



# Human Interface Device Profile (HID)

## Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.3.0  
October 21, 2015



Bluetooth and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc., USA and licensed to Stonestreet One, LLC. Bluetopia®, Stonestreet One™, and the Stonestreet One logo are registered trademarks of Stonestreet One, LLC, Louisville, Kentucky, USA. All other trademarks are property of their respective owners.  
Copyright © 2000-2014 by Stonestreet One, LLC. All rights reserved.

**Revision Control**

Rev	Update	Date	Author
1.0	SSO Latest Release	Jan 10,2014	E. Wemes
1.1	Added rtErrTimeout to the ResultType Enum	Oct 21,2015	D. Horowitz

# Table of Contents

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>5</b>
1.1	Scope .....	5
1.2	Applicable Documents .....	6
1.3	Acronyms and Abbreviations .....	7
<b>2.</b>	<b>HID PROFILE PROGRAMMING INTERFACE.....</b>	<b>9</b>
2.1	<b>HID Profile Commands.....</b>	<b>9</b>
	HID_Register_Host_Server .....	10
	HID_Register_Device_Server .....	11
	HID_Open_Request_Response.....	13
	HID_Un_Register_Server.....	13
	HID_Register_Device_SDP_Record.....	14
	HID_Register_Device_SDP_Record_Version .....	16
	HID_Connect_Remote_Device .....	18
	HID_Connect_Remote_Host.....	20
	HID_Close_Connection.....	21
	HID_Control_Request .....	22
	HID_Get_Report_Request.....	23
	HID_Get_Report_Response .....	24
	HID_Set_Report_Request.....	26
	HID_Set_Report_Response .....	27
	HID_Get_Protocol_Request .....	28
	HID_Get_Protocol_Response.....	29
	HID_Set_Protocol_Request.....	30
	HID_Set_Protocol_Response .....	31
	HID_Get_Idle_Request.....	32
	HID_Get_Idle_Response .....	33
	HID_Set_Idle_Request .....	34
	HID_Set_Idle_Response.....	35
	HID_Data_Write.....	36
	HID_Get_Server_Connection_Mode.....	38
	HID_Set_Server_Connection_Mode .....	39
	HID_Get_Data_Queueing_Parameters.....	39
	HID_Set_Data_Queueing_Parameters.....	40
2.2	<b>HID Profile Event Callback Prototypes.....</b>	<b>41</b>
	HID_Event_Callback_t.....	41
2.3	<b>HID Profile Events.....</b>	<b>42</b>
	etHID_Open_Indication.....	44
	etHID_Open_Confirmation .....	44
	etHID_Close_Indication .....	45
	etHID_Control_Indication .....	45
	etHID_Get_Report_Indication.....	45
	etHID_Get_Report_Confirmation .....	46

---

etHID_Set_Report_Indication .....	48
etHID_Set_Report_Confirmation .....	48
etHID_Get_Protocol_Indication .....	49
etHID_Get_Protocol_Confirmation.....	49
etHID_Set_Protocol_Indication.....	50
etHID_Set_Protocol_Confirmation .....	51
etHID_Get_Idle_Indication .....	52
etHID_Get_Idle_Confirmation .....	52
etHID_Set_Idle_Indication .....	53
etHID_Set_Idle_Confirmation.....	53
etHID_Data_Indication.....	54
etHID_Data_Buffer_Empty_Indication.....	55
<b>3. FILE DISTRIBUTIONS.....</b>	<b>56</b>

---

# 1. Introduction

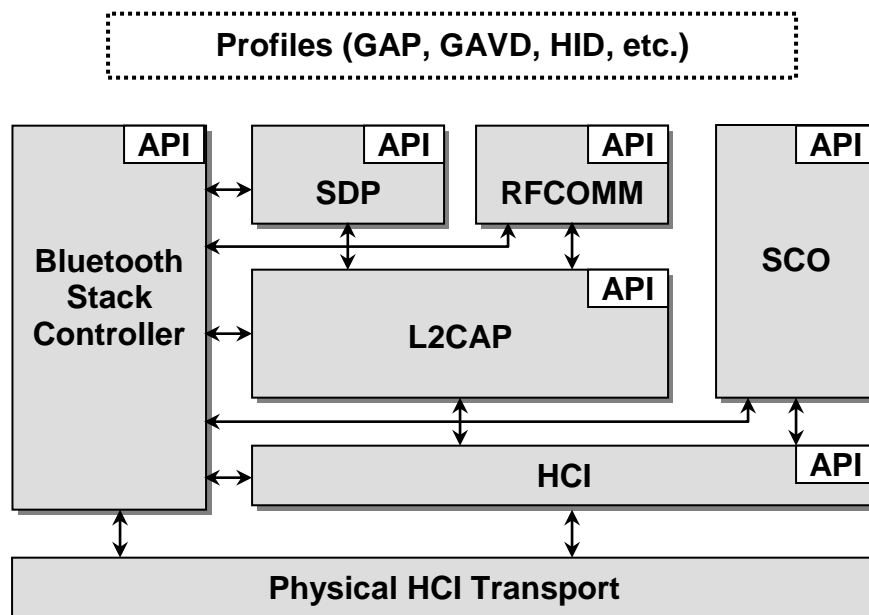
Bluetopia®, the Bluetooth Protocol Stack by Stonestreet One provides a software architecture that encapsulates the upper functionality of the Bluetooth Protocol Stack. More specifically, this stack is a software solution that resides above the Physical HCI (Host Controller Interface) Transport Layer 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
- Windows Mobile
- Windows CE
- Linux
- QNX
- Other Embedded OS



**Figure 1-1 The Stonestreet One Bluetooth Protocol Stack**

## 1.2 Applicable Documents

The following documents may be used for additional background and technical depth regarding the Bluetooth technology.

1. *Specification of the Bluetooth System, Volume 1, 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 Data 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 <sup>1</sup>	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 <sup>1</sup>	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HIDConfiguration	<p>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.</p> <pre>typedef struct {     HID_Flow_Config_t OutFlow;     HID_Flow_Config_t InFlow;     Word_t InMTU; } <b>HID_Configuration_t</b>;</pre> <p>where flow members are used for negotiating the Interrupt Channel and the MTU member is used for negotiating both Control and Interrupt Channels.</p> <p>The OutFlow member defines the minimum and maximum flow configuration in which the entity can operate properly.</p> <p>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.</p> <p>The InMTU member specifies the Maximum Transmission Unit that will be negotiated for incoming data.</p>
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 <sup>1</sup>	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<sup>1</sup>                      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 <sup>1</sup>	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 <sup>1</sup>	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
DeviceFlags	<p>Bit Field used to define miscellaneous HID Profile SDP Attributes. The following bit constants are currently defined for use with this parameter.</p> <pre> 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 </pre>
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	<p>The List of Descriptors to be used with the HID Descriptor List Attribute. An individual descriptor is defined by the following structure.</p> <pre> typedef struct {     Byte_t    DescriptorType;     Word_t    DescriptorLength;     Byte_t    *Descriptor; } <b>HID_Descriptor_t</b>; </pre> <p>where DescriptorType specifies the Class Descriptor Type as defined in the Bluetooth Human Interface Device (HID) Profile.</p> <p>The DataLength member specifies the Length of the Descriptor member that follows.</p> <p>The Descriptor member specifies the actual Class Descriptor Data to use.</p>

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: <pre>typedef struct {     unsigned long          Flags;     HID_Sniff_Subrating_Parameters_t                                 SniffSubratingParameters;     Word_t                  LinkSupervisionTimeout; } <b>HID_SDP_Record_Information_t</b>;</pre> <p>The Flags member is a bitmask variable which denotes which members of the structure are valid. The valid bits are as follow:</p> <pre>HID_SDP_RECORD_INFORMATION_FLAGS_SNIFF_SUBRATING_VALID HID_SDP_RECORD_INFORMATION_FLAGS_SUPERVISION_TIMEOUT_VALID</pre> <p>Based on the bits that are set the appropriate member of the containing structure must contain valid data.</p>
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 <sup>1</sup>	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	<p>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.</p> <pre>typedef struct {     HID_Flow_Config_t OutFlow;     HID_Flow_Config_t InFlow;     Word_t InMTU; } <b>HID_Configuration_t</b>;</pre> <p>where flow members are used for negotiating the Interrupt Channel and the MTU member is used for negotiating both Control and Interrupt Channels.</p> <p>The OutFlow member defines the minimum and maximum flow configuration in which the entity can operate properly.</p> <p>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.</p> <p>The InMTU member specifies the Maximum Transmission Unit that will be negotiated for incoming data.</p>
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<sup>1</sup> 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 <sup>1</sup>	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 <sup>1</sup>	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. <div style="margin-left: 40px;">                     hcNop                      hcHardReset                      hcSoftReset                      hcSuspend                      hcExitSuspend                      hcVirtualCableUnplug                 </div>

**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 <sup>1</sup>	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 <sup>1</sup>	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	<p>The Result Type for this Response. The following Result Type is invalid for use with this function.</p> <p>rtSuccessful</p> <p>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.</p> <p>rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal rtErrTimeout</p> <p>The following Result Type will send a DATA Transaction in response to the outstanding GET_REPORT Transaction.</p> <p>rtData</p>
ReportType	<p>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.</p> <p>rtInput rtOutput rtFeature</p>
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 <sup>1</sup>	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  
 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 <sup>1</sup>	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

rtErrTimeout

**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 <sup>1</sup>	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 <sup>1</sup>	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	<p>The Result Type for this Response. The following Result Type is invalid for use with this function.</p> <p>rtSuccessful</p> <p>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.</p> <p>rtNotReady            rtErrInvalidReportID            rtErrUnsupportedRequest            rtErrInvalidParameter            rtErrUnknown            rtErrFatal            rtErrTimeout</p>

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<sup>1</sup>

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 <sup>1</sup>	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  
rtErrTimeout

**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 <sup>1</sup>	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 <sup>1</sup>	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  
rtErrTimeout

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 <sup>1</sup>	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 <sup>1</sup>	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 rtErrTimeout

**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 <sup>1</sup>	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 <sup>1</sup>	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 <sup>1</sup>	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 be queued (simultaneously) in L2CAP. This is useful to keep L2CAP from infinitely queuing packets which can lead to either stale data or running out of memory.

### Notes:

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

A value of zero for the QueueLimit member of the L2CAP queuing parameters means that there is no queuing 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 <sup>1</sup>	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 Queuing 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 be queued (simultaneously) in L2CAP. This is useful to keep L2CAP from infinitely queuing packets which can lead to either stale data or running out of memory.

Notes:

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

A value of zero for the QueueLimit member of the L2CAP queuing parameters means that there is no queuing 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 <sup>1</sup>	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 <sup>1</sup>	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:

HIDID	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. <div style="margin-left: 40px;"> HID_OPEN_PORT_STATUS_SUCCESS  HID_OPEN_PORT_STATUS_CONNECTION_TIMEOUT  HID_OPEN_PORT_STATUS_CONNECTION_REFUSED  HID_OPEN_PORT_STATUS_UNKNOWN_ERROR </div>

**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	<p>The Control Operation Type being requested by the remote entity. The following Control Operations are currently defined.</p> <ul style="list-style-type: none"> <li>hcNop</li> <li>hcHardReset</li> <li>hcSoftReset</li> <li>hcSuspend</li> <li>hcExitSuspend</li> <li>hcVirtualCableUnplug</li> </ul>

**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	<p>The Result Type for this Response. The following Result Type is invalid as a Status for this Event Type.</p> <p>rtSuccessful</p> <p>The following Result Types indicate the reception of a HANDSHAKE Transaction as the response to the outstanding GET_REPORT Transaction.</p> <p>rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal rtErrTimeout</p> <p>The following Result Type indicates the reception of a DATA Transaction in response to the outstanding GET_REPORT Transaction.</p> <p>rtData</p>
ReportType	<p>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.</p> <p>rtInput rtOutput rtFeature</p>
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:**

<b>HIDID</b>	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.
<b>ReportType</b>	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. <div> <div>rtInput</div> <div>rtOutput</div> <div>rtFeature</div> </div>
<b>ReportLength</b>	The size of the Report in which the Report Data Payload member points.
<b>ReportDataPayload</b>	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:**

<b>HIDID</b>	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
	rtErrTimeout
	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	<p>The Result Type for this Response. The following Result Type is invalid as a Status for this Event Type.</p> <p>rtSuccessful</p> <p>The following Result Types indicate the reception of a HANDSHAKE Transaction as the response to the outstanding GET_PROTOCOL Transaction.</p> <p>rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal rtErrTimeout</p> <p>The following Result Type indicates the reception of a DATA Transaction in response to the outstanding GET_PROTOCOL Transaction.</p> <p>rtData</p>
Protocol	<p>The currently set protocol. The following Protocol Types are currently defined.</p> <p>ptReport ptBoot</p> <p>This member is only valid when the Status member is set to rtData.</p>

**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.  <div style="margin-left: 40px;"> ptReport  ptBoot </div>

**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.  <div style="margin-left: 40px;">rtData</div> <p>The following Result Types indicate the reception of a HANDSHAKE Transaction as the response to the outstanding SET_PROTOCOL Transaction.</p> <div style="margin-left: 40px;"> rtSuccessful  rtNotReady  rtErrInvalidReportID  rtErrUnsupportedRequest  rtErrInvalidParameter </div>

rtErrUnknown  
rtErrFatal  
rtErrTimeout

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  
 rtErrTimeout

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
rtErrTimeout
```

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