HID Host Profile Sub-system (HIDH)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
March 5, 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 HID Host profile sub-system provided by Bluetopia. Chapter 2 contains a description of the programming interface for this profile sub-system. And, Chapter 3 contains the header file name list for the Bluetooth HID Host profile sub-system library.

1.1 Scope

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

- Windows 95
- Windows 98
- Linux
- Windows NT 4.0
- Windows 2000
- QNX
- Windows Millennium
- Windows CE

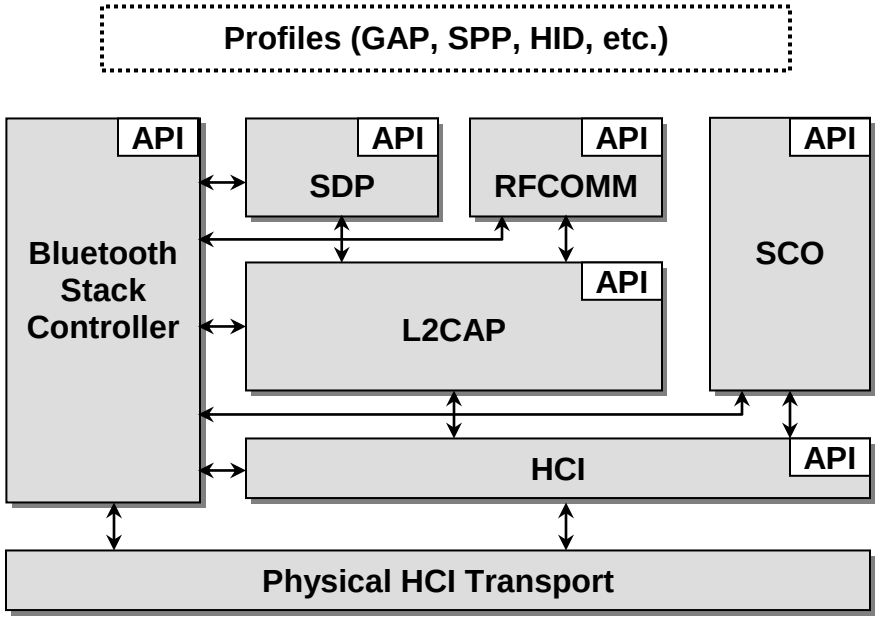

Figure 1-1 the Stonestreet One Bluetooth Protocol Stack

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 2, Core System Package [Controller volume]*, version 2.0 +EDR, November 2004.
2. *Specification of the Bluetooth System, Volume 3, Core System Package [Host volume]*, version 2.0 +EDR, November 2004.
3. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 2.1+EDR, July 26, 2007.
4. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 2.1+EDR, July 26, 2007.
5. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 2.1+EDR, July 26, 2007.
6. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 2.1+EDR, July 26, 2007.
7. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 2.1+EDR, July 26, 2007.
8. *Specification of the Bluetooth System, Bluetooth Core Specification Addendum 1*, June 26, 2008.
9. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 3.0+HS, April 21, 2009.
10. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 3.0+HS, April 21, 2009.
11. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 3.0+HS, April 21, 2009.
12. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 3.0+HS, April 21, 2009.
13. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 3.0+HS, April 21, 2009.
14. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 3.0+HS, April 21, 2009.
15. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 4.0, June 30, 2010.
16. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 4.0, June 30, 2010.
17. *Specification of the Bluetooth System, Volume 2, Core System Package [BR/EDR Controller Volume]*, version 4.0, June 30, 2010.

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

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

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

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

22. Adopted Bluetooth Profiles, Protocol and Transport specifications, various versions and dates, available from Bluetooth SIG.

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

24. *Digital cellular telecommunications system (Phase 2+); Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol (GSM 07.10),* version 7.1.0, Release 1998; commonly referred to as: ETSI TS 07.10.

25. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual,* version 4.0.1, April 5, 2012.

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
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Data Rate
GAP	Generic Access Profile
HID	Human Interface Device
HS	High Speed
LE	Low Energy
LSB	Least Significant Bit
MSB	Most Significant Bit

Term	Meaning
SDP	Service Discovery Protocol
SPP	Serial Port Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. Programming Interface

The Human Interface Device profile sub-system (HIDH) programming interface defines the protocols and procedures to be used to implement HID Host capabilities on the target platform. The profile sub-system commands are listed in section 2.1, the event callback prototype is described in section 2.2, and the profile sub-system events are itemized in section 2.3. The actual prototypes and constants outlined in this section can be found in the **HIDHAPI.H** header file in the Bluetopia distribution.

2.1 HID Host Profile Sub-system Commands

The available HID Host profile sub-system command functions are listed in the table below and are described in the text which follows.

Function	Description
HID_Host_Initialize	This function is responsible for initializing the HID Host profile sub-system.
HID_Host_Un_Initialize	This function is responsible for un-initializing the HID Host profile sub-system.
HID_Host_Open_Request_Response	This function is responsible for responding to an individual request to connect to the local HID Host profile sub-system server.
HID_Host_Connect_Remote_Device	This function is responsible for attempting an outgoing connection to a remote HID device.
HID_Host_Close_Connection	This function is responsible for disconnecting a specified remote HID device from the local HID Host profile sub-system server.
HID_Host_Get_Server_Connection_Mode	This function is responsible for retrieving the current HID Host profile sub-system server connection mode.
HID_Host_Set_Server_Connection_Mode	This function is responsible for setting the HID Host profile sub-system server connection mode.

HID_Host_Initialize

This function is responsible for initializing the HID Host profile sub-system. This entails registering any input handlers with the system that are required to add the HID events to the host input stream. Currently the implementation supports HID Mouse and HID Keyboard.

Prototype:

int BTPSAPI **HID_Host_Initialize**(unsigned int BluetoothStackID, HID_Configuration_t *HIDConfiguration, HID_Host_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 HID Configuration Specification to be used in the negotiation of the L2CAP channels associated with the HID Host client and server connections.
EventCallback	specifies the HID Host Event Callback function.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each packet.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION
BTHID_HOST_ERROR_INSUFFICIENT_RESOURCES

Possible Events:

etHID_Host_Open_Request_Indication

etHID_Host_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_Host_Un_Initialize

This function is responsible for un-initializing the local HID Host profile sub-system. This function closes all connections and cleans up all resources associated with the profile sub-system.

Prototype:

int BTPSAPI **HID_Host_Un_Initialize**(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_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_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_Host_Open_Request_Response

This function is responsible for responding to an individual connection request from a remote device to connect to the local HID Host profile sub-system (either HID mouse our HID keyboard). This function should be called in response to the receipt of an etHID_Host_Open_Request_Indication event.

Prototype:

int BTPSAPI **HID_Host_Open_Request_Response**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, Boolean_t AcceptConnection, DWord_t ConnectionFlags)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	the Bluetooth Device Address of the device that is connecting.
AcceptConnection	specifies whether to accept the pending connection request.
ConnectionFlags	Bit field used to define the Report handling behavior for the connection. The following bit constants are currently defined for use with this parameter: HID_HOST_CONNECTION_FLAGS_REPORT_MODE HID_HOST_CONNECTION_FLAGS_PARSE_BOOT If the REPORT_MODE flag is set, the initial Protocol for the connection will be ptReport. Otherwise, the initial Protocol will be ptBoot. If the PARSE_BOOT flag is set, then Boot-Mode Reports will be automatically parsed by the HID Host sub-system and will be indicated by either a Boot Keyboard Data Indication event

or Boot Mouse Data Indication event, and all other received Reports will be indicated by a Data Indication event. If the PARSE_BOOT flag is NOT set, all Reports received from the remote HID Device will be indicated by a Data Indication event.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Connect_Remote_Device

The following function is responsible for opening a connection to a Remote HID device on the specified Bluetooth device. This function will connection HID mouse and HID keyboard (if both are available). If only a single function is available (mouse or keyboard) then this function will connect only the supported HID functionality on the remote device.

Prototype:

int BTPSAPI **HID_Host_Connect_Remote_Device**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, DWord_t ConnectionFlags)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	specifies the Board Address (NON NULL) of the remote Bluetooth device to connect with.
ConnectionFlags	Bit field used to define the Report handling behavior for the connection. The following bit constants are currently defined for use with this parameter: HID_HOST_CONNECTION_FLAGS_REPORT_MODE HID_HOST_CONNECTION_FLAGS_PARSE_BOOT

If the REPORT_MODE flag is set, the initial Protocol for the connection will be ptReport. Otherwise, the initial Protocol will be ptBoot.

If the PARSE_BOOT flag is set, then Boot-Mode Reports will be automatically parsed by the HID Host sub-system and will be indicated by either a Boot Keyboard Data Indication event or Boot Mouse Data Indication event, and all other received Reports will be indicated by a Data Indication event. If the PARSE_BOOT flag is NOT set, all Reports received from the remote HID Device will be indicated by a Data Indication event.

Possible Events:

etHID_Host_Open_Confirmation

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION
BTHID_HOST_ERROR_ALREADY_CONNECTED
BTHID_HOST_ERROR_INSUFFICIENT_RESOURCES

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_Host_Close_Connection

The following function is responsible for closing an active HID Host profile sub-system connection (established through a connection made to the local Server or a connection that was made by calling the **HID_Host_Open_Remote_Device**() function).

Prototype:

int BTPSAPI **HID_Host_Close_Connection**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR, Boolean_t SendVirtualCableDisconnect)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	the Bluetooth device address of the device to disconnect.

SendVirtualCableDisconnect	If this parameter is set to TRUE, then a Virtual Cable Disconnect command will be sent to the remote HID Device when the connection is closed. This will prevent the remote device from automatically attempting to re-connect, for example, when the user presses a key.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_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_Host_Data_Write

The following function is responsible for sending HID reports over the HID Interrupt Channel to the remote Device.

Prototype:

int BTPSAPI **HID_Host_Data_Write** (unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR, Word_t ReportPayloadSize, Byte_t *ReportDataPayload)

Parameters:

BluetoothStackID[1]	3/5/2014Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	The Bluetooth device address of the connected device to which the Payload should be transmitted.
ReportPayloadSize	The amount of data, specified in bytes, to which the Report Data Payload parameter points.
ReportDataPayload	Pointer to the Report Data to be sent over the Interrupt Channel.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_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_Host_Get_Report_Request

The following function is responsible for sending a GET_REPORT Transaction to a remote Bluetooth HID Device over the HID Control Channel.

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 Host 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_Host_Get_Report_Request**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, 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.
BD_ADDR	The Bluetooth device address of the connected device to which the request should be sent.
Size	Description that indicates how the Device is to determine the size of the response report that the Host can receive. The following Get Report Size Types are currently defined: grSizeOfReport grUseBufferSize
ReportType	The Report Type of the Report 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 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 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_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Set_Report_Request

The following function is responsible for sending a SET_REPORT Transaction to a remote Bluetooth HID Device over the HID Control Channel.

Notes:

2. 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 Host 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_Host_Set_Report_Request**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, 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.
BD_ADDR	The Bluetooth device address of the connected device to which the request should be sent.

ReportType	The Report Type of the Report which this GET_REPORT Transaction is requesting. 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_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Get_Protocol_Request

The following function is responsible for sending a GET_PROTOCOL Transaction to a remote Bluetooth HID Device over the HID Control Channel.

Notes:

3. 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 Host 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_Host_Get_Protocol_Request**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR)

Parameters:

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

BD_ADDR	The Bluetooth device address of the connected device to which the request should be sent.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Set_Protocol_Request

The following function is responsible for sending a SET_PROTOCOL Transaction to a remote Bluetooth HID Device over the HID Control Channel.

Notes:

4. 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 Host 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_Host_Set_Protocol_Request**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, HID_Protocol_Type_t Protocol)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	The Bluetooth device address of the connected device to which the request should be sent.
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_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Get_Idle_Request

The following function is responsible for sending a GET_IDLE Transaction to a remote Bluetooth HID Device over the HID Control Channel.

Notes:

5. 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 Host 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_Host_Get_Idle_Request**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	The Bluetooth device address of the connected device to which the request should be sent.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Set_Idle_Request

The following function is responsible for sending a SET_IDLE Transaction to a remote Bluetooth HID Device over the HID Control Channel.

Notes:

6. 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 Host 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_Host_Set_Idle_Request**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, Byte_t IdleRate)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	The Bluetooth device address of the connected device to which the request should be sent.
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_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_ERROR_INVALID_OPERATION

Possible Events:

etHID_Host_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_Host_Set_Keyboard_Repeat

The following function is responsible for setting the internal Keyboard Repeat behavior. If both the Keyboard Repeat is enabled and the PARSE_BOOT flag is set for a connection, Bluetopia will generate Keyboard Repeat Indication events after issuing the Boot Keyboard Data Indication event for the button Press event and until issuing the Boot Keyboard Data Indication event for the button Release event. This feature simulates the auto-repeate behavior commonly seen when holding down a key on a traditional keyboard.

Prototype:

int BTPSAPI **HID_Host_Set_Keyboard_Repeat** (unsigned int BluetoothStackID, unsigned int RepeatDelay, unsigned int RepeatRate)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
RepeatDelay	The amount of delay, in milliseconds, before issuing the initial Keyboard Repeat Indication event. Setting this parameter to zero will disable the internal Keyboard Repeat behavior.
RepeatRate	The rate, specified in milliseconds, at which Keyboard Repeat Indication event will be repeated after the initial Repeat Delay.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED

Possible Events:

etHID_Host_Boot_Keyboard_Repeat_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_Host_Get_Server_Connection_Mode

The following function is responsible for retrieving the current HID Host profile sub-system server connection mode.

Prototype:

int BTPSAPI **HID_Host_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	a pointer to a Server Connection Mode variable which will receive the current Server Connection Mode.

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_ERROR_NOT_INITIALIZED
BTHID_HOST_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_Host_Set_Server_Connection_Mode

This function is responsible for setting the current HID Host profile sub-system server connection mode.

Prototype:

int BTPSAPI **HID_Host_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	The new server connection mode to set the server to use. Valid values are as follows: hsmAutomaticAccept hsmAutomaticReject hsmManualAccept

Return:

Zero if successful.

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

BTHID_HOST_ERROR_INVALID_PARAMETER
BTHID_HOST_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHID_HOST_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.

2.2 HID Host Profile Sub-system Event Callback Prototypes

The event callback functions, mentioned in the HID Host initialization function accepts a callback function described by the following prototype.

HID_Host_Event_Callback_t

Prototype of callback function passed to the **HID_Host_Initialize**() function.

Prototype:

void (BTPSAPI ***HID_Host_Event_Callback_t**)(unsigned int BluetoothStackID, HID_Host_Event_Data_t *HID_Host_Event_Data, unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize
HID_Host_Event_Data	Data describing the event for which the callback function is called. This is defined by the following structure:

```
typedef struct
{
  HID_Host_Event_Type_t Event_Data_Type;
  Word_t                Event_Data_Size;
  union
  {
    HID_Host_Open_Request_Indication_Data_t     *HID_Host_Open_Request_Indication_Data;
    HID_Host_Open_Indication_Data_t             *HID_Host_Open_Indication_Data;
    HID_Host_Open_Confirmation_Data_t           *HID_Host_Open_Confirmation_Data;
    HID_Host_Close_Indication_Data_t            *HID_Host_Close_Indication_Data;
    HID_Host_Boot_Keyboard_Data_t               *HID_Host_Boot_Keyboard_Data;
    HID_Host_Data_Indication_Data_t             *HID_Host_Data_Indication_Data;
    HID_Host_Boot_Keyboard_Repeat_Data_t        *HID_Host_Boot_Keyboard_Repeat_Data;
    HID_Host_Boot_Mouse_Data_t                  *HID_Host_Boot_Mouse_Data;
    HID_Host_Get_Report_Confirmation_Data_t     *HID_Host_Get_Report_Confirmation_Data;
    HID_Host_Set_Report_Confirmation_Data_t     *HID_Host_Set_Report_Confirmation_Data;
    HID_Host_Get_Protocol_Confirmation_Data_t   *HID_Host_Get_Protocol_Confirmation_Data;
    HID_Host_Set_Protocol_Confirmation_Data_t   *HID_Host_Set_Protocol_Confirmation_Data;
    HID_Host_Get_Idle_Confirmation_Data_t       *HID_Host_Get_Idle_Confirmation_Data;
    HID_Host_Set_Idle_Confirmation_Data_t       *HID_Host_Set_Idle_Confirmation_Data;
  } Event_Data;
} HID_Host_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 Events

The possible HIDH Profile events from the Bluetooth stack is listed in the table below and are described in the text that follows:

Event	Description
etHID_Host_Open_Request_Indication	Event that signals a remote HID device is attempting a connection
etHID_Host_Open_Indication	Event that signals that the local HID Host profile sub-system now has an active connection
etHID_Host_Open_Confirmation	Event that signals the result of a local HID Host profile sub-system connection to a

	remote device
etHID_Host_Close_Indication	Event that signals that a currently active HID Host profile sub-system connection is no longer active
etHID_Host_Boot_Keyboard_Data_Indication	Event that signals that a Boot Mode Keyboard Report has been received and parsed by the local HID Host.
etHID_Host_Boot_Keyboard_Repeat_Indication	Event that signals that a repeat keypress has been generated by the HID Host subsystem.
etHID_Host_Boot_Mouse_Data_Indication	Event that signals that a Boot Mode Mouse Report has been received and parsed by the local HID Host.
etHID_Host_Data_Indication	Event that indicates that a DATA Transaction has been receieved on the Interrupt Channel by the local HID Host.
etHID_Host_Get_Report_Confirmation	Event that indicates that the local HID Host received a response to an outstanding GET_REPORT Transaction.
etHID_Host_Set_Report_Confirmation	Event that indicates that the local HID Host received a response to an outstanding SET_REPORT Transaction.
etHID_Host_Get_Protocol_Confirmation	Event that indicates that the local HID Host received a response to an outstanding GET_PROTOCOL Transaction.
etHID_Host_Set_Protocol_Confirmation	Event that indicates that the local HID Host received a response to an outstanding SET_PROTOCOL Transaction.
etHID_Host_Get_Idle_Confirmation	Event that indicates that the local HID Host received a response to an outstanding GET_IDLE Transaction.
etHID_Host_Set_Idle_Confirmation	Event that indicates that the local HID Host received a response to an outstanding SET_IDLE Transaction.

etHID_Host_Open_Request_Indication

This event is dispatched when a remote HID device is requesting a connection to the local HID Host profile sub-system server.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
} HID_Host_Open_Request_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the Bluetooth device making the connection.

etHID_Host_Open_Indication

This event is dispatched when a remote HID device connects to the local HID Host profile sub-system server.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
} HID_Host_Open_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the Bluetooth device that is now connected.

etHID_Host_Open_Confirmation

This event is dispatched to indicate success or failure of a previously submitted connection request to a remote device.

Return Structure:

```
typedef struct
{
  BD_ADDR_t      BD_ADDR;
  unsigned int   OpenStatus;
} HID_Host_Open_Confirmation_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote device that was connected.
OpenStatus	specifies the connection status of the connection attempt. This will be one of the following 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_Host_Close_Indication

This event is dispatched when a remote device disconnects from the local device.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
} HID_Host_Close_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote device that disconnected.

etHID_Host_Boot_Keyboard_Data_Indication

This event is dispatched when the local HID Host device receives a Boot Mode Keyboard Report which is parsed by the HID Host subsystem.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
  Boolean_t KeyDown;
  Byte_t    KeyModifiers;
  Byte_t    Key;
} HID_Host_Boot_Keyboard_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
KeyDown	TRUE if the state of the key is Pressed.
KeyModifiers	Bit mask indicating what modifier keys are currently pressed. The following constants are currently defined for this member: HID_HOST_MODIFIER_FLAG_LEFT_CTRL HID_HOST_MODIFIER_FLAG_LEFT_SHIFT HID_HOST_MODIFIER_FLAG_LEFT_ALT HID_HOST_MODIFIER_FLAG_LEFT_GUI HID_HOST_MODIFIER_FLAG_RIGHT_CTRL HID_HOST_MODIFIER_FLAG_RIGHT_SHIFT HID_HOST_MODIFIER_FLAG_RIGHT_ALT HID_HOST_MODIFIER_FLAG_RIGHT_GUI
Key	The key that was pressed or released. The following constants are currently defined:

HID_HOST_RESERVED	HID_HOST_KEYBOARD_I
HID_HOST_KEYBOARD_ERROR_ROLL_OVER	HID_HOST_KEYBOARD_J
HID_HOST_KEYBOARD_POST_FAIL	HID_HOST_KEYBOARD_K
HID_HOST_KEYBOARD_ERROR_UNDEFINED	HID_HOST_KEYBOARD_L
HID_HOST_KEYBOARD_A	HID_HOST_KEYBOARD_M
HID_HOST_KEYBOARD_B	HID_HOST_KEYBOARD_N
HID_HOST_KEYBOARD_C	HID_HOST_KEYBOARD_O
HID_HOST_KEYBOARD_D	HID_HOST_KEYBOARD_P
HID_HOST_KEYBOARD_E	HID_HOST_KEYBOARD_Q
HID_HOST_KEYBOARD_F	HID_HOST_KEYBOARD_R
HID_HOST_KEYBOARD_G	HID_HOST_KEYBOARD_S
HID_HOST_KEYBOARD_H	HID_HOST_KEYBOARD_T

HID_HOST_KEYBOARD_U	HID_HOST_KEYBOARD_F10
HID_HOST_KEYBOARD_V	HID_HOST_KEYBOARD_F11
HID_HOST_KEYBOARD_W	HID_HOST_KEYBOARD_F12
HID_HOST_KEYBOARD_X	HID_HOST_KEYBOARD_PRINT_SCREEN
HID_HOST_KEYBOARD_Y	HID_HOST_KEYBOARD_SCROLL_LOCK
HID_HOST_KEYBOARD_Z	HID_HOST_KEYBOARD_PAUSE
HID_HOST_KEYBOARD_1	HID_HOST_KEYBOARD_INSERT
HID_HOST_KEYBOARD_2	HID_HOST_KEYBOARD_HOME
HID_HOST_KEYBOARD_3	HID_HOST_KEYBOARD_PAGE_UP
HID_HOST_KEYBOARD_4	HID_HOST_KEYBOARD_DELETE_FORWARD
HID_HOST_KEYBOARD_5	HID_HOST_KEYBOARD_END
HID_HOST_KEYBOARD_6	HID_HOST_KEYBOARD_PAGE_DOWN
HID_HOST_KEYBOARD_7	HID_HOST_KEYBOARD_RIGHT_ARROW
HID_HOST_KEYBOARD_8	HID_HOST_KEYBOARD_LEFT_ARROW
HID_HOST_KEYBOARD_9	HID_HOST_KEYBOARD_DOWN_ARROW
HID_HOST_KEYBOARD_0	HID_HOST_KEYBOARD_UP_ARROW
HID_HOST_KEYBOARD_RETURN	HID_HOST_KEYPAD_NUM_LOCK
HID_HOST_KEYBOARD_ESCAPE	HID_HOST_KEYPAD_SLASH
HID_HOST_KEYBOARD_DELETE	HID_HOST_KEYPAD_ASTERISK
HID_HOST_KEYBOARD_TAB	HID_HOST_KEYPAD_MINUS
HID_HOST_KEYBOARD_SPACE_BAR	HID_HOST_KEYPAD_PLUS
HID_HOST_KEYBOARD_MINUS	HID_HOST_KEYPAD_ENTER
HID_HOST_KEYBOARD_EQUAL	HID_HOST_KEYPAD_1
HID_HOST_KEYBOARD_LEFT_BRACKET	HID_HOST_KEYPAD_2
HID_HOST_KEYBOARD_RIGHT_BRACKET	HID_HOST_KEYPAD_3
HID_HOST_KEYBOARD_BACK_SLASH	HID_HOST_KEYPAD_4
HID_HOST_KEYBOARD_NON_US_POUND	HID_HOST_KEYPAD_5
HID_HOST_KEYBOARD_SEMICOLON	HID_HOST_KEYPAD_6
HID_HOST_KEYBOARD_APOSTROPHE	HID_HOST_KEYPAD_7
HID_HOST_KEYBOARD_GRAVE_ACCENT	HID_HOST_KEYPAD_8
HID_HOST_KEYBOARD_COMMA	HID_HOST_KEYPAD_9
HID_HOST_KEYBOARD_DOT	HID_HOST_KEYPAD_0
HID_HOST_KEYBOARD_SLASH	HID_HOST_KEYPAD_DOT
HID_HOST_KEYBOARD_CAPS_LOCK	HID_HOST_KEYBOARD_NON_US_SLASH
HID_HOST_KEYBOARD_F1	HID_HOST_KEYBOARD_APPLICATION
HID_HOST_KEYBOARD_F2	HID_HOST_KEYBOARD_LEFT_CONTROL
HID_HOST_KEYBOARD_F3	HID_HOST_KEYBOARD_LEFT_SHIFT
HID_HOST_KEYBOARD_F4	HID_HOST_KEYBOARD_LEFT_ALT
HID_HOST_KEYBOARD_F5	HID_HOST_KEYBOARD_LEFT_GUI
HID_HOST_KEYBOARD_F6	HID_HOST_KEYBOARD_RIGHT_CONTROL
HID_HOST_KEYBOARD_F7	HID_HOST_KEYBOARD_RIGHT_SHIFT
HID_HOST_KEYBOARD_F8	HID_HOST_KEYBOARD_RIGHT_ALT
HID_HOST_KEYBOARD_F9	HID_HOST_KEYBOARD_RIGHT_GUI

etHID_Host_Boot_Keyboard_Repeat_Indication

This event is dispatched when the local HID Host device generates an automatic repeat key press of a currently pressed key.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
  Byte_t    KeyModifiers;
  Byte_t    Key;
} HID_Host_Boot_Keyboard_Repeat_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
KeyModifiers	Bit mask indicating what modifier keys are currently pressed. For a list of constants currently defined for this member, see the KeyModifiers member of the etHID_Host_Boot_Keyboard_Data_Indication event, above.
Key	The key that is currently being held. For a list of constants currently defined for this member, see the Key member of the etHID_Host_Boot_Keyboard_Data_Indication event, above.

etHID_Host_Boot_Mouse_Data_Indication

This event is dispatched when the local HID Host device receives a Boot Mode Mouse Report which is parsed by the HID Host subsystem.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
  SByte_t   CX;
  SByte_t   CY;
  Byte_t    ButtonState;
  SByte_t   CZ;
} HID_Host_Boot_Mouse_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
CX	Amount of relative mouse movement in the sideways direction.
CY	Amount of relative mouse movement in the forward/backward direction.
ButtonState	Bit mask which indicates button activity. The following constants are currently defined: HID_HOST_LEFT_BUTTON_UP HID_HOST_LEFT_BUTTON_DOWN HID_HOST_RIGHT_BUTTON_UP HID_HOST_RIGHT_BUTTON_DOWN HID_HOST_MIDDLE_BUTTON_UP HID_HOST_MIDDLE_BUTTON_DOWN
CZ	Amount of relative mouse movement in the Z axis. Usually, this is the mouse wheel.

etHID_Host_Data_Indication

This event is dispatched when the local HID Host device receives a HID Report (DATA Transaction) on the Interrupt Channel which is not processed internally. A Report is processed internally (and NOT generate this event) only if the Report is a Boot-mode Report and connection has the PARSE_BOOT flag set.

Return Structure:

```
typedef struct
{
  BD_ADDR_t BD_ADDR;
  Word_t    ReportLength;
  Byte_t    ReportDataPayload;
} HID_Host_Data_Indication_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
ReportLength	The size of the Report to which the Report Data Payload member points.
ReportDataPayload	Pointer to the Report Data received as part as part of the DATA Transaction on the Interrupt Channel.

etHID_Host_Get_Report_Confirmation

This event is dispatched when the local HID Host device receives a response to an outstanding GET_REPORT Transaction.

Return Structure:

```
typedef struct
{
  BD_ADDR_t                BD_ADDR;
  HID_Result_Type_t        Status;
  HID_Report_Type_Type_t   ReportType;
  Word_t                   ReportLength;
  Byte_t                   *ReportDataPayload;
} HID_Host_Get_Report_Confirmation_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
Status	The Result Type for this Response. The following Result Type indicates that a Report was returned by the remote HID Device: rtData The following Result Types indicates that the GET_REPORT request was unsuccessful: rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest

rtErrInvalidParameter
rtErrUnknown
rtErrFatal

All other Result Types are invalid for this event.

ReportType	The Report Type of the received Report. This member is only valid 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 to which the Report Data Payload member points. This member is only valid when the Status member is set to rtData.
ReportDataPayload	Pointer to the received Report Data. This member is only valid when the Status member is set to rtData.

etHID_Host_Set_Report_Confirmation

This event is dispatched when the local HID Host device receives a response to an outstanding SET_REPORT Transaction.

Return Structure:

```
typedef struct
{
  BD_ADDR_t           BD_ADDR;
  HID_Result_Type_t   Status;
} HID_Host_Set_Report_Confirmation_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
Status	The Result Type for this Response. The following Result Type indicates that the SET_REPORT request was successfull: rtSuccessful The following Result Types indicate that the request was unsuccessful: rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal All other Result Types are invalid for this event.

etHID_Host_Get_Protocol_Confirmation

This event is dispatched when the local HID Host device receives a response to an outstanding GET_PROTOCOL Transaction.

Return Structure:

```
typedef struct
{
  BD_ADDR_t             BD_ADDR;
  HID_Result_Type_t     Status;
  HID_Protocol_Type_t   Protocol;
} HID_Host_Get_Protocol_Confirmation_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
Status	The Result Type for this Response. The following Result Type indicates that the GET_PROTOCOL request was successfull: rtData The following Result Types indicate the request was unsuccessful: rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal All other Result Types are invalid for this event.
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_Host_Set_Protocol_Confirmation

This event is dispatched when the local HID Host device receives a response to an outstanding SET_PROTOCOL Transaction.

Return Structure:

```
typedef struct
{
  BD_ADDR_t             BD_ADDR;
  HID_Result_Type_t     Status;
} HID_Host_Set_Protocol_Confirmation_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
Status	The Result Type for this Response. The following Result Type indicates that the SET_PROTOCOL request was successfull: rtSuccessful The following Result Types indicate the request was unsuccessful: rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal All other Result Types are invalid for this event.

etHID_Host_Get_Idle_Confirmation

This event is dispatched when the local HID Host device receives a response to an outstanding GET_IDLE Transaction.

Return Structure:

```
typedef struct
{
  BD_ADDR_t          BD_ADDR;
  HID_Result_Type_t  Status;
  Byte_t             IdleRate;
} HID_Host_Get_Idle_Confirmation_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
Status	The Result Type for this Response. The following Result Type indicates that the GET_IDLE request was successfull: rtData The following Result Types indicate the request was unsuccessful: rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal All other Result Types are invalid for this event.

IdleRate	The current Idle Rate. Note that the Idle Rate has a resolution of 4 milliseconds (providing a range of 0.004 seconds to 1.02 seconds). A value of zero indicates that Idle Reporting is disabled. This member is only valid when the Status member is set to rtData.

etHID_Host_Set_Idle_Confirmation

This event is dispatched when the local HID Host device receives a response to an outstanding SET_IDLE Transaction.

Return Structure:

```
typedef struct
{
  BD_ADDR_t           BD_ADDR;
  HID_Result_Type_t   Status;
} HID_Host_Set_Idle_Confirmation_t;
```

Event Parameters:

BD_ADDR	Address of the remote HID Device.
Status	The Result Type for this Response. The following Result Type indicates that the SET_IDLE request was successfull: rtSuccessful The following Result Types indicate the request was unsuccessful: rtNotReady rtErrInvalidReportID rtErrUnsupportedRequest rtErrInvalidParameter rtErrUnknown rtErrFatal All other Result Types are invalid for this event.

3. File Distributions

The header files that are distributed with the Bluetooth HID Host profile sub-system library are listed in the table below.

File	Contents/Description
HIDHAPI.h	Bluetooth HID Host profile sub-system API definitions
SS1BTHIDH.h	Bluetooth HID Host profile sub-system Include file