows CE
- Linux
- QNX
- Other Embedded OS

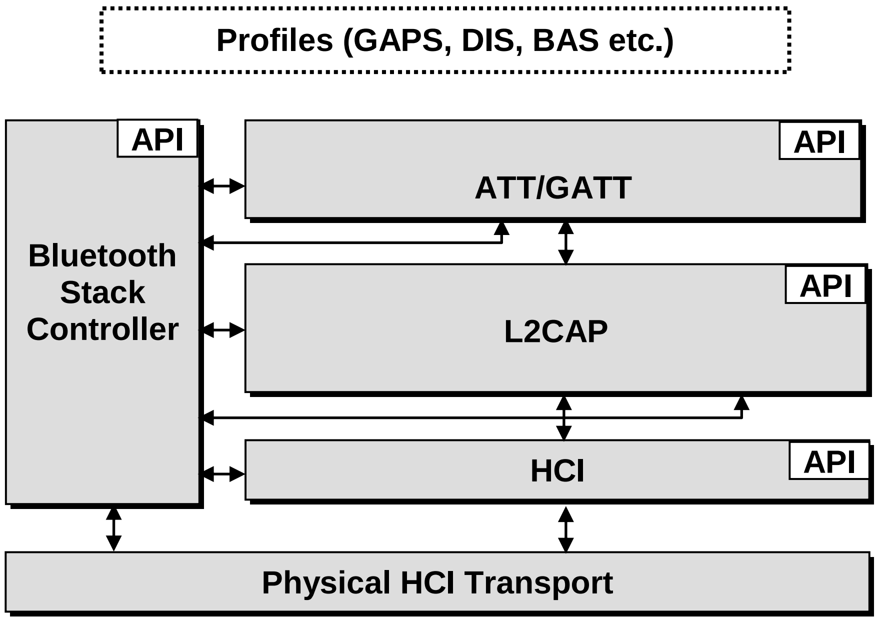

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

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, January 10, 2013.
4. *Bluetooth Doc Battery Service Specification,* version 1.0, December 27, 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
BAS	Battery 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. Battery Service Programming Interfaces

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

2.1 Battery Service Commands

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

Function	Description
BAS_Initialize_Service	Opens a BAS Server.
BAS_Initialize_Service_Handle_Range	Opens a BAS Server with the ability to control the location of the service in the GATT database.
BAS_Cleanup_Service	Closes an opened BAS Server.
BAS_Query_Number_Attributes	Queries the number of attributes
BAS_Read_Client_Configuration_Response	Responds to a read Client Configuration request from the remote device.
BAS_Battery_Level_Read_Request_Response	Responds to a read Battery Level request from the remote device.
BAS_Battery_Level_Read_Request_Error_Response	On Error, Responds to a read Battery Level request from the remote device.
BAS_Notify_Battery_Level	Sends a Battery Level Status notification to the remote device.
BAS_Query_Characteristic_Presentation_Format (It is supported only when multiple service instances exists)	Gets the presentation format of device battery level on the specified BAS Instance.
BAS_Set_Characteristic_Presentation_Format (It is supported only when multiple service instances exists)	Sets the presentation format of device battery level on the specified BAS Instance.
BAS_Decode_Characteristic_Presentation_Format (It is supported only when multiple service instances exists)	Parses the value received from a remote BAS Server interpreting it as characteristic presentation format of Battery Level.

BAS_Initialize_Service

This function opens a BAS Server on a specified Bluetooth Stack.

Prototype:

int BTPSAPI **BAS_Initialize_Service** (unsigned int BluetoothStackID, BAS_Event_Callback_t EventCallback, unsigned long CallbackParameter, unsigned int *ServiceID);

Parameters:

BluetoothStackID	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 BAS service returned from GATT_Register_Service API

Return:

Positive, non-zero if successful. The return value will be the Service Instance ID of BAS 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.

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

BAS_ERROR_INSUFFICIENT_RESOURCES
BAS_ERROR_INVALID_PARAMETER
BAS_ERROR_MAXIMUM_NUMBER_OF_INSTANCES_REACHED
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:

BAS_Initialize_Service_Handle_Range

This function opens a BAS Server on a specified Bluetooth Stack with the ability to control the location of the service in the GATT database.

Prototype:

int BTPSAPI **BAS_Initialize_Service_Handle_Range**(unsigned int BluetoothStackID, BAS_Event_Callback_t EventCallback, unsigned long CallbackParameter, unsigned int *ServiceID, GATT_Attribute_Handle_Group_t *ServiceHandleRange);

Parameters:

BluetoothStackID	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 BAS service returned from GATT_Register_Service API
ServiceHandleRange	Pointer to a Service Handle Range structure, that on input can be used to control the location of the service in the GATT database, and on output returns the handle range that the service is using in the GATT database.

Return:

Positive, non-zero if successful. The return value will be the Service Instance ID of BAS 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.

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

BAS_ERROR_INSUFFICIENT_RESOURCES
BAS_ERROR_INVALID_PARAMETER
BAS_ERROR_MAXIMUM_NUMBER_OF_INSTANCES_REACHED
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:

BAS_Cleanup_Service

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

Prototype:

int BTPSAPI **BAS_Cleanup_Service**(unsigned int BluetoothStackID, unsigned int InstanceID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This is the value that was returned from the BAS_Initialize_Service() function.

Return:

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

BAS_ERROR_INVALID_PARAMETER
BAS_ERROR_INVALID_INSTANCE_ID
BAS_ERROR_INVALID_PARAMETER

Possible Events:

BAS _Query_Number_Attributes

This function is responsible for querying the number of attributes that are contained in the BAS Service that is registered with a call to BAS_Initialize_Service() or BAS_Initialize_Service_Handle_Range().

Prototype:

unsigned int BTPSAPI **BAS_Query_Number_Attributes**(void);

Parameters:

Return:

Positive, non-zero, number of attributes that would be registered by a BAS service instance.

Zero on failure.

Possible Events:

BAS_Read_Client_Configuration_Response

This function is provided to allow a mechanism for a BAS Service to successfully respond to a received read client configuration request.

Prototype:

int BTPSAPI **BAS_Read_Client_Configuration_Response**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int TransactionID, Word_t Client_Configuration);

Parameters:

BluetoothStackID	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 is the value that was returned from the BAS_Initialize_Service() function.
TransactionID	Transaction ID of the original read request. This value was received in the etBAS_Server_Read_Client_Configuration_Request event.
Client_Configuration	Specifies Client Characteristic Configuration descriptor to send to remote device.

Return:

Zero if successful.

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

BAS_ERROR_INVALID_INSTANCE_ID
BAS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED

BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etBAS_Server_Read_Client_Configuration_Request

BAS_Battery_Level_Read_Request_Response

This function is responsible for responding to BAS Read Battery Level Request to remote device.

Prototype:

int BTPSAPI **BAS_Battery_Level_Read_Request_Response**(unsigned int BluetoothStackID, unsigned int TransactionID, Byte_t BatteryLevel)

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the original read request. This value was received in the etBAS_Server_Read_Battery_Level_Request event.
BatteryLevel	Battery Level value to send to the remote Device.

Return:

Zero if successful.

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

BAS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etBAS_Server_Read_Battery_Level_Request

BAS_Battery_Level_Read_Request_Error_Response

This function is responsible for responding to BAS Read Battery Level Request when an error occurred.

Prototype:

int BTPSAPI **BAS_Battery_Level_Read_Request_Error_Response**(unsigned int BluetoothStackID, unsigned int TransactionID, Byte_t ErrorCode);

Parameters:

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

TransactionID	Transaction ID of the original read request. This value was received in the etBAS_Server_Read_Battery_Level_Request.
ErrorCode	ErrorCode occurred during read operation

Return:

Zero if successful.

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

BAS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etBAS_Server_Read_Battery_Level_Request

BAS_Notify_Battery_Level

This function is responsible for sending a Battery Level Status notification to a specified remote device.

Prototype:

int BTPSAPI **BAS_Notify_Battery_Level**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, Byte_t BatteryLevel);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to Notify. This is the value that was returned from the BAS_Initialize_Service() function..
ConnectionID	ConnectionID of the remote device to send the notification to
BatteryLevel	Battery Level value to send to the remote Device.

Return:

Zero if successful.

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

BAS_ERROR_UNKNOWN_ERROR
BAS_ERROR_INVALID_INSTANCE_ID
BAS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

BAS_Query_Characteristic_Presentation_Format

This function is responsible for getting the presentation format of Device battery level on the specified BAS Instance. It is supported only when multiple service instances exists.

Prototype:

int BTPSAPI **BAS_Query_Characteristic_Presentation_Format**(unsigned int BluetoothStackID, unsigned int InstanceID, BAS_Presentation_Format_Data_t *CharacteristicPresentationFormat);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	InstanceID returned from a successful call to BAS_Initialize_Service().
CharacteristicPresentationFormat	A pointer to store the Battery Level presentation format of the specified BAS Instance

Return:

Zero if successful.

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

BAS_ERROR_INVALID_INSTANCE_ID
BAS_ERROR_INVALID_PARAMETER
BTPS_ERROR_FEATURE_NOT_AVAILABLE

Possible Events:

BAS_Set_Characteristic_Presentation_Format

This function is responsible for setting the presentation format of Device battery level on the specified BAS Instance. It is supported only when multiple service instances exists.

Prototype:

int BTPSAPI **BAS_Set_Characteristic_Presentation_Format**(unsigned int BluetoothStackID, unsigned int InstanceID, const BAS_Presentation_Format_Data_t *CharacteristicPresentationFormat);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	InstanceID returned from a successful call to BAS_Initialize_Service().
CharacteristicPresentationFormat	A pointer to store the Battery Level presentation format of the specified BAS Instance

Return:

Zero if successful.

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

BAS_ERROR_INVALID_INSTANCE_ID
BAS_ERROR_INVALID_PARAMETER
BTPS_ERROR_FEATURE_NOT_AVAILABLE

Possible Events:

BAS_Decode_Characteristic_Presentation_Format

This function is responsible for parsing a value received from a remote BAS Server interpreting it as characteristic presentation format of Battery Level. It is supported only when multiple service instances exists.

Prototype:

int BTPSAPI **BAS_Decode_Characteristic_Presentation_Format**(unsigned int ValueLength, Byte_t *Value, BAS_Presentation_Format_Data_t *CharacteristicPresentationFormat);

Parameters:

ValueLength	Specifies the length of the value returned by the remote BAS Server.
Value	Value is a pointer to the data returned by the remote BAS Server.
CharacteristicPresentationFormat	A pointer to store the parsed Battery Level presentation format of the specified BAS Instance

Return:

Zero if successful.

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

BAS_ERROR_INVALID_PARAMETER
BTPS_ERROR_FEATURE_NOT_AVAILABLE

Possible Events:

2.2 Battery Service Event Callback Prototypes

2.2.1 Server Event Callback

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

BAS_Event_Callback_t

Prototype of callback function passed in the BAS_Initialize_Service command.

Prototype:

```
typedef void (BTPSAPI *BAS_Event_Callback_t)(unsigned int BluetoothStackID,
    BAS_Event_Data_t *BAS_Event_Data, unsigned long CallbackParameter);
```

Parameters:

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

```
typedef struct _tagBAS_Event_Data_t
{
  BAS_Event_Type_t Event_Data_Type;
  Word_t           Event_Data_Size;
  union
  {
    BAS_Read_Client_Configuration_Data_t    *BAS_Read_Client_Configuration_Data;
    BAS_Client_Configuration_Update_Data_t  *BAS_Client_Configuration_Update_Data;
    BAS_Read_Battery_Level_Data_t           *BAS_Read_Battery_Level_Data;
  } Event_Data;
} BAS_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:

2.3 Battery Service Events

The Battery Service contains events that are received by the Server. The following sections detail those events.

2.3.1 Battery Service Server Events

The possible Battery Service Server Events from the Bluetooth stack are listed in the table below and are described in the text which follows:

Event	Description
etBAS_Server_Read_Client_Configuration_Request	Dispatched when a BAS Client requests read client configuration to a registered BAS Server.
etBAS_Server_Client_Configuration_Update	Dispatched when a BAS Client requests to update client configuration to a registered BAS Server.
etBAS_Server_Read_Battery_Level_Request	Dispatched when a BAS Client requests read battery

	level to a registered BAS Server.

etBAS_Server_Read_Client_Configuration_Request

Dispatched when a BAS Client requests read client configuration to a registered BAS Server.

Return Structure:

```
typedef struct _tagBAS_Read_Client_Configuration_Data_t
{
   unsigned int                InstanceID;
   unsigned int                ConnectionID;
   unsigned int                TransactionID;
   GATT_Connection_Type_t      ConnectionType;
   BD_ADDR_t                   RemoteDevice;
   BAS_Characteristic_Type_t   ClientConfigurationType;
} BAS_Read_Client_Configuration_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Identifier that uniquely identifies the actual connection of remote device that is making the request.
Transaction ID	Specifies the unique Transaction ID of remote device that is making the 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	Specifies read request type of remote device that is making the request. This value will be ctBatteryLevel only.

etBAS_Server_Client_Configuration_Update

Dispatched when a BAS Client requests to update client configuration to a registered BAS Server.

Return Structure:

```
typedef struct _tagBAS_Client_Configuration_Update_Data_t
{
  unsigned int                InstanceID;
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  BAS_Characteristic_Type_t   ClientConfigurationType;
  Boolean_t                   Notify;
} BAS_Client_Configuration_Update_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Identifier that uniquely identifies the actual connection of remote device that is making the 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	Specifies read request type of remote device that is making the request. This value will be ctBatteryLevel only.
Notify	Specifies new Client Configuration for the specified characteristic ClientConfigurationType. Valid values are TRUE or FALSE.

etBAS_Server_Read_Battery_Level_Request

Dispatched when a BAS Client requests read battery level to the registered BAS Server

Return Structure:

```
typedef struct _tagBAS_Read_Battery_Level_Data_t
{
  unsigned int                InstanceID;
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
} BAS_Read_Battery_Level_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Identifier that uniquely identifies the actual connection of remote device that is making the request.

Transaction ID	Specifies the unique Transaction ID of remote device that is making the request.
ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.

3. File Distributions

The header files that are distributed with the Bluetooth Battery Service Library are listed in the table below.

File	Contents/Description
BASAPI.h	Bluetooth Battery Service (GATT based) API Type Definitions, Constants, and Prototypes.
BASTypes.h	Bluetooth Battery Service Types.
SS1BTBAS.h	Bluetooth Battery Service Include file