Health Device Profile (HDP)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
January 10, 2014

Louisville, KY www.stonestreetone.com

Table of Contents

1. INTRODUCTION 4

1.1 Scope 4

1.2 Applicable Documents 5

1.3 Acronyms and Abbreviations 5

2. HDP PROFILE PROGRAMMING INTERFACE 6

2.1 HDP Profile Commands 6

HDP_Initialize 7
HDP_Cleanup 7
HDP_Connect_Request_Response 8
HDP_Register_Instance 8
HDP_UnRegister_Instance 9
HDP_Register_SDP_Record 9
HDP_Set_Connection_Mode 10
HDP_Register_Endpoint 10
HDP_Connect_Remote_Instance 11
HDP_Close_Connection 12
HDP_Create_Data_Channel_Request 12
HDP_Create_Data_Channel_Response 13
HDP_Disconnect_Data_Channel 13
HDP_Reconnect_Data_Channel_Request 14
HDP_Reconnect_Data_Channel_Response 14
HDP_Abort_Data_Channel_Request 15
HDP_Delete_Data_Channel 15
HDP_Write_Data 15
HDP_Sync_Get_Bluetooth_Clock_Value 16
HDP_Sync_Capabilities_Request 16
HDP_Sync_Capabilities_Response 17
HDP_Sync_Set_Request 18
HDP_Sync_Set_Response 18
HDP_Sync_Info_Indication 19

2.2 Health Device Profile Event Callback Prototypes 19

HDP_Event_Callback_t 19

2.3 Health Device Profile Events 21

etHDP_Connect_Request_Indication 22
etHDP_Control_Connect_Indication 23
etHDP_Control_Connect_Confirmation 23
etHDP_Control_Disconnect_Indication 23
etHDP_Control_Create_Data_Link_Indication 24
etHDP_Control_Create_Data_Link_Confirmation 24
etHDP_Control_Reconnect_Data_Link_Indication 25
etHDP_Control_Reconnect_Data_Link_Confirmation 25
etHDP_Control_Abort_Data_Link_Indication 25

etHDP_Control_Abort_Data_Link_Confirmation....26
etHDP_Control_Delete_Data_Link_Indication....26
etHDP_Control_Delete_Data_Link_Confirmation....26
etHDP_Data_Link_Connect_Indication....27
etHDP_Data_Link_Connect_Confirmation....27
etHDP_Data_Link_Disconnect_Indication....27
etHDP_Data_Link_Data_Indication....28
etHDP_Sync_Capabilities_Indication....28
etHDP_Sync_Capabilities_Confirmation....28
etHDP_Sync_Set_Indication....29
etHDP_Sync_Set_Confirmation....29
etHDP_Sync_Info_Indication....30

3. FILE DISTRIBUTIONS....31

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 HDP 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 HDP Profile library.

1.1 Scope

This reference manual provides information on the HDP 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

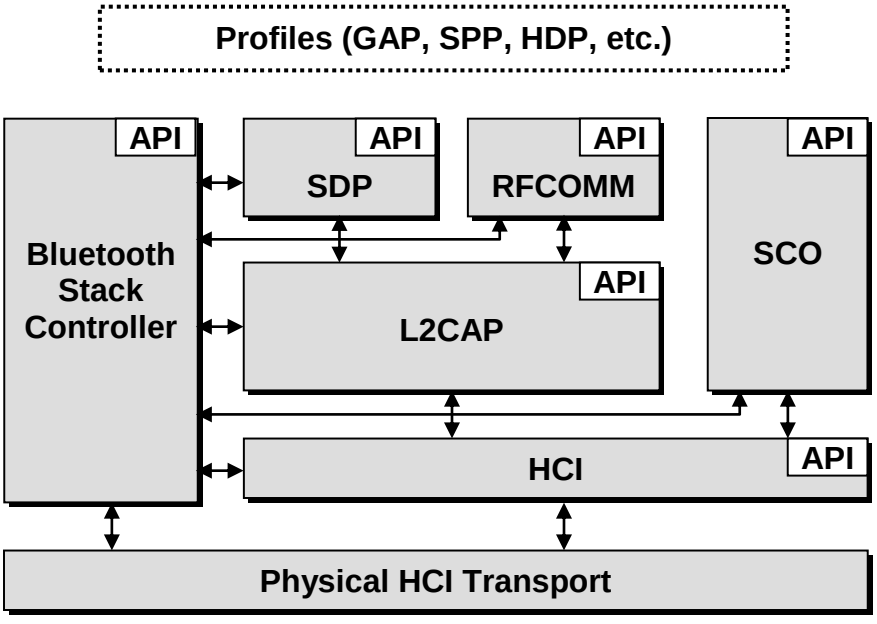

Figure 1-1 Stonestreet One Bluetooth Protocol Stack

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*, version 4.0 + BR/EDR, June 30, 2010.
2. *Specification of the Bluetooth System, Volume 3, Core System Package*, version 4.0 + BR/EDR, June 30, 2010.
3. *Health Device Profile*, version 1.0, June 26, 2008.
4. *Bluetooth Assigned Numbers,* version 1.1, February 22, 2001.
5. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual,* version 4.0.1, January 10, 2013.

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
HS	High Speed
LE	Low Energy
LSB	Least Significant Bit
MSB	Most Significant Bit
SDP	Service Discovery Protocol
SPP	Serial Port Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. HDP Profile Programming Interface

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

2.1 HDP Profile Commands

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

Function	Description
HDP_Initialize	Initializes the HDP layer.
HDP_Cleanup	Cleans up a previously initialized HDP instance.
HDP_Connect_Request_Response	Responds to a request to connect to a local HDP instance.
HDP_Register_Instance	Registers a local HDP instance so that remote HDP profile/application can connect.
HDP_UnRegister_Instance	Un-registers a local instance so that remote profile/application can no longer connect.
HDP_Register_SDP_Record	Adds a Health Device Profile SDP record to the SDP database.
HDP_Set_Connection_Mode	Controls the HDP instance connection mode.
HDP_Register_Endpoint	Registers an endpoint on a specified HDP instance.
HDP_Connect_Remote_Instance	Connects to a remote HDP instance, establishing an L2CAP channel for HDP control.
HDP_Close_Connection	Disconnects a remote HDP control channel.
HDP_Create_Data_Channel_Request	Initiates a connection to a remote HDP endpoint.
HDP_Create_Data_Channel_Response	Initiates a response for a connection request received from a remote HDP instance.
HDP_Disconnect_Data_Channel	Closes an HDP data connection.
HDP_Reconnect_Data_Channel_Request	Initiates a re-connect to a remote HDP endpoint.
HDP_Reconnect_Data_Channel_Response	Initiates a response for a reconnection request received from a remote HDP instance.
HDP_Abort_Data_Channel_Request	Initiates an abort to a remote HDP endpoint.
HDP_Delete_Data_Channel	Performs a disconnect, if currently connected, and removes all reference information about the data channel.

HDP_Write_Data	Sends data over a specified data channel.
HDP_Sync_Get_Bluetooth_Clock_Value	Retrieves the Bluetooth clock information from the specified HDP instance.
HDP_Sync_Capabilities_Request	Sends a sync capabilities request to a remote HDP instance.
HDP_Sync_Capabilities_Response	Initiates a response for a sync capabilities request received from a remote HDP instance.
HDP_Sync_Set_Request	Sends a sync set request to the remote HDP instance.
HDP_Sync_Set_Response	Initiates a response for a sync set request received from a remote HDP instance.
HDP_Sync_Info_Indication	Sends a sync info indication to the remote HDP instance.

HDP_Initialize

This initializes the HDP layer and must be called before any other HDP function.

Prototype:

int BTPSAPI **HDP_Initialize**(unsigned int BluetoothStackID);

Parameters:

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

Return:

A successful return code will be a HDP instance ID that can be used to reference the opened/initialized HDP module in ALL other functions in this module

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Cleanup

This function is responsible for responding to an individual request to connect to a local HDP instance.

Prototype:

int BTPSAPI **HDP_Cleanup**(unsigned int BluetoothStackID);

Parameters:

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

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Connect_Request_Response

This function is responsible for responding to an individual request to connect to a local HDP instance.

Prototype:

int BTPSAPI **HDP_Connect_Request_Response**(unsigned int BluetoothStackID, unsigned int HDPID, Boolean_t AcceptConnection);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize().
HDPID	Unique MCL identifier of the HDP connection for which a connection request was received.
AcceptConnection	Boolean value specifying whether to accept the connection.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Register_Instance

This function will register a Local HDP Instance that a remote HDP profile/application can connect to.

Prototype:

int BTPSAPI **HDP_Register_Instance**(unsigned int BluetoothStackID, Word_t ControlPSM, Word_t DataPSM, Byte_t SupportedProcedures, HDP_Event_Callback_t EventCallback, unsigned long CallbackParameter);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize().
ControlPSM	The L2CAP control port used by a remote HDP Instance.
DataPSM	The L2CAP data port used by a remote HDP Instance.
SupportedProcedures	Defines the features supported by this instance.
EventCallback	A callback function to be invoked with the CallbackParameter whenever there are any events of interest for the profile.
CallbackParameter	Value of the parameter used by the Event Callback.

Return:

If successful, the return value represents the HDP instance identifier. This value is used with other HDP functions to identify the instance to operate on.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_UnRegister_Instance

This function will un-register a local instance so that remote profiles/applications can no longer connect to this profile.

Prototype:

int BTPSAPI **HDP_UnRegister_Instance**(unsigned int BluetoothStackID,
unsigned int HDPID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP Instance Identifier.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Register_SDP_Record

This function adds a generic Health Device Profile 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 HDP_Un_Register_SDP_Record**( )** to SDP_Delete_Service_Record(), and is defined as follows:

   HDP_Un_Register_SDP_Record()
   (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.

Prototype:

int BTPSAPI **HDP_Register_SDP_Record**(unsigned int BluetoothStackID,
unsigned int HDPID, char *ServiceName, char *ProviderName,
Word_t *SDPServiceRecordHandle);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	Unique MCL identifier of the HDP connection for which a connection request was received.
ServiceName	Name to appear in the SDP database for this service.
ProviderName	Provider name to associate with the SDP record.
SDPServiceRecordHandle	Returned handle to the SDP database entry that may be used to remove the entry at a later time.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Set_Connection_Mode

This function sets the HDP server connection mode. This can be used to control the devices that are allowed to access this profile.

Prototype:

int BTPSAPI **HDP_Set_Connection_Mode**(unsigned int BluetoothStackID, unsigned int HDPID, HDP_Connection_Mode_t ConnectionMode);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
ConnectionMode	Mode to use for connections. This can be set to automatic or manual connection acceptance. This value can be one of the following: hcmAutomaticAccept hcmManualAccept

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Register_Endpoint

This function is used to register an endpoint on a specified HDP instance.

Prototype:

int BTPSAPI **HDP_Register_Endpoint**(unsigned int BluetoothStackID, unsigned int HDPID, HDP_MDEP_Info_t *HDPMDEPInfoPtr, HDP_Event_Callback_t EventCallback, unsigned long CallbackParameter);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
HDPMDEPInfoPtr	A pointer to a structure that contains information about the MDEP that is being registered.
EventCallback	The callback function to be invoked with the CallbackParameter whenever there are any events of interest for the profile.
CallbackParameter	User defined value that is returned by the event callback function.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Connect_Remote_Instance

This function is responsible for initiating a connection to a remote HDP instance. It should be noted that a local HDP instance is required to support this function. This provides an instance for the remote instance to re-connect to.

Prototype:

int BTPSAPI **HDP_Connect_Remote_Instance**(unsigned int BluetoothStackID, unsigned int HDPID, BD_ADDR_t RemoteBD_ADDR, Word_t ControlPSM, Word_t DataPSM);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier of the local HDP instance.
RemoteBD_ADDR	BD_ADDR of the remote device that hosts the HDP instance.
ControlPSM	Identifies the control PSM value of the remote instance.
DataPSM	Identifies the data PSM value of the remote instance.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Close_Connection

This function is responsible for disconnecting a connection between an HDP Initiator and Acceptor.

Prototype:

int BTPSAPI **HDP_Close_Connection**(unsigned int BluetoothStackID, unsigned int HDPID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Create_Data_Channel_Request

This function is responsible for initiating a connection to a remote HDP endpoint.

Prototype:

int BTPSAPI **HDP_Create_Data_Channel_Request**(unsigned int BluetoothStackID, unsigned int HDPID, Byte_t MDEP_ID, HDP_Device_Role_t Role, HDP_Channel_Mode_t ChannelMode, HDP_Channel_Config_Info_t *ConfigInfoPtr);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
MDEP_ID	Identifies the endpoint that is being targeted in this request.
Role	Identifies the local role as source or sync.
ChannelMode	Defines the channel as either reliable or streaming.
ConfigInfoPtr	Contains configuration information used by L2CAP to negotiate a reliable or streaming channel.

Return:

Success is denoted by a positive return value. This value represents the DataLinkID and is used in other HDP functions to identify the open connection.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Create_Data_Channel_Response

This function is is responsible for initiating a response for a connection request received from a remote HDP instance.

Prototype:

int BTPSAPI **HDP_Create_Data_Channel_Response**(unsigned int BluetoothStackID, unsigned int HDPID, unsigned int DataLinkID, Byte_t ResponseCode, HDP_Channel_Mode_t ChannelMode, HDP_Channel_Config_Info_t *ConfigInfoPtr);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
DataLinkID	Identifies the data connection that was received in the request.
ResponseCode	Indicates acceptance/rejection of the request.
ChannelMode	Defines the channel as either reliable or streaming.
ConfigInfoPtr	Contains configuration information used by L2CAP to negotiate a reliable or streaming channel.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Disconnect_Data_Channel

This function is responsible for the disconnetion of a data channel. If this function is called between the time of a successful call to HSP_Create_Data_Channel() and an event that indicates the status of the connection, then this will initiate an HDP abort sequence.

Prototype:

int BTPSAPI **HDP_Disconnect_Data_Channel**(unsigned int BluetoothStackID, unsigned int HDPID, unsigned int DataLinkID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
DataLinkID	Identifies the data connection that is to be disconnected.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Reconnect_Data_Channel_Request

This function is responsible for initiating a re-connecting to a remote HDP endpoint.

Prototype:

int BTPSAPI **HDP_Reconnect_Data_Channel_Request**(unsigned int BluetoothStackID, unsigned int HDPID, unsigned int DataLinkID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
DataLinkID	Identifies the data connection that has previously been created and is to be reconnected.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Reconnect_Data_Channel_Response

This function is responsible for initiating a response for a reconnection request received from a remote HDP instance.

Prototype:

int BTPSAPI **HDP_Reconnect_Data_Channel_Response**(unsigned int BluetoothStackID, unsigned int HDPID, unsigned int DataLinkID, Byte_t ResponseCode);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
DataLinkID	Identifies the data link that is to be reconnected.
ResponseCode	Indicates acceptance/rejection of the request.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Abort_Data_Channel_Request

This function is responsible for initiating an abort to a remote HDP endpoint.

Prototype:

int BTPSAPI **HDP_Abort_Data_Channel_Request**(unsigned int BluetoothStackID, unsigned int HDPID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Delete_Data_Channel

This function is responsible for the deletion of data channel information.

Prototype:

int BTPSAPI **HDP_Delete_Data_Channel**(unsigned int BluetoothStackID, unsigned int HDPID, unsigned int DataLinkID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
DataLinkID	Identifies the data channel that is to be deleted.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Write_Data

This function is responsible for sending data over a specified data channel.

Prototype:

int BTPSAPI **HDP_Write_Data**(unsigned int BluetoothStackID, unsigned int HDPID, unsigned int DataLinkID, unsigned int DataLength, unsigned char *DataPtr);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
DataLinkID	Identifies the data channel over which the data is to be sent.
DataLength	Identifies the number of octets that are to be sent.
DataPtr	Pointer to the data to be sent.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Sync_Get_Bluetooth_Clock_Value

This function Is responsible for the retreiving the Blueooth clock value of the specified HDP instance.

Prototype:

int BTPSAPI **HDP_Sync_Get_Bluetooth_Clock_Value**(unsigned int BluetoothStackID, unsigned int HDPID, DWord_t *ClockValue, Word_t *Accuracy);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
ClockValue	A pointer to memory where the value read from the Bluetooth device will be returned.
Accuracy	Identifies the accuracy of the value that was retrieved.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Sync_Capabilities_Request

This function is responsible for the sending of a sync capabilities request to the remote HDP instance.

Prototype:

int BTPSAPI **HDP_Sync_Capabilities_Request**(unsigned int BluetoothStackID, unsigned int HDPID, Word_t RequiredAccuracy);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
RequiredAccuracy	Identifies the minimum accuracy that is required by the sync master.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Sync_Capabilities_Response

This function is responsible for initiating a response for a sync capabilities request received from a remote HDP instance.

Prototype:

int BTPSAPI **HDP_Sync_Capabilities_Response**(unsigned int BluetoothStackID, unsigned int HDPID, Byte_t AccessResolution, Word_t SyncLeadTime, Word_t NativeResolution, Word_t NativeAccuracy, Byte_t ResponseCode);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
AccessResolution	The resolution at which the clock can be accessed in baseband half-slots.
SyncLeadTime	Defines the minimum time in milliseconds required to process a sync request.
NativeResolution	Defines the resolution in microseconds of the local timestamp.
NativeAccuracy	Identifies the accuracy in parts-per-million of the local time stamp.
ResponseCode	Indicates the acceptance/rejection of the request.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Sync_Set_Request

This function is responsible for the sending of a sync set request to the remote HDP instance.

Prototype:

int BTPSAPI **HDP_Sync_Set_Request**(unsigned int BluetoothStackID, unsigned int HDPID, Boolean_t UpdateInformationRequest, DWord_t ClockSyncTime, QWord_t TimestampSyncTime);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
UpdateInformationRequest	Flag that indicates the desire to have sync information indications sent.
ClockSyncTime	Identifies the Bluetooth clock time half slot at which synchronization is requested.
TimestampSyncTime	Indicates the timestamp clock value to be set at the requested Bluetooth clock time.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Sync_Set_Response

This function is responsible for initiating a response for a sync set request received from a remote HDP instance.

Prototype:

int BTPSAPI **HDP_Sync_Set_Response**(unsigned int BluetoothStackID, unsigned int HDPID, DWord_t ClockSyncTime, QWord_t TimestampSyncTime, Word_t TimestampSampleAccuracy, Byte_t ResponseCode);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
ClockSyncTime	The value of the Bluetooth clock at the time the response was sent.
TimestampSyncTime	Identifies the timestamp clock value at the time of the response.

TimestampSimpleAccuracy	Identifies the maximum error in parts-per-million of the clock sample.
ResponseCode	Indicates the acceptance/rejection of the request.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

HDP_Sync_Info_Indication

This function is responsible for the sending of a sync info indication to the remote HDP instance.

Prototype:

int BTPSAPI **HDP_Sync_Info_Indication**(unsigned int BluetoothStackID, unsigned int HDPID, DWord_t ClockSyncTime, QWord_t TimestampSyncTime, Word_t TimestampSampleAccuracy);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().
HDPID	The HDP instance identifier.
ClockSyncTime	The value of the Bluetooth clock at the time the response was sent.
TimestampSyncTime	Identifies the timestamp clock value at the time of the response.
TimestampSimpleAccuracy	Identifies the maximum error in parts-per-million of the clock sample.

Return:

Zero if successful.

On error, a negative value is returned. Refer to the HDPAPI header file for the defined error codes.

2.2 Health Device Profile Event Callback Prototypes

The event callback functions mentioned in the Health Device Profile open commands all accept the callback function described by the following prototype.

HDP_Event_Callback_t

Prototype of callback function passed in one of the HDP open commands.

Prototype:

void (BTPSAPI ***HDP_Event_Callback_t**)(unsigned int BluetoothStackID,
HDP_Event_Data_t *HDP_Event_Data, unsigned long CallbackParameter);

Parameters:

BluetoothStackID — Unique identifier assigned to this Bluetooth Protocol Stack instance via a call to BSC_Initialize().

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

```
typedef struct
{
  HDP_Event_Type_t    Event_Data_Type;
  Word_t              Event_Data_Size;
  union
  {
   HDP_Connect_Request_Indication_Data_t
            *HDP_Connect_Request_Indication_Data;
   HDP_Control_Connect_Indication_Data_t
            *HDP_Control_Connect_Indication_Data;
   HDP_Control_Connect_Confirmation_Data_t
            *HDP_Control_Connect_Confirmation_Data;
   HDP_Control_Disconnect_Indication_Data_t
          *HDP_Control_Disconnect_Indication_Data;
   HDP_Control_Create_Data_Link_Indication_t
            *HDP_Control_Create_Data_Link_Indication_Data;
   HDP_Control_Create_Data_Link_Confirmation_t
            *HDP_Control_Create_Data_Link_Confirmation_Data;
   HDP_Control_Reconnect_Data_Link_Indication_t
            *HDP_Control_Reconnect_Data_Link_Indication_Data;
   HDP_Control_Reconnect_Data_Link_Confirmation_t
          *HDP_Control_Reconnect_Data_Link_Confirmation_Data;
   HDP_Control_Abort_Data_Link_Indication_t
            *HDP_Control_Abort_Data_Link_Indication_Data;
   HDP_Control_Abort_Data_Link_Confirmation_t
            *HDP_Control_Abort_Data_Link_Confirmation_Data;
   HDP_Control_Delete_Data_Link_Indication_t
            *HDP_Control_Delete_Data_Link_Indication_Data;
   HDP_Control_Delete_Data_Link_Confirmation_t
            *HDP_Control_Delete_Data_Link_Confirmation_Data;
   HDP_Data_Link_Connect_Indication_Data_t
            *HDP_Data_Link_Connect_Indication_Data;
   HDP_Data_Link_Connect_Confirmation_Data_t
            *HDP_Data_Link_Connect_Confirmation_Data;
   HDP_Data_Link_Disconnect_Indication_Data_t
            *HDP_Data_Link_Disconnect_Indication_Data;
   HDP_Data_Link_Data_Indication_Data_t
            *HDP_Data_Link_Data_Indication_Data;
   HDP_Sync_Capabilities_Indication_t
            *HDP_Sync_Capabilities_Indication_Data;
```

```
      HDP_Sync_Capabilities_Confirmation_t
            *HDP_Sync_Capabilities_Confirmation_Data;
      HDP_Sync_Set_Indication_t
            *HDP_Sync_Set_Indication_Data;
      HDP_Sync_Set_Confirmation_t
            *HDP_Sync_Set_Confirmation_Data;
      HDP_Sync_Info_Indication_t
            *HDP_Sync_Info_Indication_Data;
   } Event_Data;
} HDP_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 provided in the callback registration.

2.3 Health Device Profile Events

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

Event	Description
etHDP_Connect_Request_Indication	Indicate that a remote service is requesting a connection to the local service.
etHDP_Control_Connect_Indication	Indicate that a remote HDP instance connects to the local HDP instance.
etHDP_Control_Connect_Confirmation	Confirm that the application when an attempt to connect to a remote HDP instance is complete.
etHDP_Control_Disconnect_Indication	Confirm the application when the remote HDP instance disconnects from the local HDP Instance.
etHDP_Control_Create_Data_Link_Indication	Indicate that a response is received for a previous create data link request.
etHDP_Control_Create_Data_Link_Confirmation	Indicate that a response is received for a previous create data link request.
etHDP_Control_Reconnect_Data_Link_Indication	Indicate that a request to reconnect a data link is received from a remote HDP instance.
etHDP_Control_Reconnect_Data_Link_Confirmation	Indicate that a response to a reconnect command is received from the remote HDP instance.

etHDP_Control_Abort_Data_Link_Indication	Indicate that a request to abort a data link create or reconnect operation is received from a remote HDP instance.
etHDP_Control_Abort_Data_Link_Confirmation	Indicate that a response to an abort command is received from the remote HDP instance.
etHDP_Control_Delete_Data_Link_Indication	Indicate that a request to delete a data link is received from a remote HDP instance.
etHDP_Control_Delete_Data_Link_Confirmation	Indicate that a response to a delete command is received from the remote HDP instance.
etHDP_Data_Link_Connect_Indication	Indicate that a data link is successfully established.
etHDP_Data_Link_Connect_Confirmation	Indicate that a create data link operation is complete.
etHDP_Data_Link_Disconnect_Indication	Indicate that an established data link is disconnected from the remote HDP instance.
etHDP_Data_Link_Data_Indication	Indicate that data is received from the remote HDP instance on an open data link.
etHDP_Sync_Capabilities_Indication	Indicate that a sync capabilities request is received from the remote HDP instance.
etHDP_Sync_Capabilities_Confirmation	Indicate that a response to a sync capabilities request received from a remote HDP instance.
etHDP_Sync_Set_Indication	Indicate that a sync set request is received from the remote HDP instance.
etHDP_Sync_Set_Confirmation	Indicate that a response to a sync set request received from a remote HDP instance.
etHDP_Sync_Info_Indication	Indicate that a sync info packet is receeived from a remote HDP instance.

etHDP_Connect_Request_Indication

Indicate that a remote service is requesting a connection to the local service.

Return Structure:

```
typedef struct
{
  unsigned int        HDPID;
  BD_ADDR_t           BD_ADDR;
} HDP_Connect_Request_Indication_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
BD_ADDR	Address of the Bluetooth device making the request.

etHDP_Control_Connect_Indication

Indicate that a remote HDP instance connects to the local HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int        HDPID;
  BD_ADDR_t           BD_ADDR;
} HDP_Control_Connect_Indication_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
BD_ADDR	Address of the Bluetooth device making the request.

etHDP_Control_Connect_Confirmation

Confirm that the application when an attempt to connect to a remote HDP instance is complete.

Return Structure:

```
typedef struct
{
  unsigned int        HDPID;
  BD_ADDR_t           BD_ADDR;
} HDP_Control_Connect_Confirmation_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
Status	Indicates success or failure of the connection.

etHDP_Control_Disconnect_Indication

Confirm the application when the remote HDP instance disconnects from the local HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int HDPID;
} HDP_Control_Disconnect_Indication_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.

etHDP_Control_Create_Data_Link_Indication

Indicate that a create data link request is received from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int          HDPID;
  unsigned int          DataLinkID;
  Byte_t                MDEPID;
  HDP_Channel_Mode_t    ChannelMode;
} HDP_Control_Create_Data_Link_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
MDEPID	Identifies the endpoint that will be supporting the data link.
ChannelMode	Contains profile specific configuration information.

etHDP_Control_Create_Data_Link_Confirmation

Indicate that a response is received for a previous create data link request.

Return Structure:

```
typedef struct
{
  unsigned int          HDPID;
  unsigned int          DataLinkID;
  Byte_t                ResponseCode;
  HDP_Channel_Mode_t    ChannelMode;
} HDP_Control_Create_Data_Link_Confirmation_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
ResponseCode	Indicates the result of the request.
ChannelMode	Contains profile specific configuration information.

etHDP_Control_Reconnect_Data_Link_Indication

Indicate that a request to reconnect a data link is received from a remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int HDPID;
  unsigned int DataLinkID;
} HDP_Control_Reconnect_Data_Link_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being reconnected.

etHDP_Control_Reconnect_Data_Link_Confirmation

Indicate that a response to a reconnect command is received from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  unsigned int   DataLinkID;
  Byte_t         ResponseCode;
} HDP_Control_Reconnect_Data_Link_Confirmation_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
ResponseCode	Indicates the result of the request.

etHDP_Control_Abort_Data_Link_Indication

Indicate that a request to abort a data link create or reconnect operation is received from a remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int HDPID;
  unsigned int DataLinkID;
} HDP_Control_Abort_Data_Link_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is associated with the request.

etHDP_Control_Abort_Data_Link_Confirmation

Indicate that a response to an abort command is received from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  unsigned int   DataLinkID;
  Byte_t         ResponseCode;
} HDP_Control_Abort_Data_Link_Confirmation_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
ResponseCode	Contains the result of the request.

etHDP_Control_Delete_Data_Link_Indication

Indicate that a request to delete a data link is received from a remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int HDPID;
  unsigned int DataLinkID;
} HDP_Control_Delete_Data_Link_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being deleted.

etHDP_Control_Delete_Data_Link_Confirmation

Indicate that a response to a delete command is received from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  unsigned int   DataLinkID;
  Byte_t         ResponseCode;
} HDP_Control_Delete_Data_Link_Confirmation_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
ResponseCode	Contains the result of the request.

etHDP_Data_Link_Connect_Indication

Indicate that a data link is successfully established.

Return Structure:

```
typedef struct
{
   unsigned int HDPID;
   unsigned int DataLinkID;
} HDP_Data_Link_Connect_Indication_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being created.

etHDP_Data_Link_Connect_Confirmation

Indicate that a create data link operation is complete.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  unsigned int   DataLinkID;
  int            Status;
} HDP_Data_Link_Connect_Confirmation_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
Status	Indicates success or failure of the connection.

etHDP_Data_Link_Disconnect_Indication

Indicate that an established data link is disconnected from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int HDPID;
  unsigned int DataLinkID;
} HDP_Data_Link_Disconnect_Indication_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being disconnected.

etHDP_Data_Link_Data_Indication

Indicate that data is received from the remote HDP instance on an open data link.

Return Structure:

```
typedef struct
{
  unsigned int     HDPID;
  unsigned int     DataLinkID;
  Word_t           DataLength;
  Byte_t          *DataPtr;
} HDP_Data_Link_Data_Indication_Data_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
DataLinkID	Identifies the data channel that is being requested.
DataLength	Specifies the amount of data that is pointed to by the DataPtr member.
DataPtr	Pointer to a buffer of DataLength bytes that represents the received data.

etHDP_Sync_Capabilities_Indication

Indicate that a sync capabilities request is received from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  Word_t         RequiredAccuracy;
} HDP_Sync_Capabilities_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
RequiredAccuracy	Indicates the accuracy of the local timestamp required by the sender of the request.

etHDP_Sync_Capabilities_Confirmation

Indicate that a response to a sync capabilities request received from a remote HDP instance.

Return Structure:

```
typedef struct
{
   unsigned int  HDPID;
   Byte_t        AccessResolution;
   Word_t        SyncLeadTime;
   Word_t        NativeResolution;
   Word_t        NativeAccuracy;
   Byte_t        ResponseCode;
} HDP_Sync_Capabilities_Confirmation_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
AccessResolution	Identifies the accuracy of the remote Bluetooth clock.
SyncLeadTime	Identifies the amount of time the remote device requires to access the local clock information.
NativeResolution	Identifies the resolution in microseconds of the remote timestamp.
NativeAccuracy	Identifies the accuracy of the remote timestamp.
ResponseCode	Contains the result of the request.

etHDP_Sync_Set_Indication

Indicate that a sync set request is received from the remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int  HDPID;
  Boolean_t     UpdateInformationRequest;
  DWord_t       ClockSyncTime;
  QWord_t       TimestampSyncTime;
} HDP_Sync_Set_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
UpdateInformationRequest	Indicates the desire to receive sync info indication.
ClockSyncTime	Identifies the Bluetooth clock value at which synchronization should occur.
TimestampSyncTime	Indicates the value to be set for the timestamp at the time of synchronization.

etHDP_Sync_Set_Confirmation

Indicate that a response to a sync set request received from a remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  DWord_t        ClockSyncTime;
  QWord_t        TimestampSyncTime;
  Word_t         TimestampSampleAccuracy;
  Byte_t         ResponseCode;
} HDP_Sync_Set_Confirmation_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
ClockSyncTime	Identifies the Bluetooth clock at the time the response was generated.
TimestampSyncTime	Identifies the timestamp value at the time the response was generated.
TimestampSampleAccuracy	Identifies the maximum error that may exist in the timestamp value.
ResponseCode	Contains the result of the request.

etHDP_Sync_Info_Indication

Indicate that a sync info packet is received from a remote HDP instance.

Return Structure:

```
typedef struct
{
  unsigned int   HDPID;
  DWord_t        ClockSyncTime;
  QWord_t        TimestampSyncTime;
  Word_t         TimestampSampleAccuracy;
} HDP_Sync_Info_Indication_t;
```

Event Parameters:

HDPID	Identifier of the HDP instance.
ClockSyncTime	Identifies the Bluetooth clock at the time the response was generated.
TimestampSyncTime	Identifies the timestamp value at the time the response was generated.
TimestampSampleAccuracy	Identifies the maximum error that may exist in the timestamp value.

3. File Distributions

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

File	Contents/Description
HDPAPI.h	Bluetooth Health Device Profile API definitions
SS1BTHDP.h	Bluetooth Health Device Profile Include file