Generic Access Profile Service (GAPS)

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), GAPS (Generic Access Profile 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 Generic Access Profile Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Generic Access 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
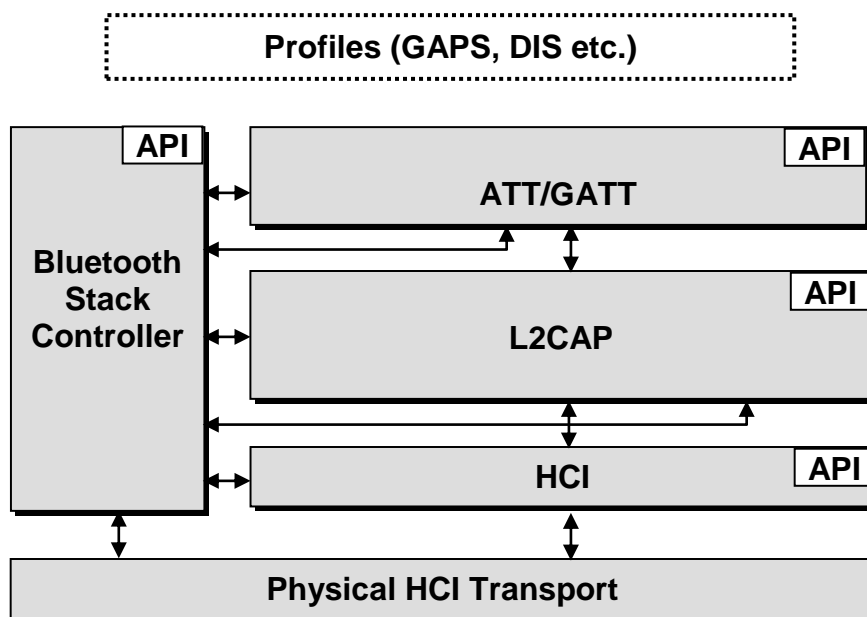
Figure 1-1   The Stonestreet One Bluetooth Protocol Stack

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.

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
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. Generic Access Profile Service Programming Interfaces

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

2.1   Generic Access Profile Service Commands

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

Function	Description
GAPS_Initialize_Service	Opens a GAPS Server.
GAPS_Initialize_Service_Handle_Range	Opens a GAPS Server with the ability to control the location of the service in the GATT database..
GAPS_Cleanup_Service	Closes an opened GAPS Server.
GAPS_Query_Number_Attributes	Queries the number of attributes.
GAPS_Set_Device_Name	Sets the Device name characteristic of generic device on the specified GAP Service Instance.
GAPS_Query_Device_Name	Gets the current device name characteristic of generic device from the specified GAP Service Instance.
GAPS_Set_Device_Appearance	Sets the Device Appearance characteristic of generic device on the specified GAP Service Instance.
GAPS_Query_Device_Appearance	Gets the current device appearance characteristic of generic device from the specified GAP Service Instance.
GAPS_Set_Preferred_Connection_Parameters	Sets the Peripheral Preferred Connection Parameter characteristic of generic device on the specified GAP Service instance.
GAPS_Query_Preferred_Connection_Parameters	Gets the Peripheral Preferred Connection Parameter characteristic of generic device from the specified GAP Service instance.
GAPS_Decode_Preferred_Connection_Parameters	Parses a Peripheral Preferred Connection Parameter characteristic value that was received from a remote device.

GAPS_Initialize_Service

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

Prototype:

int BTPSAPI **GAPS_Initialize_Service** (unsigned int BluetoothStackID, unsigned int
  *ServiceID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServiceID	Unique GATT Service ID of the registered GAPS service returned from GATT_Register_Service API

Return:

Positive, non-zero if successful. The return value will be the Service Instance ID of
GAPS 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:
> GAPS_ERROR_INSUFFICIENT_RESOURCES
> GAPS_ERROR_INVALID_PARAMETER
> GAPS_ERROR_SERVICE_ALREADY_REGISTERED
> 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:

GAPS_Initialize_Service_Handle_Range

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

Prototype:

int BTPSAPI **GAPS_Initialize_Service_Handle_Range**(unsigned int BluetoothStackID,
  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.
ServiceID	Unique GATT Service ID of the registered GAPS 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 GAPS 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:
> GAPS_ERROR_INSUFFICIENT_RESOURCES
> GAPS_ERROR_INVALID_PARAMETER
> GAPS_ERROR_SERVICE_ALREADY_REGISTERED
> 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:

GAPS_Cleanup_Service

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

Prototype:

int BTPSAPI **GAPS_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 GAPS_Initialize_Service() function.

Return: Zero if successful.

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

Possible Events:

GAPS_Query_Number_Attributes

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

Prototype:

unsigned int BTPSAPI **GAPS_Query_Number_Attributes**(void)

Parameters:

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

GAPS _ERROR_INVALID_PARAMETER
GAPS _ERROR_INVALID_INSTANCE_ID

Possible Events:

GAPS_Set_Device_Name

This function is responsible for Setting the Device name characteristic of generic device on the specified GAP Service Instance.

Prototype:

int BTPSAPI **GAPS_Set_Device_Name**(unsigned int BluetoothStackID, unsigned int InstanceID, char *DeviceName);

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 GAPS_Initialize_Service() function.
DeviceName	The Device Name to set as the current Device Name for the specified GAP Service Instance.

Return:

Zero if successful.

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

GAPS_ERROR_INVALID_PARAMETER
GAPS_ERROR_INVALID_INSTANCE_ID

Possible Events:

GAPS_Query_Device_Name

This function is responsible for querying the current device name characteristic of generic device from the specified GAP Service Instance.

Prototype:

int BTPSAPI **GAPS_Query_Device_Name**(unsigned int BluetoothStackID, unsigned int InstanceID, char *NameBuffer);

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 GAPS_Initialize_Service() function.
NameBuffer	A pointer to a structure to return the current Device for the specified GAPS Service Instance. The NameBuffer Length should be at least (GAP_MAXIMUM_DEVICE_NAME_LENGTH+1) to hold the Maximum allowable Name (plus a single character to hold the NULL terminator).

Return:

Zero if successful.

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

Possible Events:

GAPS_Set_Device_Appearance

This function is responsible for setting the Device Appearance characteristic of generic device on the specified GAP Service Instance.

Prototype:

int BTPSAPI **GAPS_Set_Device_Appearance**(unsigned int BluetoothStackID, unsigned int InstanceID, Word_t DeviceAppearance);

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 GAPS_Initialize_Service() function.
DeviceAppearance	The Device Appearance is to set as the current Device Appearance for the specified GAP Service Instance.

Return:

Zero if successful.

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

Possible Events:

GAPS_Query_Device_Appearance

This function is responsible for querying the current device appearance characteristic of generic device from the specified GAP Service Instance.

Prototype:

int BTPSAPI **GAPS_Query_Device_Appearance**(unsigned int BluetoothStackID, unsigned int InstanceID, Word_t *DeviceAppearance);

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 GAPS_Initialize_Service() function.
DeviceAppearance	A pointer to store the current Device Appearance for the specified GAP Service Instance.

Return:

Zero if successful.

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

Possible Events:

GAPS_Set_Preferred_Connection_Parameters

This function is responsible for setting the Peripheral Preferred Connection Parameter characteristic of generic device on the specified GAP Service instance.

Prototype:

int BTPSAPI **GAPS_Set_Preferred_Connection_Parameters**(unsigned int BluetoothStackID, unsigned int InstanceID, GAP_Preferred_Connection_Parameters_t *PreferredConnectionParameters);

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 GAPS_Initialize_Service() function.

PreferredConnectionparameters   The Preferred Connection Parameters to set as the current Peripheral Preferred Connection Parameters for the specified GAP Service Instance.

Return:

Zero if successful.

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

Possible Events:

GAPS_Query_Preferred_Connection_Parameters

This function is responsible for querying the Peripheral Preferred Connection Parameter characteristic of generic device from the specified GAP Service instance.

Prototype:

int BTPSAPI **GAPS_Query_Preferred_Connection_Parameters**(unsigned int BluetoothStackID, unsigned int InstanceID, GAP_Preferred_Connection_Parameters_t *PreferredConnectionParameters);

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 GAPS_Initialize_Service() function.

PreferredConnectionparameters   A pointer to a structure to store the current Peripheral Preferred Connection Parameters for the specified GAP Service Instance.

Return:

Zero if successful.

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

Possible Events:

GAPS_Decode_Preferred_Connection_Parameters

This function is responsible for decoding a Peripheral Preferred Connection Parameter characteristic value that was received from a remote device.

Prototype:

>   int BTPSAPI **GAPS_Decode_Preferred_Connection_Parameters**(unsigned int
>   ValueLength, Byte_t *Value, GAP_Preferred_Connection_Parameters_t
>   *PreferredConnectionParameters);

Parameters:

>   ValueLength   Specifies the length of the Preferred Connection
>   Parameter value returned by the remote GAPS Server.
>
>   Value   Value is a pointer to the Preferred Connection
>   Parameter data returned by the remote GAPS Server.
>
>   PreferredConnectionparameters   A pointer to a structure to store the decoded Peripheral
>   Preferred Connection Parameters value that was
>   received from the remote device.

Return:

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

Possible Events:

3. File Distributions

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

File	Contents/Description
GAPSAPI.h	Bluetooth Generic Access Profile Service (GATT based) API Type Definitions, Constants, and Prototypes
GAPSType.h	Generic Access Profile Service Types.
SS1BTGAP.h	Bluetooth Generic Access Profile Service Include file