Link Loss Service (LLS)

Application Programming Interface Reference Manual

Profile Version: 1.0

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), LLS (Link Loss 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 Link Loss Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Link Loss 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

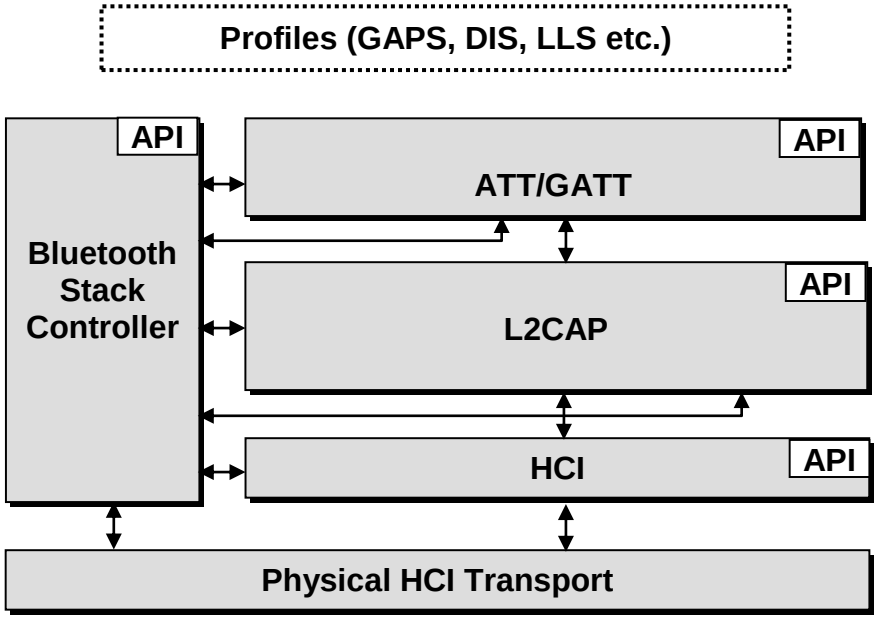

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

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 Link Loss Service Specification,* version v10r00, June 21, 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
LLS	Link Loss 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. Link Loss Service Programming Interfaces

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

2.1 Link Loss Service Commands

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

Function	Description
LLS_Initialize_Service	Opens a LLS Server.
LLS_Initialize_Service_Handle_Range	Opens a LLS Server with the ability to control the location of the service in the GATT database.
LLS_Cleanup_Service	Closes an opened LLS Server.
LLS_Query_Number_Attributes	Queries the number of attributes.
LLS_Set_Alert_Level	Sets the Alert Level on the specified LLS Instance.
LLS_Query_Alert_Level	Gets the Alert Level from the specified LLS Instance.

LLS_Initialize_Service

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

Prototype:

int BTPSAPI **LLS_Initialize_Service** (unsigned int BluetoothStackID,
LLS_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 LLS service returned from GATT_Register_Service API

Return:

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

LLS_ERROR_INSUFFICIENT_RESOURCES
LLS_ERROR_SERVICE_ALREADY_REGISTERED
LLS_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:

LLS_Initialize_Service_Handle_Range

This function is responsible for opening a LLS Server with the ability to control the location of the service in the GATT database.

Prototype:

int BTPSAPI **LLS_Initialize_Service_Handle_Range**(unsigned int BluetoothStackID, LLS_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 LLS 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 LLS 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:
LLS_ERROR_INSUFFICIENT_RESOURCES
LLS_ERROR_SERVICE_ALREADY_REGISTERED
LLS_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:

LLS_Cleanup_Service

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

Prototype:

int BTPSAPI **LLS_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 LLS_Initialize_Service() function.

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

Possible Events:

LLS_Query_Number_Attributes

This function is responsible for querying the number of attributes that are contained in the LLS Service that is registered with a call to LLS_Initialize_Service().

Prototype:

unsigned int BTPSAPI **LLS_Query_Number_Attributes**(void)

Parameters:

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

Possible Events:

LLS_Set_Alert_Level

The following function is responsible for setting the Alert Level on the specified LLS Instance.

Prototype:

int BTPSAPI **LLS_Set_Alert_Level**(unsigned int BluetoothStackID, unsigned int InstanceID, Byte_t Alert_Level);

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 LLS_Initialize_Service() function.
Alert_Level	Alert Level to set on specified LLS Instance.

Return:

Zero if successful.

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

Possible Events:

LLS_Query_Alert_Level

The following function is responsible for querying the Alert Level from the specified LLS Instance.

Prototype:

int BTPSAPI **LLS_Query_Alert_Level**(unsigned int BluetoothStackID, unsigned int InstanceID, Byte_t *Alert_Level);

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 LLS_Initialize_Service() function.
Alert_Level	A pointer to return the Alert Level from specified LLS Instance.

Return:

Zero if successful.

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

LLS_ERROR_INVALID_INSTANCE_ID
LLS_ERROR_INVALID_PARAMETER

Possible Events:

2.2 Link Loss Service Event Callback Prototypes

2.2.1 Server Event Callback

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

LLS_Event_Callback_t

Prototype of callback function passed in the LLS_Initialize_Service command.

Prototype:

typedef void (BTPSAPI ***LLS_Event_Callback_t**)(unsigned int BluetoothStackID,
LLS_Event_Data_t *LLS_Event_Data, unsigned long CallbackParameter);

Parameters:

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

```
typedef struct _tagLLS_Event_Data_t
{
  LLS_Event_Type_t Event_Data_Type;
  Word_t           Event_Data_Size;
  union
  {
    LLS_Alert_Level_Update_Data_t    *LLS_Alert_Level_Update_Data;
  } Event_Data;
} LLS_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 Link Loss Service Events

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

2.3.1 Link Loss Service Server Events

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

Event	Description
etLLS_Alert_Level_Update	Dispatched when a LLS Client requests to update Alert Level to a registered LLS Server.

etLLS_Alert_Level_Update

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

Return Structure:

```
typedef typedef struct _tagLLS_Alert_Level_Update_Data_t
{
  unsigned int                InstanceID;
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Byte_t                      AlertLevel;
} LLS_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.
AlertLevel	Specifies the current Link Loss Alert Level send by Remote Client.

3. File Distributions

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

File	Contents/Description
LLSAPI.h	Bluetooth Link Loss Service (GATT Based) API Type Definitions, Constants, and Prototypes.
LLSTypes.h	Bluetooth Link Loss Service Types.
SS1BTLLS.h	Bluetooth Link Loss Service Include file