Hands-Free Profile (HFRE)

Application Programming Interface Reference Manual

Profile Version: 1.5

Release: 4.0.1
January 10, 2014

Louisville, KY www.stonestreetone.com

Table of Contents

1. Introduction

Bluetopia®, the Bluetooth Protocol Stack by Stonestreet One provides a software architecture that encapsulates the upper functionality of the Bluetooth Protocol Stack. More specifically, this stack is a software solution that resides above the Physical HCI (Host Controller Interface) Transport Layer and extends through the L2CAP (Logical Link Control and Adaptation Protocol) and the SCO (Synchronous Connection-Oriented) Link layers. In addition to basic functionality at these layers, the Bluetooth Protocol Stack by Stonestreet One provides implementations of the Service Discovery Protocol (SDP), RFCOMM (the Radio Frequency serial COMMunications port emulator), and several of the Bluetooth Profiles. Program access to these layers, services, and profiles is handled via Application Programming Interface (API) calls.

This document focuses on the API reference that contains a description of all programming interfaces for the Bluetooth Hands-Free Profile provided by Bluetopia. Chapter 2 contains a description of the programming interface for this profile. And, Chapter 3 contains the header file name list for the Bluetooth Hands-Free Profile library.

1.1 Scope

This reference manual provides information on the Hands-Free Profile API identified in Figure 1-1 below. These APIs are available on the full range of platforms supported by Stonestreet One:

- Windows
- Windows Mobile
- Windows CE
- Linux
- QNX
- Other Embedded OS

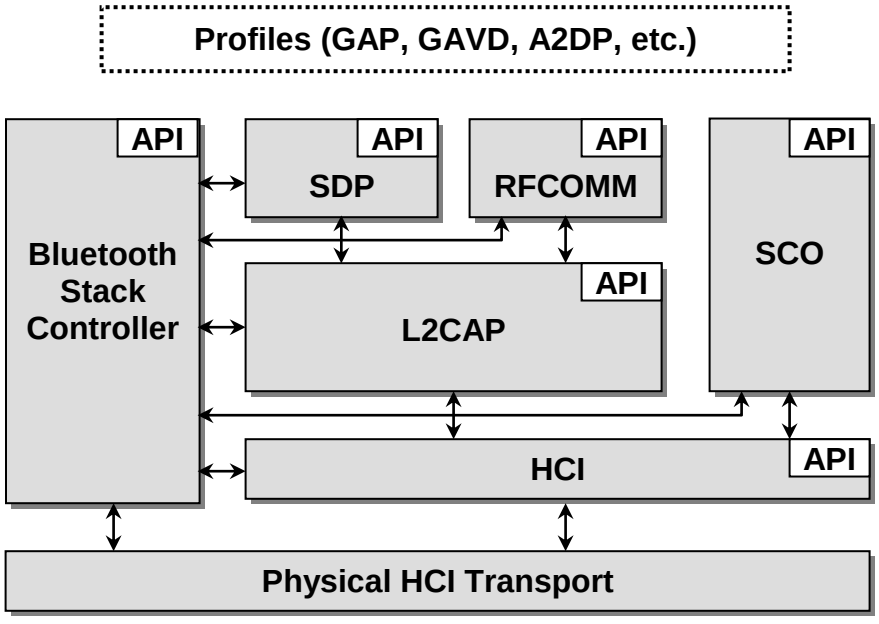

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

1.2 Applicable Documents

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

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

18. *Specification of the Bluetooth System, Volume 2, Core System Package [BR/EDR Controller Volume]*, version 4.0, June 30, 2010.
19. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 4.0, June 30, 2010.
20. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 4.0, June 30, 2010.
21. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 4.0, June 30, 2010.
22. *Specification of the Bluetooth System, Volume 6, Core System Package [Low Energy Controller Volume]*, version 4.0, June 30, 2010.
23. *Bluetooth Assigned Numbers,* version 2.25, May 24th, 2004.
24. *Hands-Free Profile 1.5*, revision V10r00, November 25, 2005.
25. *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.
26. *Digital cellular telecommunications system (Phase 2+); AT command set for GSM Mobile Equipment (ME) (GSM 07.07),* version 7.5.0, Release 1998.
27. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual,* version 4.0.1, January 10, 2013.

Possible error returns are listed for each API function call. These are the *most likely* errors, but in fact programmers should allow for the possibility of any error listed in the BTerrors.h header file to occur as the value of a function return.

1.3 Acronyms and Abbreviations

Acronyms and abbreviations used in this document and other Bluetooth specifications are listed in the table below.

Term	Meaning
API	Application Programming Interface
BD_ADDR	Bluetooth Device Address
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Data Rate
HS	High Speed
LE	Low Energy
LSB	Least Significant Bit

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

2. Hands-Free Profile Programming Interface

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

2.1 Hands-Free Profile Commands

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

Function	Description
HFRE_Open_HandsFree_Server_Port	Opens a hands-free server on the specified Bluetooth SPP serial port.
HFRE_Open_Audio_Gateway_Server_Port	Opens an audio gateway server on the specified Bluetooth SPP serial port.
HFRE_Close_Server_Port	Un-registers a HFRE Port server (which was registered by a successful call to either the HFRE_Open_HandsFree_Port() or the HFRE_Open_Audio_Gateway_Server_Port() function).
HFRE_Open_Port_Request_Response	Responds to request to connect to a server (either Handsfree or an Audo Gateway server).
HFRE_Register_HandsFree_SDP_Record	Adds a generic hands-free role service record to the SDP database.
HFRE_Register_Audio_Gateway_SDP_Record	Adds a generic audio gateway role service record to the SDP database.
HFRE_Un_Register_ SDP_Record	Removes a generic hands-free service record or audio gateway service record from the SDP database.
HFRE_Open_Remote_HandsFree_Port	Opens a remote hands-free port on the specified remote device.
HFRE_Open_Remote_Audio_Gateway_Port	Opens a remote audio gateway port on the specified remote device.
HFRE_Close_Port	Closes a HFRE port that was previously opened by any of the following mechanisms: - Successful call to HFRE_Open_Remote_ HandsFree_Port() function.

	- Successful call to HFRE_Open_Remote_Audio_Gateway_Port( ) function. - Incoming open request (Hands-Free or Audio Gateway) which the server was opened with either the HFRE_Open_ HandsFree _Server_Port() or the HFRE_Open_Audio_Gateway_Server_Port() functions.
HFRE_Query_Remote_Control_Indicator_ Status	Queries the remote control indicator status. Only hands-free units that have a valid service level connection may perform this function.
HFRE_Send_Control_Indicator_Request_ Response	Sends a response to a request by the remote Hands Free device to query the current control indicators (only may be called in response to a etHFRE_Control_Indicator_Request_Indication event).
HFRE_Update_Current_Control_Indicator _Status	Updates the current control indicator status. This function may only be performed by audio gateways that have a valid service level connection and event reporting activated (set via the Enable Remote Event Indicator Event Notification function by the remote device).
HFRE_Update_Current_Control_Indicator _Status_By_Name	Updates the current control indicator status by the name of the indicator. This function may only be performed by audio gateways that have a valid service level connection and event reporting activated (set via the Enable Remote Event Indicator Event Notification function by the remote device).
HFRE_Enable_Remote_Indicator_Event_ Notification	Enables or disables the indicator event notification on the remote device. When enabled, the remote device will send unsolicited responses to update the local device of the current control indicator values. Only hands-free units that have a valid service level connection may perform this function.
HFRE_Query_Remote_Call_Holding_Mul tiparty_Service_Support	Queries the call holding and multiparty services supported by the remote device. This function should be used by hands-free units that support three-way calling and call waiting to determine the features supported by the audio gateway. This function can only be used if a valid service level

	connection exists.
HFRE_Send_Call_Holding_Multiparty_Selection	Allows the control of multiple concurrent calls and provides means for holding calls, releasing calls, switching between two calls and adding a call to a multiparty conference. Only hands-free units that support call waiting and multiparty services and have a valid service level connection may perform this function. The selection should be one that is supported by the remote audio gateway as queried via a call to the Query Remote Call Holding Multiparty Service Support function.
HFRE_Enable_Remote_Call_Waiting_Notification	Enables or disables call waiting notification on the remote audio gateway. By default, the call waiting notification is enabled in the network but disabled for notification via the service level connection. Only hands-free units having a valid service level connection may perform this function. This function may only be used to enable call waiting notification if the local hands-free unit supports call waiting and multiparty services. It may, however, be used to disable these service by all hands-free units.
HFRE_Send_Call_Waiting_Notification	Sends call waiting notifications to the remote device. This function may only be performed by audio gateways that have call waiting notification enabled and have a valid service level connection.
HFRE_Enable_Remote_Call_Line_Identification_Notification	Enables or disabling call waiting notification on the remote audio gateway. By default the call line identification notification via the service level connection is disabled. This function may only be performed by hands-free units for which a valid service level connection exists. This function may only be used to enable call line notification if the local hands-free unit supports call line identification, it may however be used to disable this service by all hands-free units.
HFRE_Send_Call_Line_Identification_Notification	Sends call line identification notifications to the remote device. Only audio gateways that have call line identification notification enabled and have a valid service level connection may perform this function.
HFRE_Disable_Remote_Echo_Cancelatio	Disables echo cancellation and noise reduction on the remote device. This function may be

n_Noise_Reduction	performed by both the hands-free unit and the audio gateway for which a valid service level connection exists but no audio connection exists.
HFRE_Dial_Phone_Number	Dials a phone number on the remote audio gateway. This function may only be performed by hands-free units for which a valid service level connection exists.
HFRE_Dial_Phone_Number_From_Memory	Dials a phone number at a memory location found on the remote audio gateway. This function may only be performed by hands-free units for which a valid service level connection exists.
HFRE_Redial_Last_Phone_Number	Redials the last number dialed on the remote audio gateway. This function may only be performed by hands-free units for which a valid service level connection exists.
HFRE_Ring_Indication	Sends ring indications to the remote hands-free unit. This function may only be performed by audio gateways for which a valid service level connection exists.
HFRE_Answer_Incoming_Call	Sends the command to answer incoming calls on a remote audio gateway. This function may only be performed by hands-free units for which a valid service level connection exists.
HFRE_Enable_Remote_InBand_Ring_Tone_Setting	Enables or disables in-band ring tone capabilities. This function may only be performed by audio gateways for which a valid service level connection exists. This function may only be used to enable in-band ring tone capabilities if the local audio gateway supports this feature.
HFRE_Transmit_DTMF_Code	Transmits DTMF codes to the remote audio gateway to be sent as a DTMF Tone over an on-going call. Hands-free units for which a valid service level connection exists may only perform this function.
HFRE_Set_Remote_Voice_Recognition_Activation	Activates and deactivates the voice recognition which resides on the remote audio gateway when called by a hands-free unit. When called by an audio gateway, this function informs the remote hands-free unit of the current activation state of the local voice recognition function. Only local devices that were opened with the supported feature bit set for voice recognition may call this

	function.
HFRE_Set_Remote_Speaker_Gain	Allows synchronization with and setting of the remote device's speaker gain. This function may only be performed if a valid service level connection exists. When called by a hands-free unit this function is provided as a means to inform the remote audio gateway of the current speaker gain value. When called by an audio gateway this function provides a means for the audio gateway to control the speaker gain of the remote hands-free unit.
HFRE_Set_Remote_Microphone_Gain	Allows synchronization with and setting of the remote device's microphone gain. This function may only be performed if a valid service level connection exists. When called by a hands-free unit this function is provided as a means to inform the remote audio gateway of the current microphone gain value. When called by an audio gateway this function provides a means for the audio gateway to control the microphone gain of the remote hands-free unit.
HFRE_Voice_Tag_Request	Retrieves a phone number to associate with a unique voice tag to be stored in memory by the local hands-free unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. The hands-free unit must also support voice recognition to be able to use this function. When this function is called, no other function may be called until a voice tag response is received from the remote audio gateway.
HFRE_Voice_Tag_Response	Transmits a phone number to be associated with a unique voice tag by the remote hands-free unit. This function may only be performed by audio gateways that have received a voice tag request indication. Alternately, HFRE_Send_Terminating_Response() can be used to return an error to a request received from the remote hands-free device.
HFRE_Hang_Up_Call	Transmits a hang-up command to the remote audio gateway. Hands-free units for which a valid service level connection exists may only perform this function.

HFRE_Setup_Audio_Connection	Sets up an audio connection between the local and remote device. Either an audio gateway or a hands-free unit for which a valid service level connection exists may use this function.
HFRE_Release_Audio_Connection	Releases an audio connection that was set up via a call to HFRE_Setup_Audio_Connection(). Either an audio gateway or a hands-free unit may use this function.
HFRE_Send_Audio_Data	Provides the local entity a mechanism of sending SCO audio data to the remote entity. This function can only be called once an audio connection has been established.
HFRE_Query_Remote_Current_Calls_List	Retrieves the current call list from the remote audio gateway unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a current call list response is received from the remote audio gateway.
HFRE_Send_Current_Calls_List	Sends the current calls list to the remote hands-free device. This function will NOT send the Phonebook Name entry in the HFRE_Current_Call_List_Entry_t structure. Use HFRE_Send_Current_Calls_List_With_Phonebook_Name () if this field should be sent. This function may only be performed by an audio gateway that has received a request for the current calls list and has a valid service level connection. This function should be called once for each call in the current list, with the last call setting the FinalEntry parameter to TRUE. Alternately, HFRE_Send_Terminating_Response() can be used to return an empty OK response (no current calls) or an error to a request received from the remote hands-free device.
HFRE_Send_Current_Calls_List_With_ Phonebook_Name	Sends the current calls list to the remote hands-free device. This function also optionally sends the phonebook name with each entry, which is NOT included in the specification. This function may only be performed by an audio gateway that has received a request for the current calls list and has a valid service level connection. This function should be called once for each call in the current list, with the last call setting the

	FinalEntry parameter to TRUE. Alternately, HFRE_Send_Terminating_Response() can be used to return an empty OK response (no current calls) or an error to a request received from the remote hands-free device.
HFRE_Set_Network_Operator_Selection_Format	Sets the network operator selection format for remote audio gateway unit. This format is a specification defined value that is not variable. This function must be called before calling HFRE_Query_Remote_Network_Operator_Selection(). This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.
HFRE_Query_Remote_Network_Operator_Selection	Retrieves the remote network operator selection from the remote audio gateway unit. The HFRE_Set_Network_Operator_Selection_Format must be called before calling this function. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.
HFRE_Send_Network_Operator_Selection	Sends the current network operator selection to the remote hands-free device. This function may only be performed by an audio gateway that has received a request for the network operator selection and has a valid service level connection. Alternately, HFRE_Send_Terminating_Response() can be used to return an error to a request received from the remote hands-free device.
HFRE_Enable_Remote_Extended_Error_Result	Informs the remote audio gateway unit that the local hands-free device supports extended error information response. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.
HFRE_Send_Extended_Error_Result	Sends an unsolicited extended error result to the remote hands-free device. If this result is in response to a previously received request then

	HFRE_Send_Terminating_Response() should be used instead. This function may only be performed by an audio gateway that has a valid service level connection.
HFRE_Query_Subscriber_Number_Information	Retrieves the subscriber number information from the remote audio gateway unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.
HFRE_Send_Subscriber_Number_Information	Sends the current subscriber number information to the remote hands-free device. This function may only be performed by an audio gateway that has received a request for the subscriber number information and has a valid service level connection. This function should be called once for each phone number associated with the current subscriber, with the last call to this function setting the FinalEntry parameter to TRUE. Alternately, HFRE_Send_Terminating_Response() can be used to return an empty OK response (no subscriber information available) or an error to a request received from the remote hands-free device.
HFRE_Query_Response_Hold_Status	Retrieves the current response hold status from the remote audio gateway unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.
HFRE_Set_Incoming_Call_State	Sends the current incoming call state to the remote audio gateway device. This function may only be performed by a hands-free device that has a valid service level connection.
HFRE_Send_Incoming_Call_State	Sends the current incoming call state to the remote hands-free device. This function may only be performed by an audio gateway that has received a request for the response hold status and has a valid service level connection. Alternately, HFRE_Send_Terminating_Response() can be used to return an error to a request received from

	the remote hands-free device.
HFRE_Send_Terminating_Response	Sends a terminating response to the remote hands-free device. This function can be used in place of the normal response function that would be associated with a particular request when the audio gateway needs to return an empty response (OK) or an error result (ERROR, BUSY, Extended Errors, etc). This function may only be performed by an audio gateway that has received a request and has a valid service level connection.
HFRE_Enable_Arbitrary_Command_Processing	Enables the processing of arbitrary commands from the remote Hands Free device.
HFRE_Send_Arbitrary_Command	Responsible for sending arbitrary commands to remote Audio Gateway.
HFRE_Send_Arbitrary_Response	Responsible for sending an arbitrary response to a remote Hands Free device
HFRE_Get_Server_Mode	Retrieves the current Handsfree/Audio Gateway (HF/AG) Server's Server Mode for specified HF/AG Server.
HFRE_Set_Server_Mode	Changes the current HF/AG Server's Server Mode for specified HF/AG Server.

HFRE_Open_HandsFree_Server_Port

Opens a hands-free server on the specified Bluetooth SPP serial port.

Prototype:

int BTPSAPI **HFRE_Open_HandsFree_Server_Port**(unsigned int BluetoothStackID, unsigned int ServerPort, unsigned long SupportedFeaturesMask, unsigned int NumberAdditionalIndicators, char *AdditionalSupportedIndicators[], HFRE_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

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

ServerPort — The local serial port Server Number to use. This must fall in the range defined by the following constants:

SPP_PORT_NUMBER_MINIMUM
SPP_PORT_NUMBER_MAXIMUM

SupportedFeaturesMask — A bit mask that specifies the features that the hands-free unit supports. The following values are supported:

HFRE_HF_SOUND_ENHANCEMENT_SUPPORTED_BIT

HFRE_CALL_WAITING_THREE_WAY_CALLING_SUPPORTED_BIT
HFRE_CLI_SUPPORTED_BIT
HFRE_HF_VOICE_RECOGNITION_SUPPORTED_BIT
HFRE_REMOTE_VOLUME_CONTROL_SUPPORTED_BIT
HFRE_DEFAULT_HANDSFREE_BIT_MASK (no supported features)

NumberAdditionalIndicators	The number of additional indicators in the previous parameter.
AdditionalSupportedIndicators	A list of additional indicators to support.
EventCallback	Function to call when events occur on this port.
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:

A positive, non-zero, value if successful. A successful return code will be a HFRE Port ID that can be used to reference the Opened HFRE Port in ALL other functions in this module except for the HFRE_Register_Audio_Gateway_SDP_Record() function which is specific to an Audio Gateway Server NOT a Hands-Free Server

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

BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Open_Port_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.

HFRE_Open_ Audio_Gateway _Server_Port

Opens an audio gateway server on the specified Bluetooth SPP serial port.

Notes:

1. If the HFRE_AG_QUERY_INDICATOR_REQUEST_SUPPORTED_BIT bit is set in the SupportedFeaturesMask that is passed to this function then the event callback function that is also passed as a parameter to this function must be capable of handling the etHFRE_Control_Indicator_Request_Indication event and responding to it with the HFRE_Send_Control_Indicator_Request_Response() API when all of the control indicators have been updated.

Prototype:

int BTPSAPI **HFRE_Open_Audio_Gateway_Server_Port**(unsigned int BluetoothStackID, unsigned int ServerPort, unsigned long SupportedFeaturesMask, unsigned long CallHoldingSupportMask, unsigned int NumberAdditionalIndicators, HFRE_Control_Indicator_Entry_t AdditionalSupportedIndicators[], HFRE_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServerPort	Local serial port Server Number to use. This must fall in the range defined by the following constants: SPP_PORT_NUMBER_MINIMUM SPP_PORT_NUMBER_MAXIMUM
SupportedFeaturesMask	A bit mask that specifies the features that the audio gateway supports. HFRE_THREE_WAY_CALLING_SUPPORTED_BIT HFRE_AG_SOUND_ENHANCEMENT_SUPPORTED_BIT HFRE_AG_VOICE_RECOGNITION_SUPPORTED_BIT HFRE_INBAND_RINGING_SUPPORTED_BIT HFRE_VOICE_TAGS_SUPPORTED_BIT HFRE_DEFAULT_AUDIO_GATEWAY_BIT_MASK (this sets HFRE_THREE_WAY_CALLING_SUPPORTED_BIT and HFRE_INBAND_RINGING_SUPPORTED_BIT)
CallHoldingSupportMask	A bit mask that specifies the call hold and multi-party support features that the audio gateway supports. HFRE_RELEASE_ALL_HELD_CALLS HFRE_RELEASE_ALL_ACTIVE_CALLS_ACCEPT_WAITING_CALL HFRE_PLACE_ALL_ACTIVE_CALLS_ON_HOLD_ACCEPT_THE_OTHER HFRE_ADD_A_HELD_CALL_TO_CONVERSATION HFRE_CONNECT_TWO_CALLS_DISCONNECT_SUBSCRIBER
NumberAdditionalIndicators	The number of additional indicators in the previous parameter.
AdditionalSupportedIndicators	A list of additional indicators to support.
EventCallback	Function to call when events occur on this port.
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:

This function returns a positive, non-zero, value if successful. A successful return code will be a HFRE Port ID that can be used to reference the Opened HFRE Port in ALL other functions in this module except for the HFRE_Register_HandsFree_SDP_Record() function which is specific to a hands-free server NOT an audio gateway.

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

BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Open_Port_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.

HFRE_Close_Server_Port

Un-registers a HFRE Port server (which was registered by a successful call to either the HFRE_Open_HandsFree_Port() or the HFRE_Open_Audio_Gateway_Server_Port() function).

Prototype:

int BTPSAPI **HFRE_Close_Server_Port**(unsigned int BluetoothStackID,
unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The Hands-Free Port ID to close. This is the value that was returned from either HFRE_Open_HandsFree_Server_Port() or HFRE_Open_Audio_Gateway_Server_Port().

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

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.

HFRE_Open_Port_Request_Response

This function responds to requests to connect to a HF or AG Server.

Prototype:

int BTPSAPI **HFRE_Open_Port_Request_Response**(unsigned int BluetoothStackID, unsigned int HFREPortID, Boolean_t AcceptConnection)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HF/AG Port ID which must have been obtained in a call to either HFRE_Open_HandsFree_Server_Port() or the HFRE_Open_Audio_Gateway_Server_Port().
AcceptConnection	Specifies whether to accept the pending connection request (TRUE to accept).

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_PARAMETER

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

HFRE_Register_HandsFree_SDP_Record

Adds a generic hands-free service record to the SDP database.

Notes:

2. This function should only be called with the HFRE Port ID that was returned from the HFRE_Open_HandsFree_Server_Port( ) function. This function should **never** be used with the HFRE Port ID returned from the HFRE_Open_Audio_Gateway_Server_Port( ) function.
3. 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.

4. 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 **HFRE_Register_HandsFree_SDP_Record**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *ServiceName, DWord_t *SDPServiceRecordHandle)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE Port ID this command applies to. This value MUST have been obtained by calling the HFRE_Open_HandsFree_Server_Port() function.
ServiceName	Name to appear in the SDP Database for this service.
SDPServiceRecordHandle	Returned handle to the SDP Database entry that may be used to remove the entry at a later time.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

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.

HFRE_Register_Audio_Gateway_SDP_Record

Adds a generic Audio Gateway Service Record to the SDP database.

Notes:

1. This function should only be called with the HFRE Port ID that was returned from the HFRE_Open_Audio_Gateway_Server_Port( ) function. This function should **never** be used with the HFRE Port ID returned from the HFRE_Open_HandsFree_Server_Port( ) function.

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.

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 **HFRE_Register_Audio_Gateway_SDP_Record**(
unsigned int BluetoothStackID, unsigned int HFREPortID, unsigned int NetworkType, char *ServiceName, DWord_t *SDPServiceRecordHandle)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE Port ID this command applies to. This value MUST have been obtained by calling the HFRE_Open_Audio_Gateway_Server_Port() function.
NetworkType	Specifies the network type that the audio gateway is attached to or is being attached to.
ServiceName	Name to appear in the SDP Database for this service.
SDPServiceRecordHandle	Returned handle to the SDP Database entry that may be used to remove the entry at a later time.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

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.

HFRE_Un_Register_SDP_Record

Removes a generic hands-free service record or audio gateway service record from the SDP database. This MACRO simply maps to the SDP_Delete_Service_Record() function. This MACRO is provided so that the caller doesn't have to sift through the SDP API for very simplistic applications.

MACRO definition:

HFRE_Un_Register_SDP_Record(__BluetoothStackID, __HFREPortID, __SDPRecordHandle)(SDP_Delete_Service_Record(__BluetoothStackID, __SDPRecordHandle))

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE Port ID this command applies to.
SDPRecordHandle	The Hands Free SDP Record Handle that specifies the SDP Service Record Handle of the Hands Free Profile.

Return:

Possible Events:

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.

HFRE_Open_Remote_HandsFree_Port

Opens a remote hands-free port on the specified remote device.

Notes:

1. If the HFRE_AG_QUERY_INDICATOR_REQUEST_SUPPORTED_BIT bit is set in the SupportedFeaturesMask that is passed to this function then the event callback function that is also passed as a parameter to this function must be capable of handling the etHFRE_Control_Indicator_Request_Indication event and responding to it with the HFRE_Send_Control_Indicator_Request_Response() API when all of the control indicators have been updated.

Prototype:

int BTPSAPI **HFRE_Open_Remote_HandsFree_Port**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, unsigned int RemoteServerPort, unsigned long SupportedFeaturesMask, unsigned long CallHoldSupportMask, unsigned int NumberAdditionalIndicators, HFRE_Control_Indicator_Entry_t AdditionalSupportedIndicators[], HFRE_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

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

RemoteServerPort	Remote Server Channel ID to connect with. This must fall in the range defined by the following constants: SPP_PORT_NUMBER_MINIMUM SPP_PORT_NUMBER_MAXIMUM
SupportedFeaturesMask	A bit mask that specifies the features that the local audio gateway supports.
CallHoldSupportMask	A bit mask that specifies the call hold and multi-party support features that the local audio gateway supports.
NumberAdditionalIndicators	The number of additional indicators in the previous parameter.
AdditionalSupportedIndicators	A list of additional indicators to support.
EventCallback	Function to call when events occur on this port.
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:

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

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

BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Open_Port_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.

HFRE_Open_Remote_Audio_Gateway_Port

Opens a remote audio gateway port on the specified remote device.

Prototype:

```
int BTPSAPI HFRE_Open_Remote_Audio_Gateway_Port(
    unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR,
    unsigned int RemoteServerPort, unsigned long SupportedFeaturesMask,
    unsigned int NumberAdditionalIndicators, char *AdditionalSupportedIndicators[],
    HFRE_Event_Callback_t EventCallback, unsigned long CallbackParameter)
```

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Address of the Bluetooth device to connect with.
RemoteServerPort	Remote Server Channel ID to connect with. This must fall in the range defined by the following constants: SPP_PORT_NUMBER_MINIMUM SPP_PORT_NUMBER_MAXIMUM
SupportedFeaturesMask	A bit mask that specifies the features that the local hands-free unit supports.
CallHoldingSupportMask	A bit mask that specifies the call hold and multi-party support features that the local hands-free unit supports.
NumberAdditionalIndicators	The number of additional indicators in the previous parameter.
AdditionalSupportedIndicators	A list of additional indicators to support.
EventCallback	Function to call when events occur on this port.
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:

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

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

BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Port_Open_Confirmation

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

HFRE_Close_Port

Closes a HFRE port that was previously opened by any of the following mechanisms: Successful call to HFRE_Open_Remote_ HandsFree _Port() function. Successful call to HFRE_Open_Remote_Audio_Gateway_Port() function. Incoming open request (Hands-Free or Audio Gateway) which the server was opened with either the HFRE_Open_ HandsFree _Server_Port() or the HFRE_Open_Audio_Gateway_Server_Port() functions.

Prototype:

int BTPSAPI **HFRE_Close_Port**(unsigned int BluetoothStackID,
unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port to close. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

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.

HFRE_Query_Remote_Control_Indicator_Status

Querys the remote control indicator status. Only hands-free units that have a valid service level connection may perform this function.

Prototype:

int BTPSAPI **HFRE_Query_Remote_Control_Indicator_Status**(
unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Control_Indicator_Status_Confirmation

etHFRE_Close_Port_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.

HFRE_Send_Control_Indicator_Request_Response

This function is responding to a request from a remote Hands Free device for the current control indicators of the local Audio Gateway unit. This function can only be called by Audio Gateways that have received a request for the current control indicators (the event is of type etHFRE_Control_Indicator_Request_Indication).

Notes:

1. This function may only be called if the HFRE_AG_QUERY_INDICATOR_REQUEST_SUPPORTED_BIT was set in the SupportedFeaturesMask that was passed to the HFRE_Open_Audio_Gateway_Server_Port() or HFRE_Open_Remote_HandsFree_Port() API call that returned the specified HFREPortID and then only in response to a etHFRE_Control_Indicator_Request_Indication event.

Prototype:

int BTPSAPI **HFRE_Send_Control_Indicator_Request_Response**(
unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

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.

HFRE_Update_Current_Control_Indicator_Status

Updates the current control indicator status. This function checks the input parameters against the currently stored values. If they differ, the remote device maybe informed of the change. This function may only be performed by audio gateways that have a valid service level connection and event reporting activated (set via the Set Remote Event Indicator Event Notification function by the remote device) or by audio gateways that have received the etHFRE_Control_Indicator_Request_Indication event.

Prototype:

int BTPSAPI **HFRE_Update_Current_Control_Indicator_Status**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
unsigned int NumberUpdateIndicators, HFRE_Indicator_Update_t UpdateIndicators[])

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
NumberUpdateIndicators	The number of name / value pairs that are present in the list.
UpdateIndicators	A list of name / value pairs for the indicators to be updated.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Update_Current_Control_Indicator_Status_By_Name

This function updates the Current Control Indicator Status. This function may only be called by Audio Gateways.

Prototype:

int BTPSAPI **HFRE_Update_Current_Control_Indicator_Status_By_Name**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
unsigned int IndicatorValue)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HF/AG Server Port ID.
IndicatorValue	The new indicator value for the Control Indicator Status.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_PARAMETER

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

HFRE_Enable_Remote_Indicator_Event_Notification

Enables or disables the indicator event notification on the remote device. When enabled, the remote device will send unsolicited responses to update the local device of the current control indicator values. Only hands-free units that have a valid service level connection may perform this function.

Prototype:

int BTPSAPI **HFRE_Enable_Remote_Indicator_Event_Notification**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
Boolean_t EnableEventNotification)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
EnableEventNotification	Boolean flag used to activate or deactivate event notification.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Query_Remote_Call_Holding_Multiparty_Service_Support

Querys the call holding and multiparty services supported by the remote device. This function should be used by hands-free units that support three-way calling and call waiting to determine the features supported by the audio gateway. This function can only be used if a valid service level connection exists.

Prototype:

int BTPSAPI **HFRE_Query_Remote_Call_Holding_Multiparty_Service_Support**
(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED

BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Call_Hold_Multiparty_Support_Confirmation

etHFRE_Close_Port_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.

HFRE_Send_Call_Holding_Multiparty_Selection

Allows the control of multiple concurrent calls and provides means for holding calls, releasing calls, switching between two calls and adding a call to a multiparty conference. Only hands-free units that support call waiting and multiparty services and have a valid service level connection may perform this function. The selection should be one that is supported by the remote audio gateway as queried via a call to the Query Remote Call Holding Multiparty Service Support function.

Prototype:

int BTPSAPI **HFRE_Send_Call_Holding_Multiparty_Selection**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
HFRE_Call_Hold_Multiparty_Handling_Type_t CallHoldMultipartyHandling,
unsigned int Index)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
CallHoldMultipartyHandling	How to handle the currently waiting call.
Index	An optional Index value that is used by some of the CallHoldMultipartyHandling values. See the HFRE specification for more information.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Enable_Remote_Call_Waiting_Notification

Enables or disables call waiting notification on the remote audio gateway. By default, the call waiting notification is enabled in the network but disabled for notification via the service level connection. Only hands-free units having a valid service level connection may perform this function. This function may only be used to enable call waiting notification if the local hands-free unit supports call waiting and multiparty services. It may, however, be used to disable these service by all hands-free units.

Prototype:

int BTPSAPI **HFRE_Enable_Remote_Call_Waiting_Notification**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
Boolean_t EnableNotification)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
EnableNotification	Boolean flag to enable / disable this functionality.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Call_Waiting_Notification

Sends call waiting notifications to the remote device. This function may only be performed by audio gateways that have call waiting notification enabled and have a valid service level connection.

Prototype:

int BTPSAPI **HFRE_Send_Call_Waiting_Notification**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *PhoneNumber)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
PhoneNumber	The phone number. This parameter should be a pointer to a '\0' terminated string and its length **must** be between HFRE_PHONE_NUMBER_LENGTH_MINIMUM and HFRE_PHONE_NUMBER_LENGTH_MAXIMUM.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Enable_Remote_Call_Line_Identification_Notification

Enables or disabling call waiting notification on the remote audio gateway. By default the call line identification notification via the service level connection is disabled. This function may only be performed by hands-free units for which a valid service level connection exist. This function may only be used to enable call line notification if the local hands-free unit supports call line identification, it may however be used to disable this service by all hands-free units.

Prototype:

int BTPSAPI **HFRE_Enable_Remote_Call_Line_Identification_Notification**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
Boolean_t EnableNotification)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
EnableNotification	Boolean to enable / disable notification.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Call_Line_Identification_Notification

Sends call line identification notifications to the remote device. Only audio gateways that have call line identification notification enabled and have a valid service level connection may perform this function.

Prototype:

int BTPSAPI **HFRE_Send_Call_Line_Identification_Notification**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *PhoneNumber)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.

PhoneNumber	The phone number. This parameter should be a pointer to a '\0' terminated string and its length **must** be between HFRE_PHONE_NUMBER_LENGTH_MINIMUM and HFRE_PHONE_NUMBER_LENGTH_MAXIMUM.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Disable_Remote_Echo_Cancelation_Noise_Reduction

Disables echo cancelation and noise reduction on the remote device. Both the hands-free unit and the audio gateway for which a valid service level connection exists may perform this function but no audio connection exists.

Prototype:

int BTPSAPI **HFRE_Disable_Remote_Echo_Cancelation_Noise_Reduction**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Dial_Phone_Number

Dials a phone number on the remote audio gateway. This function may only be performed by hands-free units for which a valid service level connection exists.

Prototype:

int BTPSAPI **HFRE_Dial_Phone_Number**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *PhoneNumber)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
PhoneNumber	The phone number. This parameter should be a pointer to a '\0' terminated string and its length **must** be between HFRE_PHONE_NUMBER_LENGTH_MINIMUM and HFRE_PHONE_NUMBER_LENGTH_MAXIMUM.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Dial_Phone_Number_From_Memory

Dials a phone number at a memory location found on the remote audio gateway. Hands-free units for which a valid service level connection exists may only perform this function.

Prototype:

int BTPSAPI **HFRE_Dial_Phone_Number_From_Memory**(
unsigned int BluetoothStackID, unsigned int HFREPortID, unsigned int MemoryLocation)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.
MemoryLocation	Memory location where the phone number to dial resides in the remote audio gateway.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Redial_Last_Phone_Number

Redials the last number dialed on the remote audio gateway. Hands-free units for which a valid service level connection exists may only perform this function.

Prototype:

int BTPSAPI **HFRE_Redial_Last_Phone_Number**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	The HFRE port. This is the value that was returned from one of the above Open functions.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Ring_Indication

Sends ring indications to the remote hands-free unit. This function may only be performed by audio gateways for which a valid service level connection exists.

Prototype:

int BTPSAPI **HFRE_Ring_Indication**(unsigned int BluetoothStackID,
unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Answer_Incoming_Call

Sends the command to answer incoming calls on a remote audio gateway. Hands-free units for which a valid service level connection exists may only perform this function.

Prototype:

int BTPSAPI **HFRE_Answer_Incoming_Call**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Enable_Remote_InBand_Ring_Tone_Setting

Enables or disables in-band ring tone capabilities. This function may only be performed by audio gateways for which a valid service level connection exists. This function may only be used to enable in-band ring tone capabilities if the local audio gateway supports this feature.

Prototype:

int BTPSAPI **HFRE_Enable_Remote_InBand_Ring_Tone_Setting**(
unsigned int BluetoothStackID, unsigned int HFREPortID, Boolean_t EnableInBandRing)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
EnableInBandRing	Boolean to enable / disable in-band ringing.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Transmit_DTMF_Code

Transmits DTMF codes to the remote audio gateway to be sent as a DTMF Tone over an on-going call. This function may only be performed by hands-free units for which a valid service level connection exists.

Prototype:

int BTPSAPI **HFRE_Transmit_DTMF_Code**(unsigned int BluetoothStackID, unsigned int HFREPortID, char DTMFCode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
DTMFCode	The DTMF code to be transmitted. This code must be a character 0-9, *, #, or A-D.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Set_Remote_Voice_Recognition_Activation

Activates and deactivates the voice recognition which resides on the remote audio gateway when called by a hands-free unit. When called by an audio gateway, this function informs the remote hands-free unit of the current activation state of the local voice recognition function. Only local devices that were opened with the supported feature bit set for voice recognition may call this function.

Prototype:

int BTPSAPI **HFRE_Set_Remote_Voice_Recognition_Activation**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
Boolean_t VoiceRecognitionActive)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
VoiceRecognitionActive	Boolean to Activate/ Deactivate voice recognition.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Set_Remote_Speaker_Gain

Allows synchronization with and setting of the remote device's speaker gain. This function may only be performed if a valid service level connection exists. When called by a hands-free unit this function is provided as a means to inform the remote audio gateway of the current speaker gain value. When called by an audio gateway this function provides a means for the audio gateway to control the speaker gain of the remote hands-free unit.

Prototype:

int BTPSAPI **HFRE_Set_Remote_Speaker_Gain**(unsigned int BluetoothStackID, unsigned int HFREPortID, unsigned int SpeakerGain)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
SpeakerGain	The new speaker gain value. The speaker gain parameter **must** be between the values of HFRE_SPEAKER_GAIN_MINIMUM and HFRE_SPEAKER_GAIN_MAXIMUM.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Set_Remote_Microphone_Gain

Allows synchronization with and setting of the remote device's microphone gain. This function may only be performed if a valid service level connection exists. When called by a hands-free unit this function is provided as a means to inform the remote audio gateway of the current microphone gain value. When called by an audio gateway this function provides a means for the audio gateway to control the microphone gain of the remote hands-free unit.

Prototype:

int BTPSAPI **HFRE_Set_Remote_Microphone_Gain**(unsigned int BluetoothStackID, unsigned int HFREPortID, unsigned int MicrophoneGain)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
MicrophoneGain	The new microphone gain value. The microphone gain parameter **must** be between the values of HFRE_MICROPHONE_GAIN_MINIMUM and HFRE_MICROPHONE_GAIN_MAXIMUM.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Voice_Tag_Request

Retrieves a phone number to associate with a unique voice tag to be stored in memory by the local hands-free unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. The hands-free unit must also support voice recognition to be able to use this function. When this function is called, no other function may be called until a voice tag response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Voice_Tag_Request**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Voice_Tag_Request_Confirmation

etHFRE_Close_Port_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.

HFRE_Voice_Tag_Response

Transmits a phone number to be associated with a unique voice tag by the remote hands-free unit. This function may only be performed by audio gateways that have received a voice tag request indication.

Prototype:

int BTPSAPI **HFRE_Voice_Tag_Response**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *PhoneNumber)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
PhoneNumber	The phone number to be associated with the voice tag. If the Audio Gateway wishes to reject the request the PhoneNumber parameter should be set to NULL to indicate this. Otherwise this parameter should be a pointer to a '\0' terminated string and its length **must** be between HFRE_PHONE_NUMBER_LENGTH_MINIMUM and HFRE_PHONE_NUMBER_LENGTH_MAXIMUM.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INSUFFICIENT_RESOURCES
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Hang_Up_Call

Transmits a hang-up command to the remote audio gateway. Hands-free units for which a valid service level connection exists may only perform this function.

Prototype:

int BTPSAPI **HFRE_Hang_Up_Call**(unsigned int BluetoothStackID,
unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Setup_Audio_Connection

Sets up an audio connection between the local and remote device. Either an audio gateway or a hands-free unit for which a valid service level connection exists may use this function.

Prototype:

int BTPSAPI **HFRE_Setup_Audio_Connection**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Audio_Connection_Indication

etHFRE_Close_Port_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.

HFRE_Release_Audio_Connection

Releases an audio connection that was set up via a call to HFRE_Setup_Audio_Connection(). Either an audio gateway or a hands-free unit may use this function.

Prototype:

int BTPSAPI **HFRE_Release_Audio_Connection**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Audio_Disconnection_Indication

etHFRE_Close_Port_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.

HFRE_Send_Audio_Data

Provides the local entity a mechanism of sending SCO audio data to the remote entity. This function can only be called once an audio connection has been established.

Notes:

If this function returns BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE then the application must wait for the etHFRE_Audio_Transmit_Buffer_Empty_Indication event and re-transmit the selected data.

Prototype:

int BTPSAPI **HFRE_Send_Audio_Data**(unsigned int BluetoothStackID, unsigned int HFREPortID, Byte_t AudioDataLength, Byte_t *AudioData)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
AudioDataLength	Length, in bytes, of the audio data to send.
AudioData	Pointer to the audio data to send.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER
BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

Possible Events:

etHFRE_Close_Port_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.

HFRE_Query_Remote_Current_Calls_List

Retrieves the current call list from the remote audio gateway unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a current call list response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Query_Remote_Current_Calls_List**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Current_Calls_List

Sends the current calls list to the remote hands-free device. This function will NOT send the Phonebook Name entry in the HFRE_Current_Call_List_Entry_t structure. Use HFRE_Send_Current_Calls_List_With_Phonebook_Name () if this field should be sent. This function may only be performed by an audio gateway that has received a request for the current calls list and has a valid service level connection. This function should be called once for each call in the current list, with the last call setting the FinalEntry parameter to TRUE. Alternately, HFRE_Send_Terminating_Response() can be used to return an empty OK response (no current calls) or an error to a request received from the remote hands-free device.

Prototype:

int BTPSAPI **HFRE_Send_Current_Calls_List**(unsigned int BluetoothStackID, unsigned int HFREPortID, HFRE_Current_Call_List_Entry_t *CurrentCallListEntry, Boolean_t FinalEntry)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
CurrentCallListEntry	The information for a single current call in the call list.
FinalEntry	Indicates if this entry will be the last entry sent in the call list.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Current_Calls_List_With_Phonebook_Name

Sends the current calls list to the remote hands-free device. This function also optionally sends the phonebook name with each entry, which is NOT included in the specification. This function may only be performed by an audio gateway that has received a request for the current calls list and has a valid service level connection. This function should be called once for each call in the current list, with the last call setting the FinalEntry parameter to TRUE. Alternately, HFRE_Send_Terminating_Response() can be used to return an empty OK response (no current calls) or an error to a request received from the remote hands-free device.

Prototype:

int BTPSAPI **HFRE_Send_Current_Calls_List_With_Phonebook_Name**(unsigned int BluetoothStackID, unsigned int HFREPortID, HFRE_Current_Call_List_Entry_t *CurrentCallListEntry, Boolean_t FinalEntry)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
CurrentCallListEntry	The information for a single current call in the call list.
FinalEntry	Indicates if this entry will be the last entry sent in the call list.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Set_Network_Operator_Selection_Format

Sets the network operator selection format for remote audio gateway unit. This format is a specification defined value that is not variable. This function must be called before calling HFRE_Query_Remote_Network_Operator_Selection(). This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Set_Network_Operator_Selection_Format**(
unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Query_Remote_Network_Operator_Selection

Retrieves the remote network operator selection from the remote audio gateway unit. The HFRE_Set_Network_Operator_Selection_Format must be called before calling this function. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Query_Remote_Network_Operator_Selection**(
unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Network_Operator_Selection

Sends the current network operator selection to the remote hands-free device. This function may only be performed by an audio gateway that has received a request for the network operator selection and has a valid service level connection. Alternately, HFRE_Send_Terminating_Response() can be used to return an error to a request received from the remote hands-free device.

Prototype:

int BTPSAPI **HFRE_Send_Network_Operator_Selection**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
unsigned int NetworkMode, char *NetworkOperator)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
NetworkMode	The current network mode value as defined by the HFRE specification. This value is typically ignored by HFRE device (use HFRE_NETWORK_MODE_AUTOMATIC as default). The following defines are provided for convenience: HFRE_NETWORK_MODE_AUTOMATIC HFRE_NETWORK_MODE_MANUAL HFRE_NETWORK_MODE_DEREGISTER HFRE_NETWORK_MODE_SETONLY HFRE_NETWORK_MODE_MANUAL_AUTO
NetworkOperator	The name of the currently selected network operator.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Enable_Remote_Extended_Error_Result

Informs the remote audio gateway unit that the local hands-free device supports extended error information response. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Enable_Remote_Extended_Error_Result**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
Boolean_t EnableExtendedErrorResults)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
EnableExtendedErrorResults	A Boolean value that indicates if extended error information is supported by the local device.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Extended_Error_Result

Sends an unsolicited extended error result to the remote hands-free device. If this result is in response to a previously received request then HFRE_Send_Terminating_Response() should be used instead. This function may only be performed by an audio gateway that has a valid service level connection.

Prototype:

int BTPSAPI **HFRE_Send_Extended_Error_Result**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
unsigned int ResultCode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
ResultCode	The result code value to send to the remote device. These values are defined in the HFRE specification.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Query_Subscriber_Number_Information

Retrieves the subscriber number information from the remote audio gateway unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Query_Subscriber_Number_Information**(
unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Subscriber_Number_Information

Sends the current subscriber number information to the remote hands-free device. This function may only be performed by an audio gateway that has received a request for the subscriber number information and has a valid service level connection. This function should be called once for each phone number associated with the current subscriber, with the last call to this function setting the FinalEntry parameter to TRUE. Alternately, HFRE_Send_Terminating_Response() can be used to return an empty OK response (no subscriber information available) or an error to a request received from the remote hands-free device.

Prototype:

int BTPSAPI **HFRE_Query_Subscriber_Number_Information**(
unsigned int BluetoothStackID, unsigned int HFREPortID, char *PhoneNumber,
unsigned int ServiceType, unsigned int NumberFormat, Boolean_t FinalEntry)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
PhoneNumber	The phone number string associated with the current subscriber number information.
ServiceType	The service type value for the current subscriber number information. The meaning of this value is defined in the HFRE specification. Two defines are provided for convenience: HFRE_SERVICE_TYPE_VOICE HFRE_SERVICE_TYPE_FAX
NumberFormat	The number format value for the current subscriber number information. The meaning of this value is defined in the HFRE specification. Two defines are provided for convenience: HFRE_DEFAULT_NUMBER_FORMAT HFRE_DEFAULT_NUMBER_FORMAT_INTERNATIONAL
FinalEntry	This value indicates if this is the last entry with subscriber number information in response to the original request.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Query_Response_Hold_Status

Retrieves the current response hold status from the remote audio gateway unit. This function may only be performed by a hands-free unit for which a valid Service Level Connection Exists. When this function is called, no other function may be called until a response is received from the remote audio gateway.

Prototype:

int BTPSAPI **HFRE_Query_Response_Hold_Status**(
unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Set_Incoming_Call_State

Sends the current incoming call state to the remote audio gateway device. This function may only be performed by a hands-free device that has a valid service level connection.

Prototype:

int BTPSAPI **HFRE_Set_Incoming_Call_State**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
HFRE_Call_State_t CallState)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
CallState	Specifies the callstate of the current incoming call. The following values are supported: csHold csAccept csReject csNone

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Incoming_Call_State

Sends the current incoming call state to the remote hands-free device. This function may only be performed by an audio gateway that has received a request for the response hold status and has a valid service level connection. Alternately, HFRE_Send_Terminating_Response() can be used to return an error to a request received from the remote hands-free device.

Prototype:

int BTPSAPI **HFRE_Send_Incoming_Call_State**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
HFRE_Call_State_t CallState)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
CallState	Specifies the callstate of the current incoming call. The following values are supported: csHold csAccept csReject csNone

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Send_Terminating_Response

Sends a terminating response to the remote hands-free device. This function can be used in place of the normal response function that would be associated with a particular request when the audio gateway needs to return an empty response (OK) or an error result (ERROR, BUSY, Extended Errors, etc). This function may only be performed by an audio gateway that has received a request and has a valid service level connection.

Prototype:

int BTPSAPI **HFRE_Send_Terminating_Response**(
unsigned int BluetoothStackID, unsigned int HFREPortID,
HFRE_Extended_Result_t ResultType, unsigned int ResultValue)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

ResultType	The type of result to return in this terminating response. The following values are supported: erOK erError erNoCarrier erBusy erNoAnswer erDelayed erBlacklisted erResultCode
ResultValue	The actual value used for an extended error result. This value is only used for a ResultType of erResultCode.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Close_Port_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.

HFRE_Enable_Arbitrary_Command_Processing

This function enables the processing of arbitrary commands from the remote Hands Free device. If arbitrary command processing is not enable, the AG will silently respond to any arbitrary commands with an error response. Once enabled the caller is responsible for responsible for responding to ALL arbitrary command indications which are dispatched by etHFRE_Arbitrary_Command_Indication. If the arbitrary command is not supported the caller should simply respond with HFRE_Send_Terminating_Response(). Once Arbitrary Command processing is enabled for an Audio Gateway (AG) it cannot be disabled. The default values is disabled.If enabled by a call to this function the caller is guaranteed that a etHFRE_Arbritrary_Command_Indication will not be dispatched before a Service Level Indication is present. This function is not applicable to Hands Free devices.

Prototype:

Int BTPSAPI **HFRE_Enable_Arbitrary_Command_Processing**(unsigned int BluetoothStackID, unsigned int HFREPortID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Arbitrary_Command_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.

HFRE_Send_Arbitrary_Command

This function is responsible for sending an arbitrary command to a remote Audio Gateway. This function may only be performed by a Hands-Free with a valid Service Level Connection.

Notes:

All calls to this function must pass an arbitrary command string that ends end with a carriage return (0x0D). There is a restriction for the first call to a new arbitrary command in that it must start with the "AT" string (minus the "'s). Subsequent calls (for the same arbitrary command do not have to start with this string (but still must end with the carriage return). A call to this function is considered subsequent if a command result event (etHFRE_Command_Result) has not been received by the remote device. This allows multiple lines (or fragmented commands) to be sent. This is useful when sending SMS messages over the Hands Free API utilizing arbitrary commands (as the remote device will respond with a command prompt-like shell ">" for each line of the message to be sent.

Prototype:

int BTPSAPI **HFRE_Send_Arbitrary_Command**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *ArbitraryCommand)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
ArbitraryCommand	Null terminated ASCII string that represents the arbitrary command to send. See notes above regarding special requirements for this string.

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Arbitrary_Command_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.

HFRE_Send_Arbitrary_Response

This function is responsible for sending an arbitrary response to the remote Hands Free Device (i.e. non Bluetooth Hands Free Profile Response). May only be an Audio Gateway with a valid Service Level connection.

Notes:

The string that is passed to this function must start with a carriage return/line feed pair (0x0D/0x0A). The string does not have to end with this sequence (although it will need it to be recognized by the remote Hands Free device), however, it must start with it. This allows the Audio Gateway to send command prompt-like shell responses (e.g. ">") by simply sending the carriage return/line feed pair followed by the shell prompt. Each new line can then be sent beginning with a carriage return/line feed pair.

Prototype:

int BTPSAPI **HFRE_Send_Arbitrary_Response**(unsigned int BluetoothStackID, unsigned int HFREPortID, char *ArbitraryCommand)

Parameters:

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

HFREPortID	HFRE Port ID for which the connection has been established.
ArbitraryResponse	NULL terminated ASCII string that represents the arbitrary response to send. This string MUST begin with a carriage return/line feed (0x0D 0x0A or "\r\n").

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED
BTHFRE_ERROR_INVALID_OPERATION
BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Possible Events:

etHFRE_Arbitrary_Response_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.

HFRE_Get_Server_Mode

This function retrieves the current HF/AG Server Mode for a specified HF/AG Gateway server. The default Server Mode is HFRE_SERVER_MODE_AUTOMATIC_ACCEPT_CONNECTION. The function is used for HF/AG Servers which use Bluetooth Security Mode 2.

Prototype:

int BTPSAPI **HFRE_Get_Server_Mode**(unsigned int BluetoothStackID, unsigned int HFREPortID, unsigned long *ServerModeMask)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
ServerModeMask	Pointer to a Server Mode variable which will receive the current Server Mode. Possible return values are: HFRE_SERVER_MODE_AUTOMATIC_ACCEPT_CONNECTION HFRE_SERVER_MODE_MANUAL_ACCEPT_CONNECTION

Return:

Zero if successful.

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

BTHFRE_ERROR_NOT_INITIALIZED

BTHFRE_ERROR_INVALID_BLUETOOTH_STACK_ID
BTHFRE_ERROR_INVALID_PARAMETER

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

HFRE_Set_Server_Mode

This function changes the HF/AG Server Mode for a specified HF/AG Gateway server. The default Server Mode is HFRE_SERVER_MODE_AUTOMATIC_ACCEPT_CONNECTION. The function is used for HF/AG Servers which use Bluetooth Security Mode 2.

Prototype:

int BTPSAPI **HFRE_Set_Server_Mode**(unsigned int BluetoothStackID, unsigned int HFREPortID, unsigned long ServerModeMask)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
HFREPortID	HFRE Port ID for which the connection has been established.
ServerModeMask	Server Mode variable that holds the mode that the HF/AG Server Mode will be changed to. The possible values are: HFRE_SERVER_MODE_AUTOMATIC_ACCEPT_CONNECTION HFRE_SERVER_MODE_MANUAL_ACCEPT_CONNECTION

Return:

Zero if successful.

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

Notes:

1. The BluetoothStackID parameter is not included in versions of Bluetopia that have been optimized to only control a single Bluetooth device, such as some embedded versions of Bluetopia. Please refer to the appropriate header file to determine if this parameter is part of the function call or not.

2.2 Hands-Free Profile Event Callback Prototypes

The event callback functions mentioned in the hands-free profile Open commands all accept the callback function described by the following prototype.

HFRE_Event_Callback_t

Prototype of callback function passed in one of the HFRE Open commands.

Prototype:

void (BTPSAPI ***HFRE_Event_Callback_t**)(unsigned int BluetoothStackID, HFRE_Event_Data_t *HFRE_Event_Data, unsigned long CallbackParameter)

Parameters:

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

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

```
typedef struct
{
  HFRE_Event_Type_t      Event_Data_Type;
  Word_t                 Event_Data_Size;
  union
  {
    HFRE_Open_Port_Request_Indication_Data_t
          *HFRE_Open_Port_Request_Indication_Data;
    HFRE_Open_Port_Indication_Data_t
          *HFRE_Open_Port_Indication_Data;
    HFRE_Open_Port_Confirmation_Data_t
          *HFRE_Open_Port_Confirmation_Data;
    HFRE_Open_Service_Level_Connection_Indication_Data_t
          *HFRE_Open_Service_Level_Connection_Indication_Data;
    HFRE_Control_Indicator_Status_Indication_Data_t
          *HFRE_Control_Indicator_Status_Indication_Data;
    HFRE_Control_Indicator_Status_Confirmation_Data_t
          *HFRE_Control_Indicator_Status_Confirmation_Data_t;
    HFRE_Call_Hold_Multiparty_Support_Confirmation_Data_t
          *HFRE_Call_Hold_Multiparty_Support_Confirmation_Data;
    HFRE_Call_Hold_Multiparty_Selection_Indication_Data_t
          *HFRE_Call_Hold_Multiparty_Selection_Indication_Data;
    HFRE_Call_Waiting_Notification_Activation_Indication_Data_t
          *HFRE_Call_Waiting_Notification_Activation_Indication_Data;
    HFRE_Call_Waiting_Notification_Indication_Data_t
          *HFRE_Call_Waiting_Notification_Indication_Data;
    HFRE_Call_Line_Identification_Notification_Activation_Indication_Data_t
          *HFRE_Call_Line_Identification_Notification_Activation_Indication_Data;
    HFRE_Call_Line_Identification_Notification_Indication_Data_t
          *HFRE_Call_Line_Identification_Notification_Indication_Data;
    HFRE_Disable_Sound_Enhancement_Indication_Data_t
          *HFRE_Disable_Sound_Enhancement_Indication_Data;
    HFRE_Dial_Phone_Number_Indication_Data_t
          *HFRE_Dial_Phone_Number_Indication_Data;
    HFRE_Dial_Phone_Number_From_Memory_Indication_Data_t
          *HFRE_Dial_Phone_Number_From_Memory_Indication_Data;
```

HFRE_ReDial_Last_Phone_Number_Indication_Data_t
*HFRE_ReDial_Last_Phone_Number_Indication_Data;
HFRE_Ring_Indication_Data_t
*HFRE_Ring_Indication_Data;
HFRE_Answer_Call_Indication_Data_t
*HFRE_Answer_Call_Indication_Data;
HFRE_InBand_Ring_Tone_Setting_Indication_Data_t
*HFRE_InBand_Ring_Tone_Setting_Indication_Data;
HFRE_Generate_DTMF_Tone_Indication_Data_t
*HFRE_Generate_DTMF_Tone_Indication_Data;
HFRE_Voice_Recognition_Notification_Indication_Data_t
*HFRE_Voice_Recognition_Notification_Indication_Data;
HFRE_Speaker_Gain_Indication_Data_t
*HFRE_Speaker_Gain_Indication_Data;
HFRE_Microphone_Gain_Indication_Data_t
*HFRE_Microphone_Gain_Indication_Data;
HFRE_Voice_Tag_Request_Indication_Data_t
*HFRE_Voice_Tag_Request_Indication_Data;
HFRE_Voice_Tag_Request_Confirmation_Data_t
*HFRE_Voice_Tag_Request_Confirmation_Data;
HFRE_Hang_Up_Indication_Data_t
*HFRE_Hang_Up_Indication_Data;
HFRE_Audio_Connection_Indication_Data_t
*HFRE_Audio_Connection_Indication_Data;
HFRE_Audio_Disconnection_Indication_Data_t
*HFRE_Audio_Disconnection_Indication_Data;
HFRE_Audio_Data_Indication_Data_t
*HFRE_Audio_Data_Indication_Data;
HFRE_Close_Port_Indication_Data_t
*HFRE_Close_Port_Indication_Data;
HFRE_Current_Calls_List_Indication_Data_t
*HFRE_Current_Calls_List_Indication_Data;
HFRE_Current_Calls_List_Confirmation_Data_t
*HFRE_Current_Calls_List_Confirmation_Data;
HFRE_Network_Operator_Selection_Format_Indication_Data_t
*HFRE_Network_Operator_Selection_Format_Indication_Data;
HFRE_Network_Operator_Selection_Indication_Data_t
*HFRE_Network_Operator_Selection_Indication_Data;
HFRE_Network_Operator_Selection_Confirmation_Data_t
*HFRE_Network_Operator_Selection_Confirmation_Data;
HFRE_Extended_Error_Result_Activation_Indication_Data_t
*HFRE_Extended_Error_Result_Activation_Indication_Data;
HFRE_Subscriber_Number_Information_Indication_Data_t
*HFRE_Subscriber_Number_Information_Indication_Data;
HFRE_Subscriber_Number_Information_Confirmation_Data_t
*HFRE_Subscriber_Number_Confirmation_Indication_Data;
HFRE_Response_Hold_Status_Indication_Data_t
*HFRE_Response_Hold_Status_Indication_Data;
HFRE_Response_Hold_Status_Confirmation_Data_t
*HFRE_Response_Hold_Status_Confirmation_Data;

```
      HFRE_Incoming_Call_State_Indication_Data_t
            *HFRE_Incoming_Call_State_Indication_Data;
      HFRE_Incoming_Call_State_Confirmation_Data_t
            *HFRE_Incoming_Call_State_Confirmation_Data;
      HFRE_Command_Result_Data_t
            *HFRE_Command_Result_Data;
      HFRE_Arbitrary_Command_Indication_Data_t
            *HFRE_Arbitrary_Command_Indication_Data;
      HFRE_Arbitrary_Response_Indication_Data_t
            *HFRE_Arbitrary_Response_Indication_Data;
      HFRE_Control_Indicator_Request_Indication_Data_t
            *HFRE_Control_Indicator_Request_Indication_Data;
   } Event_Data;
} HFRE_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 Hands-Free Profile Events

The possible Hands-Free Profile events from the Bluetooth stack are listed in the table below and are described in the text that follows:

Event	Description
etHFRE_Open_Port_Request_Indication	Dispatched when a Remote Client requests a connection to a Local Server.
etHFRE_Open_Port_Indication	Dispatched when a client connects to a registered server.
etHFRE_Open_Port_Confirmation	Dispatched when a client receives a Connection Response from a remote server to which the client had previously attempted to connect.
etHFRE_Open_Service_Level_Connection_Indication	Indicates that a service level connection has been opened.
etHFRE_Close_Port_Indication	Indicates that a port has been closed (unregistered).

etHFRE_Control_Indicator_Status_Indication	Indicates that remote device's control indicators have changed.
etHFRE_Control_Indicator_Status_Confirmation	Response to query of current indicator status on the remote device.
etHFRE_Call_Hold_Multiparty_Support_Confirmation	Response to query of supported features on remote device.
etHFRE_Call_Hold_Multiparty_Selection_Indication	Indication to remote device of call hold multiparty selection.
etHFRE_Call_Waiting_Notification_Activation_Indication	Request to remote device to send notification of call waiting.
etHFRE_Call_Waiting_Notification_Indication	Indication of call waiting from remote device.
etHFRE_Call_Line_Identification_Notification_Activation_Indication	Request to remote device to send notification of call line identification.
etHFRE_Call_Line_Identification_Notification_Indication	Indication of call line identification from remote device.
etHFRE_Disable_Sound_Enhancement_Indication	Request from remote device to audio gateway to disable sound enhancement.
etHFRE_Dial_Phone_Number_Indication	Request from remote device to audio gateway to dial a phone number.
etHFRE_Dial_Phone_Number_From_Memory_Indication	Request from remote device to audio gateway to dial a phone number from memory.
etHFRE_ReDial_Last_Phone_Number_Indication	Request from remote device to audio gateway to dial the last phone number dialed.
etHFRE_Ring_Indication	Indication of ringing from audio gateway to remote device.
etHFRE_Generate_DTMF_Tone_Indication	Request from remote device to audio gateway to generate DTMF tones.
etHFRE_Answer_Call_Indication	Request to answer incoming call from remote device to audio gateway.
etHFRE_InBand_Ring_Tone_Setting_Indication	Dispatched to a local Hands-Free when the remote Audio Gateway wants to change the In-Band Ring Tone Setting during an ongoing Service Level Connection.
etHFRE_Voice_Recognition_Notification_Indication	Request from remote device to activate voice recognition on an audio gateway OR indication that voice recognition has been activated from audio gateway to remote device.

etHFRE_Speaker_Gain_Indication	Request to remote device or audio gateway to change the speaker gain.
etHFRE_Microphone_Gain_Indication	Request to remote device or audio gateway to change the microphone gain.
etHFRE_Voice_Tag_Request_Indication	Request for voice tag from remote device to audio gateway.
etHFRE_Voice_Tag_Request_Confirmation	Response to a voice tag request from audio gateway to remote device.
etHFRE_Hang_Up_Indication	Request to hang up call from remote device to audio gateway.
etHFRE_Audio_Connection_Indication	Indication that an audio connection has been established to either audio gateway or remote device.
etHFRE_Audio_Disconnection_Indication	Indication that an audio connection has been disconnected to either audio gateway or remote device.
etHFRE_Audio_Data_Indication	Audio data indication to either audio gateway or remote device.
etHFRE_Audio_Transmit_Buffer_Empty_Indication	Indicates that the audio data buffer for the specified device has space for at least 1 more packet.
etHFRE_Current_Calls_List_Indication	Request from the remote hands-free device to retreive the current call list from the audio gateway.
etHFRE_Current_Calls_List_Confirmation	Response from the remote audio gateway to the local hands free device. The confirmation contains information about a single call list entry. The device can detect that no further confirmations are expected once it receives the etHFRE_Command_Result event indicating a successful terminating response (OK).
etHFRE_Network_Operator_Selection_Format_Indication	Indicates that the remote hands-free device has set the network operator selection format for the local audio gateway. In most cases no action is required other than returning a successful result.
etHFRE_Network_Operator_Selection_Indication	Request from the remote hands-free device to retreive the current operator selection information for the local audio gateway.
etHFRE_Network_Operator_Selection_Confirmation	Response from the remote audio gateway containing the request operator selection

	information sent to the local hands-free device.
etHFRE_Extended_Error_Result_Activation_Indication	Request from a remote hands-free device to indicate that it supports the reception of extended error information. Following this event the audio gateway can return extended error information to the remote hands-free device.
etHFRE_Subscriber_Number_Information_Indication	Request from a remote hands-free device to retrieve the current subscriber number information from the local audio gateway.
etHFRE_Subscriber_Number_Information_Confirmation	Response from the remote audio gateway containing the request subscriber number information. Each event contains a single set of subscriber information. The device can detect that no further confirmations are expected once it receives the etHFRE_Command_Result event indicating a successful terminating response (OK).
etHFRE_Response_Hold_Status_Indication	Request from a remote hands-free device to retrieve the current response and hold status of the local audio gateway.
etHFRE_Response_Hold_Status_Confirmation	Response from the remote hands-free device containing the current response and hold state of the remote audio gateway.
etHFRE_Incoming_Call_State_Indication	This event is received either when a local audio gateway receives a command to set the current call state OR by a local hands-free device to indicate the remote audio gateway
etHFRE_Incoming_Call_State_Confirmation	Response from a remote audio gateway containing its current response and hold status.
etHFRE_Command_Result	Indication to the local hands-free device that the remote audio gateway has sent a terminating response indicating either success (OK) or one of the possible failure/error types. This event is also generated when the remote audio gateway generates an unsoliticed result code to indicate an unexpected error.
etHFRE_Arbitrary_Command_Indication	Dispatched to a local Audio Gateway unit when the remote Hands Free Device issues an arbitrary command.
etHFRE_Arbitrary_Response_Indication	Dispatched to a local Hands Free unit when the remote Audo Gateway issues an arbitrary

	response.
etHFRE_Control_Indicator_Request_Indication	Dispatched to a local Audio Gateway unit when the remote Hands Free unit has queried the current control indicators and the Audio Gateway has registered for these events.

etHFRE_Open_Port_Request_Indication

Dispatched when a Remote Client Requests a Connection to a Local Server.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  BD_ADDR_t    BD_ADDR;
} HFRE_Open_Port_Request_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
BD_ADDR	Address of the Bluetooth Device making the request.

etHFRE_Open_Port_Indication

Dispatched when a client connects to a registered server.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  BD_ADDR_t    BD_ADDR;
} HFRE_Open_Port_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
BD_ADDR	Address of the Bluetooth Device making the request.

etHFRE_Open_Port_Confirmation

Dispatched when a client receives a connection response from a remote server to which the client had previously attempted to connect.

Return Structure:

```
typedef struct
{
  unsigned int   HFREPortID;
  unsigned int   PortOpenStatus;
} HFRE_Open_Port_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
PortOpenStatus	One of the following possible status values: HFRE _OPEN_PORT_STATUS_SUCCESS HFRE _OPEN_PORT_STATUS_CONNECTION_TIMEOUT HFRE_OPEN_PORT_STATUS_CONNECTION_REFUSED HFRE_OPEN_PORT_STATUS_AUDIO_CONNECTION_ERROR HFRE _OPEN_PORT_STATUS_UNKNOWN_ERROR

etHFRE_Open_Service_Level_Connection_Indication

Indicates that a service level connection has been opened.

Notes:

1. The Remote Supported Features Valid, Remote Supported Features, and Remote Call Hold Multiparty Support members are only used when the remote device supported the "AT+BRSF" Command as documented in the adopted Version 1.0 Hands-Free Profile. If the remote device operates using an earlier version of Hands-Free Profile these values will be set to a state indicating that they are not used and this information must be obtained by other means by the application (SDP Record, Query_xxx_() function).
2. The Remote Call Hold Multiparty Support member will only be valid if the local and remote device both have the "Three-way Calling Support" bit set in their supported features. See note above for additional requirements.
3. The Remote Call Hold Multipary Support member will always be set to HFRE_CALL_HOLD_MULTIPARTY_SUPPORTED_FEATURES_ERROR in the case when this indication is received by an Audio Gateway as Hands-Free units have no Call Hold Multiparty Supported Features to query.

Return Structure:

```
typedef struct
{
  unsigned int      HFREPortID;
  Boolean_t         RemoteSupportedFeaturesValid;
  unsigned long     RemoteSupportedFeatures;
  unsigned long     RemoteCallHoldMultipartrySupport;
} HFRE_Open_Service_Level_Connection_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

RemoteSupportedFeaturesValid	Specifies if the Remote Supported Features member is valid.
RemoteSupportedFeatures	Specifies the Supported Features of the Remote Device.
RemoteCallHoldMultiparySupport	Specifies the Call Hold and Multipary Support of the of the Remote Device.

etHFRE_Close_Port_Indication

Indicates that a port has been closed (unregistered).

Return Structure:

```
typedef struct
{
  unsigned int   HFREPortID;
  unsigned int   PortCloseStatus;
} HFRE_Close_Port_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
PortCloseStatus	One of the following possible status values: HFRE _CLOSE_PORT_STATUS_SUCCESS HFRE_CLOSE_PORT_STATUS_AUDIO_CONNECTION_ERROR HFRE _CLOSE_PORT_STATUS_CONNECTION_TIMEOUT HFRE _CLOSE_PORT_STATUS_UNKNOWN_ERROR

etHFRE_Control_Indicator_Status_Indication

Indicates that remote device's control indicators have changed.

Return Structure:

```
typedef struct
{
  unsigned int                  HFREPortID;
  HFRE_Control_Indicator_Entry_t  HFREControlIndicatorEntry;
} HFRE_Control_Indicator_Status_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
HFREControlIndicatorEntry	The control indicator that has changed.

etHFRE_Control_Indicator_Status_Confirmation

Response to query of current indicator status on the remote device.

Return Structure:

```
typedef struct
{
  unsigned int                      HFREPortID;
  HFRE_Control_Indicator_Entry_t    HFREControlIndicatorEntry;
} HFRE_Control_Indicator_Status_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
HFREControlIndicatorEntry	The current control indicator status.

etHFRE_Call_Hold_Multiparty_Support_Confirmation

Response to query of supported features on remote device.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  unsigned long    CallHoldSupportMask;
} HFRE_Call_Hold_Multiparty_Support_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
CallHoldSupportMask	One of the following values: HFRE_RELEASE_ALL_HELD_CALLS HFRE_RELEASE_ALL_ACTIVE_CALLS_ACCEPT_WAITING_CALL HFRE_PLACE_ALL_ACTIVE_CALLS_ON_HOLD_ACCEPT_THE_OTHER HFRE_ADD_A_HELD_CALL_TO_CONVERSATION HFRE_CONNECT_TWO_CALLS_DISCONNECT_SUBSCRIBER

etHFRE_Call_Hold_Multiparty_Selection_Indication

Indication to remote device of call hold multiparty selection.

Return Structure:

```
typedef struct
{
  unsigned int                                HFREPortID;
  HFRE_Call_Hold_Multiparty_Handling_Type_t    CallHoldMultipartyHandling;
} HFRE_Call_Hold_Multiparty_Selection_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
CallHoldMultipartyHandling	Specifies how to handle call holding.

etHFRE_ Call_Waiting_Notification_Activation_Indication

Request to remote device to send notification of call waiting.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  Boolean_t        Enabled;
} HFRE_Call_Waiting_Notification_Activation_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
Enabled	Enable / disable call waiting notification.

etHFRE_ Call_Waiting_Notification _Indication

Indication of call waiting from remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  char            *PhoneNumber;
} HFRE_Call_Waiting_Notification_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
PhoneNumber	Phone number of the waiting call.

etHFRE_ Call_Line_Identification_Notification_Activation_Indication

Request to remote device to send notification of call line identification.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  Boolean_t        Enabled;
} HFRE_Call_Line_Identification_Notification_Activation_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
Enabled	Enable / disable call line identification.

etHFRE_ Call_Line_Identification_Notification_ Indication

Indication of call line identification from remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  char            *PhoneNumber;
} HFRE_Call_Line_Identification_Notification_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
PhoneNumber	The phone number of the incoming call.

etHFRE_ Disable_Sound_Enhancement_Indication

Request from remote device to audio gateway to disable sound enhancement.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
} HFRE_Disable_Sound_Enhancement_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.

etHFRE_ Dial_Phone_Number_Indication

Request from remote device to audio gateway to dial a phone number.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  char             *PhoneNumber;
} HFRE_Dial_Phone_Number_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
PhoneNumber	The phone number to dial.

etHFRE_ Dial_Phone_Number_From_Memory_Indication

Request from remote device to audio gateway to dial a phone number from memory.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  unsigned int    MemoryLocation;
} HFRE_Dial_Phone_Number_From_Memory_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
MemoryLocation	Memory location where the phone number to dial is stored.

etHFRE_ ReDial_Last_Phone_Number_Indication

Request from remote device to audio gateway to dial the last phone number dialed.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
} HFRE_ReDial_Last_Phone_Number_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.

etHFRE_Ring_Indication

Indication of ringing from audio gateway to remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
} HFRE_Ring_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.

etHFRE_ Generate_DTMF_Tone_Indication

Request from remote device to audio gateway to generate DTMF tones.

Return Structure:

```
typedef struct .
{
  unsigned int   HFREPortID;
  char           DTMFCode;
} HFRE_Generate_DTMF_Tone_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
DTMFCode	DTMF digit to generate.

etHFRE_ Answer_Call_Indication

Request to answer incoming call from remote device to audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
} HFRE_Answer_Call_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.

etHFRE_ InBand_Ring_Tone_Setting_Indication

Dispatched to a local Hands-Free when the remote Audio Gateway wants to change the In-Band Ring Tone Setting during an ongoing Service Level Connection.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  Boolean_t       Enabled;
} HFRE_InBand_Ring_Tone_Setting_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
Enabled	Enable / disable in-band ring tones.

etHFRE_ Voice_Recognition_Notification_Indication

Request from remote device to activate voice recognition on an audio gateway OR indication that voice recognition has been activated from audio gateway to remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  Boolean_t       VoiceRecognitionActive;
} HFRE_Voice_Recognition_Notification_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
VoiceRecognitionActive	Current Voice Recognition State.

etHFRE_ Speaker_Gain_Indication

Request to remote device or audio gateway to change the speaker gain.

Return Structure:

```
typedef struct
{
  unsigned int   HFREPortID;
  unsigned int   SpeakerGain;
} HFRE_Speaker_Gain_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
SpeakerGain	The new speaker gain value.

etHFRE_ Microphone_Gain_Indication

Request to remote device or audio gateway to change the microphone gain.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  unsigned int    MicrophoneGain;
} HFRE_Microphone_Gain_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
MicrophoneGain	The new microphone gain value.

etHFRE_ Voice_Tag_Request_Indication

Request for voice tag from remote device to audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int   HFREPortID;
} HFRE_Voice_Tag_Request_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.

etHFRE_ Voice_Tag_ Request_Confirmation

Response to a voice tag request from audio gateway to remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  char            *PhoneNumber;
} HFRE_Voice_Tag_Request_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.
PhoneNumber	Phone number associated with the voice tag.

etHFRE_ Hang_Up_Indication

Request to hang up call from remote device to audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
} HFRE_Hang_Up_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE server connection.

etHFRE_ Audio_Connection_Indication

Indication that an audio connection has been established to either audio gateway or remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
  unsigned int    AudioConnectionOpenStatus;
} HFRE_Audio_Connection_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
AudioConnectionOpenStatus	One of the following possible status values: HFRE_AUDIO_CONNECTION_STATUS_SUCCESS HFRE_AUDIO_CONNECTION_STATUS_UNKNOWN_ERROR

etHFRE_ Audio_Disconnection_Indication

Indication that an audio connection has been disconnected to either audio gateway or remote device.

Return Structure:

```
typedef struct
{
  unsigned int    HFREPortID;
} HFRE_Audio_Disconnection_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

etHFRE_ Audio_Data _Indication

Audio data indication to either audio gateway or remote device.

Return Structure:

```
typedef struct
{
  unsigned int     HFREPortID;
  Byte_t           AudioDataLength;
  Byte_t          *AudioData;
  Word_t           PacketStatus
} HFRE_Audio_Data_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
AudioDataLength	Length of the audio data.
AudioData	Pointer to the audio data.
PacketStatus	The status (as reported by the baseband) of the packet. Valid valid values are one of the following: HCI_SCO_FLAGS_PACKET_STATUS_MASK_CORRECTLY_RECEIVED_DATA HCI_SCO_FLAGS_PACKET_STATUS_MASK_POSSIBLY_INVALID_DATA HCI_SCO_FLAGS_PACKET_STATUS_MASK_NO_DATA_RECEIVED HCI_SCO_FLAGS_PACKET_STATUS_MASK_DATA_PARTIALLY_LOST

etHFRE_Audio_Transmit_Buffer_Empty_Indication

Indicates that audio transmit buffer for the specified device has space for at least 1 more packet.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
} HFRE_Audio_Transmit_Buffer_Empty_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

etHFRE_Current_Calls_List_Indication

Request from the remote hands-free device to retreive the current call list from the audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int   HFREPortID;
} HFRE_Current_Calls_List_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

etHFRE_Current_Calls_List_Confirmation

Response from the remote audio gateway to the local hands free device. The confirmation contains information about a single call list entry. The device can detect that no further confirmations are expected once it receives the etHFRE_Command_Result event indicating a successful terminating response (OK).

Return Structure:

```
typedef struct
{
  unsigned int                        HFREPortID;
  HFRE_Current_Call_List_Entry_t HFRECurrentCallListEntry;
} HFRE_Current_Calls_List_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
HFRECurrentCallListEntry	Structure which contains the information from the provided current call list entry.

etHFRE_Network_Operator_Selection_Format_Indication

Indicates that the remote hands-free device has set the network operator selection format for the local audio gateway. In most cases no action is required other than returning a successful result.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
  unsigned int Format;
} HFRE_Network_Operator_Selection_Format_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
Format	Format value included in the original request. This value can generally be ignored.

etHFRE_Network_Operator_Selection_Indication

Request from the remote hands-free device to retreive the current operator selection information for the local audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
} HFRE_Network_Operator_Selection_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

etHFRE_Network_Operator_Selection_Confirmation

Response from the remote audio gateway containing the request operator selection information sent to the local hands-free device.

Return Structure:

```
typedef struct
{
  unsigned int  HFREPortID;
  unsigned int  NetworkMode;
  char         *NetworkOperator;
} HFRE_Network_Operator_Selection_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
NetworkMode	The network mode of the AG which sent this response. The possible values for mode are defined in the HFRE and/or GSM specification.
NetworkOperator	A null-terminated ASCII string containing the operator name string returned by the remote AG (e.g. “MyCarrier”).

etHFRE_Extended_Error_Result_Activation_Indication

Request from a remote hands-free device to indicate that it supports the reception of extended error information. Following this event the audio gateway can return extended error information to the remote hands-free device.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
  Boolean_t    Enabled;
} HFRE_Extended_Error_Result_Activation_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
Enabled	Indicates if extended error result activation is supported by the remote hands-free device.

etHFRE_Subscriber_Number_Information_Indication

Request from a remote hands-free device to retrieve the current subscriber number information from the local audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
} HFRE_Subscriber_Number_Information_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

etHFRE_Subscriber_Number_Information_Confirmation

Response from the remote audio gateway containing the request subscriber number information. Each event contains a single set of subscriber information. The device can detect that no further confirmations are expected once it receives the etHFRE_Command_Result event indicating a successful terminating response (OK).

Return Structure:

```
typedef struct
{
  unsigned int  HFREPortID;
  unsigned int  ServiceType;
  unsigned int  NumberFormat;
  char          *PhoneNumber;
} HFRE_Subscriber_Number_Information_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
ServiceType	Indicates the service type value provided by the remote AG.
NumberFormat	Indicates the number format provided by the remote AG.

PhoneNumber	This null-terminated ASCII string contains the phone number provided by the remote AG for this subscriber number response.

etHFRE_Response_Hold_Status_Indication

Request from a remote hands-free device to retrieve the current response and hold status of the local audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
} HFRE_Response_Hold_Status_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

etHFRE_Response_Hold_Status_Confirmation

Response from the remote hands-free device containing the current response and hold state of the remote audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int          HFREPortID;
  HFRE_Call_State_t CallState;
} HFRE_Response_Hold_Status_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
CallState	Indicates the current call state of the remote hands-free device.

etHFRE_Incoming_Call_State_Indication

This event is received either when a local audio gateway receives a command to set the current call state OR by a local hands-free device to indicate the remote audio gateway.

Return Structure:

```
typedef struct
{
  unsigned int          HFREPortID;
  HFRE_Call_State_t CallState;
} HFRE_Incoming_Call_State_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

CallState	Indicates the current call state of the remote device.

etHFRE_Incoming_Call_State_Confirmation

Response from a remote audio gateway containing its current response and hold status.

Return Structure:

```
typedef struct
{
  unsigned int      HFREPortID;
  HFRE_Call_State_t CallState;
} HFRE_Incoming_Call_State_Confirmation_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
CallState	Indicates the current call state of the remote device.

etHFRE_Command_Result

Indication to the local hands-free device that the remote audio gateway has sent a terminating response indicating either success (OK) or one of the possible failure/error types. This event is also generated when the remote audio gateway generates an unsolicited result code to indicate an unexpected error.

Return Structure:

```
typedef struct
{
  unsigned int                HFREPortID;
  HFRE_Extended_Result_t      ResultType;
  unsigned int                ResultValue;
} HFRE_Command_Result_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
ResultType	The type of result sent by the remote device. The following values are supported: erOK erError erNoCarrier erBusy erNoAnswer erDelayed erBlacklisted erResultCode
ResultValue	If appropriate, the value of an extended error result. This value is only valid for a ResultType of erResultCode, otherwise it should be zero.

etHFRE_Arbitrary_Command_Indication

This event is dispatched to a local Audio Gateway unit when the remote Hands Free Device issues a command that is not recognized by the local Audio Gateway (i.e. an arbitrary AT command).

Return Structure:

```
typedef struct
{
  unsigned int      HFREPortID;
  char              *HFRECommandData;
} HFRE_Arbitrary_Command_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
HFRECommandData	Null terminated string that represents the actual command data that was received.

etHFRE_Arbitrary_Response_Indication

This event is dispatched to a local Hands Free unit when the remote Audio Gateway issues an arbitrary response.

Return Structure:

```
typedef struct
{
  unsigned int      HFREPortID;
  char              *HFREResponseData;
} HFRE_Arbitrary_Response_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.
HFREResponseData	Null terminated string that represents the actual response data that was received.

etHFRE_Control_Indicator_Request_Indication

This event is dispatched to a local Audio Gateway unit when the remote Hands Free unit issues a request for the current control indicator and the local Audio Gateway has registered for these events.

Notes:

1. This event will only be dispatched if the HFRE_AG_QUERY_INDICATOR_REQUEST_SUPPORTED_BIT bit is set in the SupportedFeaturesMask that was passed to either the HFRE_Open_Remote_HandsFree_Port() or HFRE_Open_Audio_Gateway_Server_Port() that returned the specified HFREPortID.

2. If an application receives this event it must call the HFRE_Update_Current_Control_Indicator_Status() API to update the value of all the supported control indicators (if their value has changed since the last time the API was called) and then call the HFRE_Send_Control_Indicator_Request_Response() API to respond to the request.

Return Structure:

```
typedef struct
{
  unsigned int HFREPortID;
} HFRE_Control_Indicator_Request_Indication_Data_t;
```

Event Parameters:

HFREPortID	Identifier of the HFRE connection.

3. File Distributions

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

File	Contents/Description
HFREAPI.h	Bluetooth Hands-Free Profile API definitions
SS1BTHFR.h	Bluetooth Hands-Free Profile Include file