Human Interface Device Profile (HID)

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®, 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 Human Interface Device Profile provided by Bluetopia. Chapter 2 contains a description of the programming interfaces for this profile. And, Chapter 3 contains the header file name list for the Bluetooth Human Interface Device Profile library.

1.1   Scope

This reference manual provides information on the Human Interface Device Profile API. This API is available on the full range of platforms supported by Stonestreet One:

- Windows
- Linux

- Windows Mobile
- QNX

- Windows CE
- Other Embedded OS
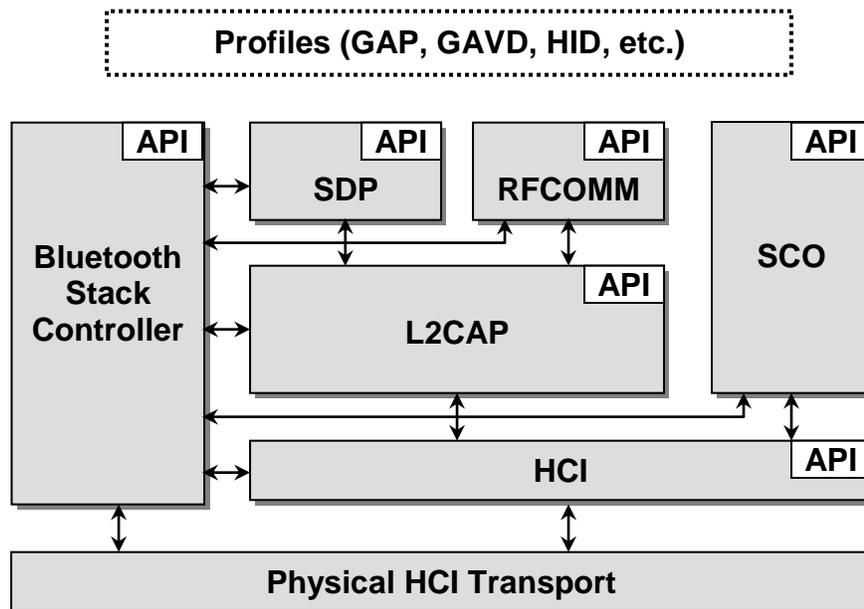
Figure 1-1   The Stonestreet One Bluetooth Protocol Stack

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, Core*, version 1.1, February 22, 2001.

2. *Specification of the Bluetooth System, Volume 2, Profiles*, version 1.1, February 22, 2001.

3. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 2.0 + EDR, November 4, 2004.

4. *Specification of the Bluetooth System, Volume 2, Core System Package*, version 2.0 + EDR, November 4, 2004.

5. *Specification of the Bluetooth System, Volume 3, Core System Package*, version 2.0 + EDR, November 4, 2004.

6. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 2.1+EDR, July 26, 2007.

7. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 2.1+EDR, July 26, 2007.

8. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 2.1+EDR, July 26, 2007.

9. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 2.1+EDR, July 26, 2007.

10. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 2.1+EDR, July 26, 2007.

11. *Specification of the Bluetooth System, Bluetooth Core Specification Addendum 1*, June 26, 2008.

12. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 3.0+HS, April 21, 2009.

13. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 3.0+HS, April 21, 2009.

14. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 3.0+HS, April 21, 2009.

15. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 3.0+HS, April 21, 2009.

16. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 3.0+HS, April 21, 2009.

17. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 3.0+HS, April 21, 2009.

18. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 4.0, June 30, 2010.

19. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 4.0, June 30, 2010.

20. *Specification of the Bluetooth System, Volume 2, Core System Package [BR/EDR Controller Volume]*, version 4.0, June 30, 2010.

21. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 4.0, June 30, 2010.

22. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 4.0, June 30, 2010.

23. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 4.0, June 30, 2010.

24. *Specification of the Bluetooth System, Volume 6, Core System Package [Low Energy Controller Volume]*, version 4.0, June 30, 2010.

25. *Bluetooth Human Interface Device (HID) Profile*, version 0.95c, May 5, 2002.

26. *Universal Serial Bus Specification*, version 1.1, September 23, 1998.

27. *Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID)*, version 1.11, June 27, 2001.

28. *Universal Serial Bus (USB) HID Usage Tables*, version 1.11, June 27, 2001.

29. *Universal Serial Bus (USB) Device Class Definition for Physical Interface Devices (PID)*, version 1.0, September 8, 1999.

30. *Universal Serial Bus (USB) Language Identifiers (LANGIDs)*, version 1.0, March 29, 2000.

31. *Bluetooth Assigned Numbers,* version 1.1, February 22, 2001.

32. *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
BD_ADDR	Bluetooth Device Address

Term	Meaning
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Date Rate
HS	High Speed
LE	Low Energy
LSB	Least Significant Bit
MSB	Most Significant Bit
SDP	Service Discovery Protocol
SPP	Serial Port Protocol
HID	Human Interface Device
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. HID Profile Programming Interface

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

2.1   HID Profile Commands

The available Human Interface Device Profile command functions are listed in the table below and are described in the text that follows.

Function	Description
HID_Register_Host_Server	Register a Bluetooth HID Host Server.
HID_Register_Device_Server	Register a Bluetooth HID Device Server.
HID_Open_Request_Response	Responds to individual request to connect to local HID server.
HID_Un_Register_Server	Un-Register the Registered Server.
HID_Register_Device_SDP_Record	Add a Generic Bluetooth HID Device Service Record to the SDP Database.
HID_Register_Device_SDP_Record_Version	Add a Generic Bluetooth HID Device Service Record of a specified HID profile version to the SDP Database.
HID_Un_Register_Device_SDP_Record	Remove a Generic Bluetooth HID Device Service Record from the SDP Database.
HID_Connect_Remote_Device	Open a connection to a Remote Bluetooth HID Device.
HID_Connect_Remote_Host	Open a connection to a Remote Bluetooth HID Host.
HID_Close_Connection	Close a previously Opened Connection to a Remote Bluetooth HID Host or Device.
HID_Control_Request	Send a HID_CONTROL Transaction to the Remote Entity.
HID_Get_Report_Request	Send a GET_REPORT Transaction to a Remote Bluetooth HID Device.
HID_Get_Report_Response	Send a response to an outstanding GET_REPORT Transaction to a Remote Bluetooth HID Host.
HID_Set_Report_Request	Send a SET_REPORT Transaction to a Remote Bluetooth HID Device.

HID_Set_Report_Response	Send a response to an outstanding SET_REPORT Transaction to a Remote Bluetooth HID Host.
HID_Get_Protocol_Request	Send a GET_PROTOCOL Transaction to a Remote Bluetooth HID Device.
HID_Get_Protocol_Response	Send a response to an outstanding GET_PROTOCOL Transaction to a Remote Bluetooth HID Host.
HID_Set_Protocol_Request	Send a SET_PROTOCOL Transaction to a Remote Bluetooth HID Device.
HID_Set_Protocol_Response	Send a response to an outstanding SET_PROTOCOL Transaction to a Remote Bluetooth HID Host.
HID_Get_Idle_Request	Send a GET_IDLE Transaction to a Remote Bluetooth HID Device.
HID_Get_Idle_Response	Send a response to an outstanding GET_IDLE Transaction to a Remote Bluetooth HID Host.
HID_Set_Idle_Request	Send a SET_IDLE Transaction to a Remote Bluetooth HID Device.
HID_Set_Idle_Response	Send a response to an outstanding SET_IDLE Transaction to a Remote Bluetooth HID Host.
HID_Data_Write	Send Report Data over the Interrupt Channel to the Remote Device.
HID_Get_Server_Connection_Mode	Retrieves the current HID Server Connection Mode.
HID_Set_Server_Connection_Mode	Sets the HID Server Connection Mode.

HID_Register_Host_Server

The following function is responsible for registering a Bluetooth HID Host Server. Note that only one Server can be registered for an individual Bluetooth Stack instance.

Prototype:

int BTPSAPI **HID_Register_Host_Server**(unsigned int BluetoothStackID, HID_Configuration_t *HIDConfiguration, HID_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

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

HIDConfiguration   The configuration specification to be used in the negotiation of the L2CAP channels associated with this Bluetooth HID Host Server, defined by the following structure.

　　　　　　　　typedef struct

```
                              {
                                HID_Flow_Config_t  OutFlow;
                                HID_Flow_Config_t  InFlow;
                                Word_t             InMTU;
                              } HID_Configuration_t;
```

where flow members are used for negotiating the Interrupt Channel and the MTU member is used for negotiating both Control and Interrupt Channels.

The OutFlow member defines the minimum and maximum flow configuration in which the entity can operate properly.

The InFlow member defines the maximum flow configuration in which this entity can support. The minimum flow configuration need not be set as it is ignored.

The InMTU member specifies the Maximum Transmission Unit that will be negotiated for incoming data.

EventCallback   Function to call when events occur on this server.

CallbackParameter   A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Zero if successful. The HIDID to be used in function calls associated with connections made on this server will be returned in the Open Indication Event via the Callback associated with this Server.

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

> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION
> BTHID_ERROR_INSUFFICIENT_RESOURCES
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_INVALID_PARAMETER

Possible Events:

etHID_Open_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.

HID_Register_Device_Server

The following function is responsible for registering a Bluetooth HID Device Server. Note that only one Server can be registered for an individual Bluetooth Stack instance.

Prototype:

> int BTPSAPI **HID_Register_Device_Server**(unsigned int BluetoothStackID,
>   HID_Configuration_t *HIDConfiguration, HID_Event_Callback_t EventCallback,
>   unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]

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

HIDConfiguration

The configuration specification to be used in the negotiation of the L2CAP channels associated with this Bluetooth HID Device Server, defined by the following structure.

```
typedef struct
{
  HID_Flow_Config_t  OutFlow;
  HID_Flow_Config_t  InFlow;
  Word_t             InMTU;
} HID_Configuration_t;
```

where flow members are used for negotiating the Interrupt Channel and the MTU member is used for negotiating both Control and Interrupt Channels.

The OutFlow member defines the minimum and maximum flow configuration in which the entity can operate properly.

The InFlow member defines the maximum flow configuration in which this entity can support. The minimum flow configuration need not be set as it is ignored.

The InMTU member specifies the Maximum Transmission Unit that will be negotiated for incoming data.

EventCallback

Function to call when events occur on this server.

CallbackParameter

A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Zero if successful. The HIDID to be used in function calls associated with connections made on this server will be returned in the Open Indication Event via the Callback associated with this Server.

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

> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION
> BTHID_ERROR_INSUFFICIENT_RESOURCES
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_INVALID_PARAMETER

Possible Events:

etHID_Open_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.

HID_Open_Request_Response

This function is responsible for responding to an individual request to connect to connect to a local hid server to respond to etHID_Open_Request_Indication

Prototype:

int BTPSAPI **HID_Open_Request_Response**(unsigned int BluetoothStackID, unsigned int HIDID, Boolean_t AcceptConnection)

Parameters:

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

HIDID　　　　　　　　　　　Unique HID ID of the HID Server for which an open request was received

AcceptConnection　　　　　　Boolean specifying whether to accept the pending connection request (TRUE to accept).

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_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.

HID_Un_Register_Server

This function is responsible for Un-Registering a server registered by a successful call to either the HID_Register_Host_Server() function or the HID_Register_Device_Server() function. Note, this function does NOT delete any Service Record Handles (i.e., added via an HID_Register_Device_SDP_Record( ) function call).

Prototype:

int BTPSAPI **HID_Un_Register_Server**(unsigned int BluetoothStackID)

Parameters:

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

Return:

Zero if successful.

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

                        BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
                        BTHID_ERROR_NOT_INITIALIZED
                        BTHID_ERROR_INVALID_OPERATION
                        BTHID_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.

HID_Register_Device_ SDP_Record

This function adds a generic Bluetooth HID Device Service Record to the SDP database.

Notes:

1. 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 HID_Un_Register_Device_SDP_Record( ) to
    SDP_Delete_Service_Record(), and is defined as follows:

    **HID_Un_Register_Device_SDP_Record(__BluetoothStackID, __HIDID,
    __SDPRecordHandle) (SDP_Delete_Service_Record(__BluetoothStackID,
    __SDPRecordHandle))**

2. 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.

3. A HID LANGID Base List Attribute ID is created that specifies 0x0100, English
    (United States).

4. A HID Country Code Attribute ID is created that specifies that the hardware is not
    localized (a value of zero).

Prototype:

int BTPSAPI **HID_Register_Device_SDP_Record**(unsigned int BluetoothStackID,
    unsigned long DeviceFlags, Word_t DeviceReleaseNumber, Word_t ParserVersion,

Byte_t DeviceSubclass, unsigned int NumberDescriptors,
HID_Descriptor_t DescriptorList[], char *ServiceName,
DWord_t *SDPServiceRecordHandle)

Parameters:

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

DeviceFlags   Bit Field used to define miscellaneous HID Profile SDP
                             Attributes. The following bit constants are currently defined
                             for use with this parameter.

                                 HID_VIRTUAL_CABLE_BIT
                                 HID_RECONNECT_INITIATE_BIT
                                 HID_SDP_DISABLE_BIT
                                 HID_BATTERY_POWER_BIT
                                 HID_REMOTE_WAKE_BIT
                                 HID_NORMALLY_CONNECTABLE_BIT
                                 HID_BOOT_DEVICE_BIT

DeviceReleaseNumber   The Device Release Number to be used with the HID Device
                             Release Number SDP Attribute.

ParserVersion   The Parser Version to be used with the HID Parser Version
                             SDP Attribute.

DeviceSubclass   The Device Subclass to be used with the HID Device Subclass
                             SDP Attribute.

NumberDescriptors   The Number of Descriptors that appear in the Descriptor List
                             parameter.

DescriptorList   The List of Descriptors to be used with the HID Descriptor List
                             Attribute. An individual descriptor is defined by the following
                             structure.

                                 typedef struct
                                 {
                                   Byte_t   DescriptorType;
                                   Word_t   DescriptorLength;
                                   Byte_t   *Descriptor;
                                 } **HID_Descriptor_t**;

                             where DescriptorType specifies the Class Descriptor Type as
                             defined in the Bluetooth Human Interface Device (HID)
                             Profile.

                             The DataLength member specifies the Length of the Descriptor
                             member that follows.

                             The Descriptor member specifies the actual Class Descriptor
                             Data to use.

ServiceName	The Service Name to be used with the Service Name SDP Attribute.
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:

> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

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.

HID_Register_Device_SDP_Record_Version

This function adds a generic Bluetooth HID Device Service Record to the SDP database of the specified HID version.

Notes:

1. 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 HID_Un_Register_Device_SDP_Record( ) to SDP_Delete_Service_Record(), and is defined as follows:

   HID_Un_Register_Device_SDP_Record(__BluetoothStackID, __HIDID, __SDPRecordHandle) (SDP_Delete_Service_Record(__BluetoothStackID, __SDPRecordHandle))

2. 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.

3. A HID LANGID Base List Attribute ID is created that specifies 0x0100, English (United States).

4. A HID Country Code Attribute ID is created that specifies that the hardware is not localized (a value of zero).

Prototype:

int BTPSAPI **HID_Register_Device_SDP_Record_Version**(
   unsigned int BluetoothStackID, unsigned long DeviceFlags,

Word_t DeviceReleaseNumber, Word_t ParserVersion, Byte_t DeviceSubclass,
unsigned int NumberDescriptors, HID_Descriptor_t DescriptorList[],
char *ServiceName, HID_Version_t HIDVersion,
HID_SDP_Record_Information_t *RecordInformation,
DWord_t *SDPServiceRecordHandle)

Parameters:

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

DeviceFlags   Bit Field used to define miscellaneous HID Profile SDP
                           Attributes. The following bit constants are currently defined
                           for use with this parameter.

                                HID_VIRTUAL_CABLE_BIT
                                HID_RECONNECT_INITIATE_BIT
                                HID_SDP_DISABLE_BIT
                                HID_BATTERY_POWER_BIT
                                HID_REMOTE_WAKE_BIT
                                HID_NORMALLY_CONNECTABLE_BIT
                                HID_BOOT_DEVICE_BIT

DeviceReleaseNumber   The Device Release Number to be used with the HID Device
                           Release Number SDP Attribute.

ParserVersion   The Parser Version to be used with the HID Parser Version
                           SDP Attribute.

DeviceSubclass   The Device Subclass to be used with the HID Device Subclass
                           SDP Attribute.

NumberDescriptors   The Number of Descriptors that appear in the Descriptor List
                           parameter.

DescriptorList   The List of Descriptors to be used with the HID Descriptor List
                           Attribute. An individual descriptor is defined by the following
                           structure.

                                typedef struct
                                {
                                  Byte_t   DescriptorType;
                                  Word_t   DescriptorLength;
                                  Byte_t   *Descriptor;
                                } **HID_Descriptor_t**;

                           where DescriptorType specifies the Class Descriptor Type as
                           defined in the Bluetooth Human Interface Device (HID)
                           Profile.

                           The DataLength member specifies the Length of the Descriptor
                           member that follows.

                           The Descriptor member specifies the actual Class Descriptor
                           Data to use.

ServiceName   The Service Name to be used with the Service Name SDP
                               Attribute.

HIDVersion   The HID profile version of the SDP record to register. Valid
                               values are the following:
                                   hpvVersion1_0
                                   hpvVersion1_1

RecordInformation   An optional pointer to a structure of additional information to
                               publish in the SDP record. This structure is defined as follows:
                                   typedef struct
                                   {
                                     unsigned long   Flags;
                                     HID_Sniff_Subrating_Parameters_t
                                                              SniffSubratingParameters;
                                     Word_t   LinkSupervisionTimeout;
                                   } **HID_SDP_Record_Information_t**;

                               The Flags member is a bitmask variable which denotes which
                               members of the structure are valid. The valid bits are as
                               follow:

  HID_SDP_RECORD_INFORMATION_FLAGS_SNIFF_SUBRATING_VALID
  HID_SDP_RECORD_INFORMATION_FLAGS_SUPERVISION_TIMEOUT_VALID

                               Based on the bits that are set the appropriate member of the
                               containing structure must contain valid data.

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:

                               BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
                               BTHID_ERROR_INVALID_PARAMETER
                               BTHID_ERROR_NOT_INITIALIZED
                               BTHID_ERROR_INVALID_OPERATION

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.

HID_Connect_Remote_Device

This function is used to open a connection to a remote Bluetooth HID Device.

Prototype:

> int BTPSAPI **HID_Connect_Remote_Device**(unsigned int BluetoothStackID,
> BD_ADDR_t BD_ADDR, HID_Configuration_t *HIDConfiguration,
> HID_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   Board Address of the Remote Bluetooth HID Device to connect with.

HIDConfiguration   The configuration specification to be used in the negotiation of the L2CAP channels associated with this Bluetooth HID Host Client, defined by the following structure.

> ```
> typedef struct
> {
>   HID_Flow_Config_t  OutFlow;
>   HID_Flow_Config_t  InFlow;
>   Word_t             InMTU;
> } HID_Configuration_t;
> ```

where flow members are used for negotiating the Interrupt Channel and the MTU member is used for negotiating both Control and Interrupt Channels.

The OutFlow member defines the minimum and maximum flow configuration in which the entity can operate properly.

The InFlow member defines the maximum flow configuration in which this entity can support. The minimum flow configuration need not be set as it is ignored.

The InMTU member specifies the Maximum Transmission Unit that will be negotiated for incoming data.

EventCallback   Function to call when events occur on this connection.

CallbackParameter   A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

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

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INSUFFICIENT_RESOURCES

Possible Events:

etHID_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.

HID_Connect_Remote_Host

This function is used to open a connection to a remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Connect_Remote_Host**(unsigned int BluetoothStackID,
        BD_ADDR_t BD_ADDR, HID_Configuration_t *HIDConfiguration,
        HID_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   Board Address of the remote Bluetooth HID Host to connect
                             with.

HIDConfiguration   The configuration specification to be used in the negotiation of
                             the L2CAP channels associated with this Bluetooth HID
                             Device Client, defined by the following structure.

```
typedef struct
{
  HID_Flow_Config_t  OutFlow;
  HID_Flow_Config_t  InFlow;
  Word_t             InMTU;
} HID_Configuration_t;
```

                             where flow members are used for negotiating the Interrupt
                             Channel and the MTU member is used for negotiating both
                             Control and Interrupt Channels.

                             The OutFlow member defines the minimum and maximum
                             flow configuration in which the entity can operate properly.

                             The InFlow member defines the maximum flow configuration
                             in which this entity can support. The minimum flow
                             configuration need not be set as it is ignored.

                             The InMTU member specifies the Maximum Transmission
                             Unit that will be negotiated for incoming data.

EventCallback   Function to call when events occur on this connection.

CallbackParameter   A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

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

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INSUFFICIENT_RESOURCES

Possible Events:

etHID_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.

HID_Close_Connection

This function is used to close a previously opened connection to a remote Bluetooth HID Host or Device. This function is capable of closing connections on a registered Bluetooth HID Host Server or Bluetooth HID Device Server without un-registering the server. This function may also be used to close connections established through calls to the HID_Connect_Remote_Device() function or the HID_Connect_Remote_Host() function.

Prototype:

int BTPSAPI **HID_Close_Connection**(unsigned int BluetoothStackID, unsigned int HIDID)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the connection functions for Clients.

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED

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.

HID_Control_Request

This function is used to send a HID_CONTROL Transaction to the remote entity.

Notes:

1. Bluetooth HID Devices are only capable of sending HID_CONTROL Transactions with a Control Operation value of hcVirtualCableUnplug. All other Control Operations when performed by a Bluetooth HID Device will return an error result.

2. Transactions on the Control Channel normally consist of two phases, a Request by the Host and a Response by the Device. However, HID_CONTROL transactions require no Response phase. Also note that HID Control Requests are not allowed while other transactions are being processed unless the Control Operation Type is hcVirtualCableUnplug, which may be sent at any time.

Prototype:

int BTPSAPI **HID_Control_Request**(unsigned int BluetoothStackID, unsigned int HIDID, HID_Control_Operation_Type_t ControlOperation)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

ControlOperation   The Control Operation Type to be sent as the only parameter of a HID_CONTROL Transaction. The following Control Operations are currently defined.

> hcNop
> hcHardReset
> hcSoftReset
> hcSuspend
> hcExitSuspend
> hcVirtualCableUnplug

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER

BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_ERROR_NOT_INITIALIZED
BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Get_Report_Request

This function is used to send a GET_REPORT Transaction to a remote Bluetooth HID Device.

Notes:

1. Control Channel transfers have two phases, a Request by the host and a Response by the device. Only ONE host control channel Request shall be outstanding at a time. Reception of a HID Get Report Confirmation event indicates that a Response has been received and the Control Channel is now free for further Transactions.

Prototype:

int BTPSAPI **HID_Get_Report_Request**(unsigned int BluetoothStackID, unsigned int HIDID, HID_Get_Report_Size_Type_t Size, HID_Report_Type_Type_t ReportType, Byte_t ReportID, Word_t BufferSize)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
Size	Description that indicates how the device is to determine the size of the response report buffer that the host has allocated. The following Get Report Size Types are currently defined. grSizeOfReport grUseBufferSize
ReportType	The Report Type of the Report in which this GET_REPORT Transaction is requesting. The following Report Types are valid for this parameter in this function. rtInput rtOutput

	rtFeature
ReportID	The Report ID of the Report in which this GET_REPORT Transaction is requesting. To exclude this information from the GET_REPORT Transaction the following constant may be used for this parameter.
	HID_INVALID_REPORT_ID
BufferSize	The Buffer Size in which the host has allocated for the response report buffer. This will be the maximum number of bytes that should be received in the response phase of this transaction. This parameter will only be included in the GET_REPORT Transaction if the Size parameter to this function is set to grUseBufferSize.

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_Indication
etHID_Get_Report_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.

HID_Get_Report_Response

This function is used to send a response to an outstanding GET_REPORT Transaction to a remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Get_Report_Response**(unsigned int BluetoothStackID,
    unsigned int HIDID, HID_Result_Type_t ResultType,
    HID_Report_Type_Type_t ReportType, Word_t ReportPayloadSize,
    Byte_t *ReportDataPayload)

Parameters:

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

HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
ResultType	The Result Type for this Response. The following Result Type is invalid for use with this function.

       rtSuccessful

The following Result Types will send a HANDSHAKE Transaction as the response to the outstanding GET_REPORT Transaction. The Result Code parameter of the HANDSHAKE Transaction will be of the type indicated by the Result Type.

       rtNotReady
       rtErrInvalidReportID
       rtErrUnsupportedRequest
       rtErrInvalidParameter
       rtErrUnknown
       rtErrFatal

The following Result Type will send a DATA Transaction in response to the outstanding GET_REPORT Transaction.

       rtData

ReportType	The Report Type of the Report being sent as the response to a GET_REPORT Transaction. This parameter is only used when the Result Type parameter is set to rtData. The following Report Types are valid for this parameter in this function.

       rtInput
       rtOutput
       rtFeature

ReportPayloadSize	The Size of the Report to which the Report Data Payload parameter points. This parameter is only used when the Result Type parameter is set to rtData.
ReportDataPayload	Pointer to the Report Data to be sent. This parameter is only used when the Result Type parameter is set to rtData.

Return:

Zero if successful.

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

                    BTHID_ERROR_INVALID_PARAMETER
                    BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
                    BTHID_ERROR_NOT_INITIALIZED
                    BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Set_Report_Request

This function is used to send a SET_REPORT Transaction to a remote Bluetooth HID Device.

Notes:

1. Control Channel transfers have two phases, a Request by the host and a Response by the device. Only ONE host control channel Request shall be outstanding at a time. Reception of a HID Set Report Confirmation event indicates that a Response has been received and the Control Channel is now free for further Transactions.

Prototype:

int BTPSAPI **HID_Set_Report_Request**(unsigned int BluetoothStackID,
    unsigned int HIDID, HID_Report_Type_Type_t ReportType,
    Word_t ReportPayloadSize, Byte_t *ReportDataPayload)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
ReportType	The Report Type of the Report being sent as part of the SET_REPORT Transaction. The following Report Types are valid for this parameter in this function. 　　rtInput 　　rtOutput 　　rtFeature
ReportPayloadSize	The Size of the Report to which the Report Data Payload parameter points.
ReportDataPayload	Pointer to the Report Data to be sent as part of the SET_REPORT Transaction.

Return:

Zero if successful.

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

BTHID_ERROR_INVALID_PARAMETER

Stonestreet One   Page 25 of 55

BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_ERROR_NOT_INITIALIZED
BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_Indication
etHID_Set_Report_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.

HID_Set_Report_Response

This function is used to send a response to an outstanding SET_REPORT Transaction to a remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Set_Report_Response**(unsigned int BluetoothStackID,
    unsigned int HIDID, HID_Result_Type_t ResultType)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
ResultType	The Result Type for this Response. The following Result Type is invalid for use with this function.

rtData

The following Result Types will send a HANDSHAKE Transaction as the response to the outstanding SET_REPORT Transaction. The Result Code parameter of the HANDSHAKE Transaction will be of the type indicated by the Result Type.

rtSuccessful
rtNotReady
rtErrInvalidReportID
rtErrUnsupportedRequest
rtErrInvalidParameter
rtErrUnknown
rtErrFatal

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Get_Protocol_Request

This function is used to send a GET_PROTOCOL Transaction to a remote Bluetooth HID Device.

Notes:

1. Control Channel transfers have two phases, a Request by the host and a Response by the device. Only ONE host control channel Request shall be outstanding at a time. Reception of a HID Get Protocol Confirmation event indicates that a Response has been received and the Control Channel is now free for further Transactions.

Prototype:

int BTPSAPI **HID_Get_Protocol_Request**(unsigned int BluetoothStackID, unsigned int HIDID)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED

BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_Indication
etHID_Get_Protocol_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.

HID_Get_Protocol_Response

This function is used to send a response to an outstanding GET_PROTOCOL Transaction to a remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Get_Protocol_Response**(unsigned int BluetoothStackID,
    unsigned int HIDID, HID_Result_Type_t ResultType, HID_Protocol_Type_t Protocol)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be
                             performed with. This is the value that was received via an
                             Open Indication Event for Servers or from the return value of
                             the Connection Functions for Clients.

ResultType   The Result Type for this Response. The following Result Type
                             is invalid for use with this function.

                                 rtSuccessful

                             The following Result Types will send a HANDSHAKE
                             Transaction as the response to the outstanding
                             GET_PROTOCOL Transaction. The Result Code parameter of
                             the HANDSHAKE Transaction will be of the type indicated by
                             the Result Type.

                                 rtNotReady
                                 rtErrInvalidReportID
                                 rtErrUnsupportedRequest
                                 rtErrInvalidParameter
                                 rtErrUnknown
                                 rtErrFatal

                             The following Result Type will send a DATA Transaction in
                             response to the outstanding GET_PROTOCOL Transaction.

                                 rtData

Protocol   The Protocol Type to be sent in the response. The following
                                      Protocol Types are currently defined.

    ptReport
    ptBoot

This parameter is only used when the Result Type parameter is set to rtData.

Return:

Zero if successful.

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

        BTHID_ERROR_INVALID_PARAMETER
        BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
        BTHID_ERROR_NOT_INITIALIZED
        BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Set_Protocol_Request

This function is used to send a SET_PROTOCOL Transaction to a remote Bluetooth HID Device.

Notes:

1. Control Channel transfers have two phases, a Request by the host and a Response by the device. Only ONE host control channel Request shall be outstanding at a time. Reception of a HID Set Protocol Confirmation event indicates that a Response has been received and the Control Channel is now free for further Transactions.

Prototype:

int BTPSAPI **HID_Set_Protocol_Request**(unsigned int BluetoothStackID,
    unsigned int HIDID, HID_Protocol_Type_t Protocol)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be
                                      performed with. This is the value that was received via an
                                      Open Indication Event for Servers or from the return value of
                                      the Connection Functions for Clients.

       Protocol   The Protocol to use as the parameter to this SET_PROTOCOL Transaction. The following Protocol Types are currently defined.

                ptReport
                ptBoot

Return:

Zero if successful.

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

                BTHID_ERROR_INVALID_PARAMETER
                BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
                BTHID_ERROR_NOT_INITIALIZED
                BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_Indication
etHID_Set_Protocol_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.

HID_Set_Protocol_Response

This function is used to send a response to an outstanding SET_PROTOCOL Transaction to a remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Set_Protocol_Response**(unsigned int BluetoothStackID, unsigned int HIDID, HID_Result_Type_t ResultType)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

ResultType   The Result Type for this Response. The following Result Type is invalid for use with this function.

        rtData

The following Result Types will send a HANDSHAKE Transaction as the response to the outstanding

SET_PROTOCOL Transaction. The Result Code parameter of the HANDSHAKE Transaction will be of the type indicated by the Result Type.

> rtSuccessful
> rtNotReady
> rtErrInvalidReportID
> rtErrUnsupportedRequest
> rtErrInvalidParameter
> rtErrUnknown
> rtErrFatal

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Get_Idle_Request

This function is used to send a GET_IDLE Transaction to a remote Bluetooth HID Device.

Notes:

1. Control Channel transfers have two phases, a Request by the host and a Response by the device. Only ONE host control channel Request shall be outstanding at a time. Reception of a HID Get Idle Confirmation event indicates that a Response has been received and the Control Channel is now free for further Transactions.

Prototype:

int BTPSAPI **HID_Get_Idle_Request**(unsigned int BluetoothStackID, unsigned int HIDID)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be performed with. This is the value that was received via an

Open Indication Event for Servers or from the return value of
the Connection Functions for Clients.

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_Indication
etHID_Get_Idle_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.

HID_Get_Idle_Response

This function is used to send a response to an outstanding GET_IDLE Transaction to a
remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Get_Idle_Response**(unsigned int BluetoothStackID,
    unsigned int HIDID, HID_Result_Type_t ResultType, Byte_t IdleRate)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
ResultType	The Result Type for this Response. The following Result Type is invalid for use with this function.

> rtSuccessful

The following Result Types will send a HANDSHAKE
Transaction as the response to the outstanding GET_IDLE
Transaction. The Result Code parameter of the HANDSHAKE
Transaction will be of the type indicated by the Result Type.

> rtNotReady

rtErrInvalidReportID
rtErrUnsupportedRequest
rtErrInvalidParameter
rtErrUnknown
rtErrFatal

The following Result Type will send a DATA Transaction in response to the outstanding GET_IDLE Transaction.

rtData

IdleRate   The Idle Rate to be sent in the response. This parameter is only used when the Result Type parameter is set to rtData.

Return:

Zero if successful.

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

BTHID_ERROR_INVALID_PARAMETER
BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_ERROR_NOT_INITIALIZED
BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Set_Idle_Request

This function is used to send a SET_IDLE Transaction to a remote Bluetooth HID Device.

Notes:

1. Control Channel transfers have two phases, a Request by the host and a Response by the device. Only ONE host control channel Request shall be outstanding at a time. Reception of a HID Set Idle Confirmation event indicates that a Response has been received and the Control Channel is now free for further Transactions.

Prototype:

int BTPSAPI **HID_Set_Idle_Request**(unsigned int BluetoothStackID, unsigned int HIDID, Byte_t IdleRate)

Parameters:

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

HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
IdleRate	The Idle Rate to use as the parameter to this SET_IDLE Transaction. The Idle Rate LSB is weighted to 4ms (i.e. the Idle Rate resolution is 4ms with a range from 4ms to 1.020s).

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_Indication
etHID_Set_Idle_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.

HID_Set_Idle_Response

This function is used to send a response to an outstanding SET_IDLE Transaction to a remote Bluetooth HID Host.

Prototype:

int BTPSAPI **HID_Set_Idle_Response**(unsigned int BluetoothStackID, unsigned int HIDID, HID_Result_Type_t ResultType)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HIDID	The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
ResultType	The Result Type for this Response. The following Result Type is invalid for use with this function.
	rtData

The following Result Types will send a HANDSHAKE Transaction as the response to the outstanding SET_IDLE Transaction. The Result Code parameter of the HANDSHAKE Transaction will be of the type indicated by the Result Type.

> rtSuccessful
> rtNotReady
> rtErrInvalidReportID
> rtErrUnsupportedRequest
> rtErrInvalidParameter
> rtErrUnknown
> rtErrFatal

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION

Possible Events:

etHID_Close_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.

HID_Data_Write

This function is used to send Report Data over the Interrupt Channel to the remote Device.

Prototype:

int BTPSAPI **HID_Data_Write** (unsigned int BluetoothStackID, unsigned int HIDID, HID_Report_Type_Type_t ReportType, Word_t ReportPayloadSize, Byte_t *ReportDataPayload)

Parameters:

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

HIDID   The unique identifier of the connection this command is to be performed with. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

ReportType	The Report Type of the Report being sent as part of the DATA Transaction on the Interrupt Channel. The following Report Types are valid for this parameter in this function.

> rtInput
> rtOutput

Reports of Type rtInput are only valid for Bluetooth HID Devices sending reports to Bluetooth HID Hosts.

Reports of Type rtOutput are only valid for Bluetooth HID Hosts sending reports to Bluetooth HID Devices.

ReportPayloadSize	The Size of the Report to which the Report Data Payload parameter points.
ReportDataPayload	Pointer to the Report Data to be sent as part of the DATA Transaction on the Interrupt Channel.

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_OPERATION
> BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

Note that if the error code BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE is returned, then the etHID_Data_Buffer_Empty_Indication event will be dispatched to denote that Interrupt Channel L2CAP queue is empty and can accept more data. See the HID_Get_Data_Queueing_Parameters () and HID_Set_Data_Queueing_Parameters() for more information on the usage of this mechanism.

Possible Events:

etHID_Close_Indication

etHID_Data_Buffer_Empty_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.

HID_Get_Server_Connection_Mode

This function retrieves the current HID Server Connection Mode. The default Server Connection Mode is hsmAutomaticAccept. This function is used for HID Servers that use Bluetooth Security Mode 2.

Prototype:

int BTPSAPI **HID_Get_Server_Connection_Mode**(unsigned int BluetoothStackID,
    HID_Server_Connection_Mode_t *ServerConnectionMode)

Parameters:

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

ServerConnectionMode   Pointer to Server Connection Mode variable which will receive
                               the current Server Connection Mode. May be one of the
                               following:
                                    hsmAutomaticAccept
                                    hsmAutomaticReject
                                    hsmManualAccept

Return:

Zero if successful.

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

                               BTHID_ERROR_INVALID_PARAMETER
                               BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
                               BTHID_ERROR_NOT_INITIALIZED

Possible Events:

etHID_Close_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.

HID_Set_Server_Connection_Mode

This function changes the HID Server Connection Mode. The default Server Connection
Mode is hsmAutomaticAccept. This function is used for HID Servers that use Bluetooth
Security Mode 2.

Prototype:

int BTPSAPI **HID_Set_Server_Connection_Mode**(unsigned int BluetoothStackID,
    HID_Server_Connection_Mode_t ServerConnectionMode)

Parameters:

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

ServerConnectionMode   Value to set the Server Connection Mode to. May be one of the
                               following:
                                    hsmAutomaticAccept

> hsmAutomaticReject
> hsmManualAccept

Return:

Zero if successful.

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

> BTHID_ERROR_INVALID_PARAMETER
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_ERROR_NOT_INITIALIZED

Possible Events:

etHID_Close_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.

HID_Get_Data_Queueing_Parameters

Retrieves the current HID/L2CAP data queueing parameters. These parameters dictate how the data packets are queued into L2CAP (when calling the HID_Data_Write() function). This mechanism allows for the ability to implement a streaming type interface by limiting the number of packets that can queued (simultaneously) in L2CAP. This is useful to keep L2CAP from infinitely queing packets which can lead to either stale data or running out of memory.

Notes:

This function sets the queing parameters globally for HID. Setting the Queing parameters for an individual connections is currently not supported.

A value of zero for the QueueLimit member of the L2CAP queing parameters means that there is no queing active (i.e. all packets are queued, regardless of the queue depth).

Prototype:

int BTPSAPI **HID_Get_Data_Queueing_Parameters**(unsigned int BluetoothStackID, L2CA_Queueing_Parameters_t *QueueingParameters)

Parameters:

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

QueueingParameters   Pointer to a structure that will contain the currently configured Queing Parameters that are currently used by HID. See the L2CAP_Enhanced_Data_Write() function (in Bluetopia Core API documentation) for more information on the values for this parameter.

Return:

Zero if successful.

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

> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_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.

HID_Set_Data_Queueing_Parameters

Sets the current HID/L2CAP data queueing parameters. These parameters dictate how the data packets are queued into L2CAP (when calling the HID_Data_Write() function). This mechanism allows for the ability to implement a streaming type interface by limiting the number of packets that can queued (simultaneously) in L2CAP. This is useful to keep L2CAP from infinitely queing packets which can lead to either stale data or running out of memory.

Notes:

This function sets the queing parameters globally for HID. Setting the Queing parameters for an individual connection is currently not supported.

A value of zero for the QueueLimit member of the L2CAP queing parameters means that there is no queing active (i.e. all packets are queued, regardless of the queue depth).

Prototype:

int BTPSAPI **HID_Set_Data_Queueing_Parameters**(unsigned int BluetoothStackID, L2CA_Queueing_Parameters_t *QueueingParameters)

Parameters:

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

QueueingParameters   Pointer to a structure that contains the new Queing Parameters to set. See the L2CAP_Enhanced_Data_Write() function (in Bluetopia Core API documentation) for more information on the values for this parameter.

Return:

Zero if successful.

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

> BTHID_ERROR_NOT_INITIALIZED
> BTHID_ERROR_INVALID_BLUETOOTH_STACK_ID
> BTHID_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.

2.2   HID Profile Event Callback Prototypes

The event callback functions mentioned in the Human Interface Device Profile Registration or Connection commands all accept the callback function described by the following prototype.

HID_Event_Callback_t

Prototype of callback function passed in one of the HID Registration or Connection commands.

Prototype:

void (BTPSAPI ***HID_Event_Callback_t**)(unsigned int BluetoothStackID,
    HID_Event_Data_t *HID_Event_Data, unsigned long CallbackParameter)

Parameters:

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

HID_Event_Data   Data describing the event for which the callback function is
                             called. This is defined by the following structure:

```
typedef struct
{
  HID_Event_Type_t  Event_Data_Type;
  Word_t            Event_Data_Size;
  union
  {
    HID_Open_Request_Indication_Data_t    *HID_Open_Request_Indication_Data;
    HID_Open_Indication_Data_t            *HID_Open_Indication_Data;
    HID_Open_Confirmation_Data_t          *HID_Open_Confirmation_Data;
    HID_Close_Indication_Data_t           *HID_Close_Indication_Data;
    HID_Control_Indication_Data_t         *HID_Control_Indication_Data;
    HID_Get_Report_Indication_Data_t      *HID_Get_Report_Indication_Data;
    HID_Get_Report_Confirmation_Data_t    *HID_Get_Report_Confirmation_Data;
    HID_Set_Report_Indication_Data_t      *HID_Set_Report_Indication_Data;
    HID_Set_Report_Confirmation_Data_t    *HID_Set_Report_Confirmation_Data;
    HID_Get_Protocol_Indication_Data_t    *HID_Get_Protocol_Indication_Data;
    HID_Get_Protocol_Confirmation _Data_t *HID_Get_Protocol_Confirmation_Data;
    HID_Set_Protocol_Indication_Data_t    *HID_Set_Protocol_Indication_Data;
    HID_Set_Protocol_Confirmation_Data_t  *HID_Set_Protocol_Confirmation_Data;
    HID_Get_Idle_Indication_Data_t        *HID_Get_Idle_Indication_Data;
    HID_Get_Idle_Confirmation_Data_t      *HID_Get_Idle_Confirmation _Data;
    HID_Set_Idle_Indication_Data_t        *HID_Set_Idle_Indication_Data;
```

```
                    HID_Set_Idle_Confirmation_Data_t          *HID_Set_Idle_Confirmation_Data;
                    HID_Data_Indication_Data_t                *HID_Data_Indication_Data;
                    HID_Data_Buffer_Empty_Indication_Data_t
                          *HID_Data_Buffer_Empty_Indication_Data;
                 } Event_Data;
               } HID_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   HID Profile Events

The possible Human Interface Device Profile events from the Bluetooth stack are listed in the table below and are described in the text that follows:

Event	Description
etHID_Open_Indication	Dispatched when a remote client connects to a locally registered server.
etHID_Open_Confirmation	Dispatched to the local client to indicate success or failure of a previously submitted connection request.
etHID_Close_Indication	Dispatched when the remote entity disconnects from the local entity.
etHID_Control_Indication	Dispatched when the local entity receives a HID_CONTROL Transaction.
etHID_Get_Report_Indication	Dispatched when the local Bluetooth HID Device receives a GET_REPORT Transaction.
etHID_Get_Report_Confirmation	Dispatched when the local Bluetooth HID Host receives a response to an outstanding GET_REPORT Transaction.
etHID_Set_Report_Indication	Dispatched when the local Bluetooth HID Device receives a SET_REPORT Transaction.
etHID_Set_Report_Confirmation	Dispatched when the local Bluetooth HID Host receives a response to an outstanding SET_REPORT

	Transaction.
etHID_Get_Protocol_Indication	Dispatched when the local Bluetooth HID device receives a GET_PROTOCOL Transaction.
etHID_Get_Protocol_Confirmation	Dispatched when the local Bluetooth HID Host receives a response to an outstanding GET_PROTOCOL Transaction.
etHID_Set_Protocol_Indication	Dispatched when the local Bluetooth HID Device receives a SET_PROTOCOL Transaction.
etHID_Set_Protocol_Confirmation	Dispatched when the local Bluetooth HID Host receives a response to an outstanding SET_PROTOCOL Transaction.
etHID_Get_Idle_Indication	Dispatched when the local Bluetooth HID Device receives a GET_IDLE Transaction.
etHID_Get_Idle_Confirmation	Dispatched when the local Bluetooth HID Host receives a response to an outstanding GET_IDLE Transaction.
etHID_Set_Idle_Indication	Dispatched when the local Bluetooth HID Device receives a SET_IDLE Transaction.
etHID_Set_Idle_Confirmation	Dispatched when the local Bluetooth HID Host receives a response to an outstanding SET_IDLE Transaction.
etHID_Data_Indication	Dispatched when the local entity receives a DATA Transaction on the Interrupt Channel.
etHID_Data_Buffer_Empty_Indication	Dispatched by the local entity to the local application when a HID Data Channel no longer has any data queued to be sent.

etHID_Open_Indication

Dispatched when a remote client connects to a locally registered server.

Return Structure:

```
typedef struct
{
  unsigned int   HIDID;
  BD_ADDR_t BD_ADDR;
} HID_Open_Indication_Data_t;
```

Event Parameters:

HIDD   The unique identifier of this connection. This value will
                              represent the HIDID that can be passed to all other functions
                              that require it.

BD_ADDR	The Board Address of the remote client that has connected to the locally registered server.

etHID_Open_Confirmation

Dispatched to the local client to indicate success or failure of a previously submitted connection request.

Return Structure:

typedef struct
{
  unsigned int   HIDID;
  unsigned int   OpenStatus;
} **HID_Open_Confirmation_Data_t**;

Event Parameters:

HIDID	The unique identifier of the connection for which this event is intended. This is the value that was received from the return value of the Connection Function.
OpenStatus	The open status for the Connection Request. The following constants are currently defined as possible values.

> HID_OPEN_PORT_STATUS_SUCCESS
> HID_OPEN_PORT_STATUS_CONNECTION_TIMEOUT
> HID_OPEN_PORT_STATUS_CONNECTION_REFUSED
> HID_OPEN_PORT_STATUS_UNKNOWN_ERROR

etHID_Close_Indication

Dispatched when the remote entity disconnects from the local entity.

Return Structure:

typedef struct
{
  unsigned int   HIDID;
} **HID_Close_Indication_Data_t**;

Event Parameters:

HIDID	The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

etHID_Control_Indication

Dispatched when the local entity receives a HID_CONTROL Transaction.

Notes:

1. Bluetooth HID Hosts are only capable of receiving HID_CONTROL Transactions with a Control Operation value of hcVirtualCableUnplug.

Return Structure:

```
typedef struct
{
  unsigned int                          HIDID;
  HID_Control_Operation_Type_t          ControlOperation;
} HID_Control_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

ControlOperation   The Control Operation Type being requested by the remote entity. The following Control Operations are currently defined.

> hcNop
> hcHardReset
> hcSoftReset
> hcSuspend
> hcExitSuspend
> hcVirtualCableUnplug

etHID_Get_Report_Indication

Dispatched when the local Bluetooth HID Device receives a GET_REPORT Transaction.

Return Structure:

```
typedef struct
{
  unsigned int                   HIDID;
  HID_Get_Report_Size_Type_t     Size;
  HID_Report_Type_Type_t         ReportType;
  Byte_t                         ReportID;
  Word_t                         BufferSize;
} HID_Get_Report_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

Size   The Size specifies the type of allocation that the Bluetooth HID Host has used for the returned report. The following Get Report Size Types are currently defined.

> grSizeOfReport

grUseBufferSize

ReportType   The Report Type of the Report in which GET_REPORT
                        Transaction is requesting. The following Report Types are
                        valid as Report Type members for this event.

                            rtInput
                            rtOutput
                            rtFeature

ReportID   The Report ID of the Report in which the GET_REPORT
                        Transaction is requesting. If this member is set to
                        HID_INVALID_REPORT_ID this member is not used or invalid
                        with in this event.

BufferSize   The Buffer Size in which the host has allocated for the
                        response report buffer. This will be the maximum number of
                        bytes that should be received in the response phase of this
                        transaction. This member is only valid within this event if the
                        Size member within this event is set to grUseBufferSize.

etHID_Get_Report_Confirmation

Dispatched when the local Bluetooth HID Host receives a response to an outstanding
GET_REPORT Transaction.

Return Structure:

```
typedef struct
{
  unsigned int              HIDID;
  HID_Result_Type_t         Status;
  HID_Report_Type_Type_t    ReportType;
  Word_t                    ReportLength;
  Byte_t                    *ReportDataPayload;
} HID_Get_Report_Confirmation_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                        intended. This is the value that was received via an Open
                        Indication Event for Servers or from the return value of the
                        Connection Functions for Clients.

Status   The Result Type for this Response. The following Result Type
                        is invalid as a Status for this Event Type.

                            rtSuccessful

                        The following Result Types indicate the reception of a
                        HANDSHAKE Transaction as the response to the outstanding
                        GET_REPORT Transaction.

                            rtNotReady
                            rtErrInvalidReportID

                                         rtErrUnsupportedRequest
                                           rtErrInvalidParameter
                                           rtErrUnknown
                                           rtErrFatal

The following Result Type indicates the reception of a DATA Transaction in response to the outstanding GET_REPORT Transaction.

           rtData

ReportType	The Report Type of the Report being sent as the response to a GET_REPORT Transaction. This member is only used when the Status member is set to rtData. The following Report Types are valid for this member in this event.

           rtInput
           rtOutput
           rtFeature

ReportLength	The size of the Report in which the Report Data Payload member points. This member is only used when the Status member is set to rtData.
ReportDataPayload	Pointer to the Report Data being received. This member is only used when the Status member is set to rtData.

etHID_Set_Report_Indication

Dispatched when the local Bluetooth HID Device receives a SET_REPORT Transaction.

Return Structure:

```
typedef struct
{
  unsigned int             HIDID;
  HID_Report_Type_Type_t   ReportType;
  Word_t                   ReportLength;
  Byte_t                   *ReportDataPayload;
} HID_Set_Report_Indication_Data_t;
```

Event Parameters:

HIDID	The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
ReportType	The Report Type of the Report being received as part of the SET_REPORT Transaction. The following Report Types are valid for this member in this event.

           rtInput
           rtOutput
           rtFeature

ReportLength	The size of the Report in which the Report Data Payload member points.
ReportDataPayload	Pointer to the Report Data received as part of the SET_REPORT Transaction.

etHID_Set_Report_Confirmation

Dispatched when the local Bluetooth HID Host receives a response to an outstanding SET_REPORT Transaction.

Return Structure:

typedef struct
{
  unsigned int   HIDID;
  HID_Result_Type_t   Status;
} **HID_Set_Report_Confirmation_Data_t**;

Event Parameters:

HIDID	The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.
Status	The Result Type for this Response. The following Result Types is invalid as a Status for this Event.

        rtData

        The following Result Types indicate the reception of a HANDSHAKE Transaction as the response to the outstanding SET_REPORT Transaction.

          rtSuccessful
          rtNotReady
          rtErrInvalidReportID
          rtErrUnsupportedRequest
          rtErrInvalidParameter
          rtErrUnknown
          rtErrFatal

        Where rtSuccessful indicates that the SET_REPORT Transaction was successful.

etHID_Get_Protocol_Indication

Dispatched when the local Bluetooth HID device receives a GET_PROTOCOL Transaction.

Return Structure:

```
typedef struct
{
  unsigned int   HIDID;
} HID_Get_Protocol_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

etHID_Get_Protocol_Confirmation

Dispatched when the local Bluetooth HID Host receives a response to an outstanding GET_PROTOCOL Transaction.

Return Structure:

```
typedef struct
{
  unsigned int              HIDID;
  HID_Result_Type_t         Status;
  HID_Protocol_Type_t       Protocol;
} HID_Get_Protocol_Confirmation_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

Status   The Result Type for this Response. The following Result Type is invalid as a Status for this Event Type.

    rtSuccessful

The following Result Types indicate the reception of a HANDSHAKE Transaction as the response to the outstanding GET_PROTOCOL Transaction.

    rtNotReady
    rtErrInvalidReportID
    rtErrUnsupportedRequest
    rtErrInvalidParameter
    rtErrUnknown
    rtErrFatal

The following Result Type indicates the reception of a DATA Transaction in response to the outstanding GET_PROTOCOL Transaction.

    rtData

Protocol   The currently set protocol. The following Protocol Types are
                                      currently defined.

> ptReport
> ptBoot

                                      This member is only valid when the Status member is set to
                                      rtData.

etHID_Set_Protocol_Indication

Dispatched when the local Bluetooth HID Device receives a SET_PROTOCOL
Transaction.

Return Structure:

```
typedef struct
{
  unsigned int            HIDID;
  HID_Protocol_Type_t     Protocol;
} HID_Set_Protocol_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                                      intended. This is the value that was received via an Open
                                      Indication Event for Servers or from the return value of the
                                      Connection Functions for Clients.

Protocol   The Protocol Type that the Remote Bluetooth HID Host is
                                      requesting the Local Bluetooth HID Device change to. The
                                      following Protocol Types are currently defined.

> ptReport
> ptBoot

etHID_Set_Protocol_Confirmation

Dispatched when the local Bluetooth HID Host receives a response to an outstanding
SET_PROTOCOL Transaction.

Return Structure:

```
typedef struct
{
  unsigned int            HIDID;
  HID_Result_Type_t   Status;
} HID_Set_Protocol_Confirmation_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                                      intended. This is the value that was received via an Open
                                      Indication Event for Servers or from the return value of the
                                      Connection Functions for Clients.

Status   The Result Type for this Response. The following Result
                                    Types is invalid as a Status for this Event.

        rtData

The following Result Types indicate the reception of a
HANDSHAKE Transaction as the response to the outstanding
SET_PROTOCOL Transaction.

        rtSuccessful
        rtNotReady
        rtErrInvalidReportID
        rtErrUnsupportedRequest
        rtErrInvalidParameter
        rtErrUnknown
        rtErrFatal

Where rtSuccessful indicates that the SET_PROTOCOL
Transaction was successful.

etHID_Get_Idle_Indication

Dispatched when the local Bluetooth HID Device receives a GET_IDLE Transaction.

Return Structure:

```
typedef struct
{
  unsigned int   HIDID;
} HID_Get_Idle_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                                    intended. This is the value that was received via an Open
                                    Indication Event for Servers or from the return value of the
                                    Connection Functions for Clients.

etHID_Get_Idle_Confirmation

Dispatched when the local Bluetooth HID Host receives a response to an outstanding
GET_IDLE Transaction.

Return Structure:

```
typedef struct
{
  unsigned int         HIDID;
  HID_Result_Type_t    Status;
  Byte_t               IdleRate;
} HID_Get_Idle_Confirmation_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                        intended. This is the value that was received via an Open
                        Indication Event for Servers or from the return value of the
                        Connection Functions for Clients.

Status   The Result Type for this Response. The following Result Type
                        is invalid as a Status for this Event Type.

> rtSuccessful

The following Result Types indicate the reception of a
HANDSHAKE Transaction as the response to the outstanding
GET_IDLE Transaction.

> rtNotReady
> rtErrInvalidReportID
> rtErrUnsupportedRequest
> rtErrInvalidParameter
> rtErrUnknown
> rtErrFatal

The following Result Type indicates the reception of a DATA
Transaction in response to the outstanding GET_IDLE
Transaction.

> rtData

IdleRate   The current Idle Rate. This member is only valid when the
                        Status member is set to rtData.

etHID_Set_Idle_Indication

Dispatched when the local Bluetooth HID Device receives a SET_IDLE Transaction.

Return Structure:

```
typedef struct
{
  unsigned int   HIDID;
  Byte_t         IdleRate;
} HID_Set_Idle_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                               intended. This is the value that was received via an Open
                               Indication Event for Servers or from the return value of the
                               Connection Functions for Clients.

IdleRate   The Idle Rate that the Remote Bluetooth HID Host is
                               requesting the Local Bluetooth HID Device change to. The
                               Idle Rate LSB is weighted to 4ms (i.e. the Idle Rate resolution
                               is 4ms with a range from 4ms to 1.020s).

etHID_Set_Idle_Confirmation

Dispatched when the local Bluetooth HID Host receives a response to an outstanding
SET_IDLE Transaction.

Return Structure:

```
typedef struct
{
  unsigned int        HIDID;
  HID_Result_Type_t   Status;
} HID_Set_Idle_Confirmation_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                               intended. This is the value that was received via an Open
                               Indication Event for Servers or from the return value of the
                               Connection Functions for Clients.

Status   The Result Type for this Response. The following Result
                               Types is invalid as a Status for this Event.

                                   rtData

                               The following Result Types indicate the reception of a
                               HANDSHAKE Transaction as the response to the outstanding
                               SET_IDLE Transaction.

                                   rtSuccessful
                                   rtNotReady
                                   rtErrInvalidReportID
                                   rtErrUnsupportedRequest
                                   rtErrInvalidParameter
                                   rtErrUnknown

rtErrFatal

Where rtSuccessful indicates that the SET_IDLE Transaction was successful.

etHID_Data_Indication

Dispatched when the local entity receives a DATA Transaction on the Interrupt Channel.

Return Structure:

```
typedef struct
{
  unsigned int              HIDID;
  HID_Report_Type_Type_t    ReportType;
  Word_t                    ReportLength;
  Byte_t                   *ReportDataPayload;
} HID_Data_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is intended. This is the value that was received via an Open Indication Event for Servers or from the return value of the Connection Functions for Clients.

ReportType   The Report Type of the Report being received as part of the DATA Transaction on the Interrupt Channel. The following Report Types are valid for this member in this event.

        rtInput
        rtOutput

Reports of Type rtInput are only valid for Bluetooth HID Devices sending reports to Bluetooth HID Hosts.

Reports of Type rtOutput are only valid for Bluetooth HID Hosts sending reports to Bluetooth HID Devices

ReportLength   The size of the Report in which the Report Data Payload member points.

ReportDataPayload   Pointer to the Report Data received as part of the DATA Transaction on the Interrupt Channel.

etHID_Data_Buffer_Empty_Indication

Dispatched by the local entity to the local application when a HID Data Channel no longer has any data queued to be sent on the Data Channel.

Return Structure:

```
typedef struct
{
  unsigned int       HIDID;
} HID_Data_Buffer_Empty_Indication_Data_t;
```

Event Parameters:

HIDID   The unique identifier of the connection for which this event is
                                 intended. This is the value that was received via an Open
                                 Indication Event for Servers or from the return value of the
                                 Connection Functions for Clients.

3.   File Distributions

The header files that are distributed with the Bluetooth Human Interface Device Profile Library are listed in the table below.

File	Contents/Description
HIDAPI.h	Bluetooth Human Interface Device Profile API definitions
SS1BTHID.h	Bluetooth Human Interface Device Profile Include file