Audio Profile Sub-system (AUD)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
September 11, 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 audio profile sub-system provided by Bluetopia. Chapter 2 contains a description of the programming interface for this profile sub-system. And, Chapter 3 contains the header file name list for the Bluetooth audio profile sub-system library.

1.1 Scope

This reference manual provides information on the Audio Sub-system API (depicted in Figure 1-1 below). This API is available on the full range of platforms supported by Stonestreet One:

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

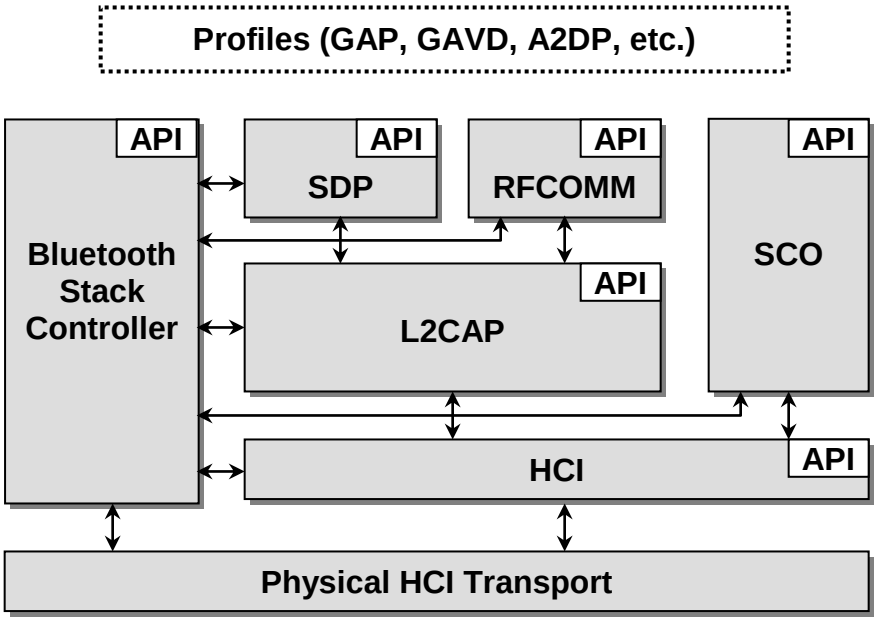

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

1.2 Applicable Documents

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

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

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

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

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

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

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

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

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

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

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

1.3 Acronyms and Abbreviations

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

Term	Meaning
A2DP	Advanced Audio Distribution Profile
API	Application Programming Interface
AUD	Audio profile Sub-system
AVRCP	Audio/Video Remote Control Profile
AVCTP	Audio/Video Control Transport Protocol
AVDTP	Audio/Video Distribution Transport Protocol
BD_ADDR	Bluetooth Device Address
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Data Rate
GAP	Generic Access Profile

Term	Meaning
GAVD	Generic Audio/Video Distribution
HS	High Speed
LE	Low Energy
LSB	Least Significant Bit
MSB	Most Significant Bit
SDP	Service Discovery Protocol
SPP	Serial Port Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. Audio Profile Sub-system Programming Interface

The audio profile sub-system (AUD) programming interface defines the protocols and procedures to be used to implement audio (SRC/SNK) and A/V remote control capabilities (Controller/Target). The sub-system commands are listed in section 2.1, the event callback prototype is described in section 2.2, and the events are itemized in section 2.3. The actual prototypes and constants outlined in this section can be found in the **AUDAPI.H** header file in the Bluetopia distribution.

2.1 Audio Profile Sub-system Commands

The available AUD command functions are listed in the table below and are described in the text which follows.

Function	Description
AUD_Initialize	This function is responsible for initializing the audio profile sub-system.
AUD_Un_Initialize	This function is responsible for un-initializing the audio profile sub-system.
AUD_Change_Media_Codec_Parameters	This function allows the ability to change select SBC media codec parameters that are used for SBC endpoints
AUD_Open_Request_Response	This function is responsible for responding to an individual request to connect to a local audio profile sub-system server.
AUD_Open_Remote_Stream	This function is responsible for opening an audio stream (SRC/SNK) connection to a remote audio stream service (SNK/SRC).
AUD_Close_Stream	This function is responsible for closing a previously opened (either local or remote) audio stream.
AUD_Open_Remote_Control	This function is responsible for opening a remote control connection to a remote device
AUD_Close_Remote_Control	This function is responsible for closing an existing remote control connection to a remote device
AUD_Open_Browsing_Channel	This function is responsible for initiating a Browsing connection to a remote device.
AUD_Close_Browsing_Channel	This function is responsible for disconnecting any Opened Browsing channel to the specified remote device.

AUD_Change_Stream_State	This function allows a mechanism for requesting an audio stream state change on an open audio stream.
AUD_Query_Stream_State	This function allows a mechanism to determine the current audio stream state of an open audio stream.
AUD_Change_Stream_Format	This function allows a mechanism for requesting an audio stream format change on an open audio stream.
AUD_Query_Stream_Format	This function allows a mechanism to determine the current audio stream format of an open audio stream.
AUD_Query_Stream_Configuration	This function allows a mechanism to determine the current audio stream configuration of an open (and configured) audio stream.
AUD_Send_Remote_Control_Command	The following function is responsible for sending a remote control command to a remotely connected device.
AUD_Send_Remote_Control_Response	The following function is responsible for send a remote control response to a remotely connected device.
AUD_Send_Encoded_Audio_Data	The following function is responsible for sending encoded audio data to a remote audio SNK (local SRC stream).
AUD_Send_RTP_Encoded_Audio_Data	The following function is responsible for sending encoded audio data to a remote audio SNK (local SRC stream) while specifying RTP information for the data.
AUD_Get_Server_Connection_Mode	This function is responsible for retrieving the current audio profile sub-system server connection mode.
AUD_Set_Server_Connection_Mode	This function is responsible for setting the audio profile sub-system server connection mode.
AUD_Query_Stream_Channel_Information	This function is responsible for querying the channel information for the Media channel.

AUD_Initialize

This function is responsible for initializing the audio profile sub-system. This function accepts the SRC/SNK initialization information (at least one MUST be specified – SRC or SNK, however, both can be specified – SRC and SNK). This function also accepts the audio profile sub-system event callback function (and parameter) that will be issued whenever an audio profile sub-system event occurs.

Notes:

If there is a specific feature that is NOT required, then NULL can be passed a the pointer in the initialization structure. The only requirement/exception is that there must be at least one feature to be initialized (i.e. at least one of Stream SRC, Stream SNK, Remote Control Controller, or Remote Control Target). Note that there is no requirement that at least one of Stream and Remote Control be specified. This means that this module can function as a Remote Control entity only (either Controller, Target, or both) or an A2DP entity only (either SRC, SNK, or both) or any combination of A2DP and Remote Control features.

Prototype:

int BTPSAPI **AUD_Initialize**(unsigned int BluetoothStackID, AUD_Initialization_Info_t *InitializationInfo, AUD_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

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

InitializationInfo — Pointer to an audio profile sub-system initialization information structure. This structure has the following format:

```
typedef struct
{
  unsigned long                          InitializationFlags;
  AUD_Stream_Initialization_Info_t
                                          *SRCInitializationInfo;
  AUD_Stream_Initialization_Info_t
                                         *SNKInitializationInfo;
  AUD_Remote_Control_Initialization_Info_t
                                    *RemoteControlInitializationInfo;
  Word_t                                  MediaMTU;
} AUD_Initialization_Info_t;
```

where InitializationFlags is currently defined as a bitmask that has zero or more of the following value(s):

AUD_INITIALIZATION_INFO_FLAGS_OVERRIDE_MEDIA_MTU

SRCInitializationInfo and SNKIntializationInfo are defined to be:

```
typedef struct
{
```

```
   unsigned long              InitializationFlags;
   char                      *EndpointSDPDescription;
   unsigned int               NumberSupportedStreamFormats;
   AUD_Stream_Format_t        StreamFormat[AUD_
                                STREAM_FORMAT_
                                MAXIMUM_NUMBER_
                                FORMATS];
   unsigned int               NumberConcurrentStreams;
} AUD_Initialization_Info_t;
```

where, InitializationFlags is currently defined as a bit-mask that has zero or more of the following values:

```
AUD_STREAM_INITIALIZATION_FLAGS_NUMBER_
    CONCURRENT_STREAMS_PRESENT
AUD_STREAM_INITIALIZATION_FLAGS_GROUP_
    CONCURRENT_STREAMS
```

The RemoteControlInitializationInfo structure is defined to be:

```
typedef struct
{
   unsigned long                    InitializationFlags;
   AVRCP_Version_t                  SupportedVersion;
   AUD_Remote_Control_Role_Info_t *ControllerRoleInfo;
   AUD_Remote_Control_Role_Info_t *TargetRoleInfo;
} AUD_Remote_Control_Initialization_Info_t;
```

where, InitializationFlags is current unused, SupportedVersion is currently defined as one of the following:

```
apvVersion1_0
apvVersion1_3
apvVersion1_4
```

and ControllerRoleInfo and TargetRoleInfo are defined as follows:

```
typedef struct
{
   Word_t      SupportedFeaturesFlags;
   char       *ProviderName;
   char       *ServiceName;
} AUD_Remote_Control_Role_Info_t;
```

where, SupportedFeaturesFlags are the specified AVRCP supported features flags.

EventCallback	Specifies the audio profile sub-system Event Callback function.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each audio sub-system event.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_UNABLE_TO_INITIALIZE_GAVD
BTAUD_ERROR_UNABLE_TO_INITIALIZE_AVCTP
BTAUD_ERROR_INSUFFICIENT_RESOURCES

Notes:

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

AUD_Un_Initialize

This function is responsible for Un-Registering the audio profile sub-system. This function closes all connections and cleans up all resources associated with the audio profile sub-system.

Prototype:

int BTPSAPI **AUD_Un_Initialize**(unsigned int BluetoothStackID)

Parameters:

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

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION

Notes:

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

AUD_Change_Media_Codec_Parameters

This function is responsible for changing select SBC Media Codec Capabilities. In particular this function allows the minimum and maximum bit pools that are published for the specified source or sink endpoints to be updated.

Prototype:

int BTPSAPI AUD_Change_Media_Codec_Parameters(unsigned int BluetoothStackID, AUD_Stream_Type_t StreamType, unsigned int MinimumBitPool, unsigned int MaximumBitPool)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
StreamType	Audio stream type of the stream to update the SBC media codec capabilities. Values for this parameter are as follows: astSNK astSRC
MinimumBitPool	Specifies the new minimum bit pool value that is to be used for the SBC Media Codec Capabilities for the stream endpoint.
MaximumBitPool	Specifies the new maximum bit pool value that is to be used for the SBC Media Codec Capabilities for the stream endpoint.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION

Notes:

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

AUD_Open_Request_Response

This function is responsible for responding to an individual connection request to connect to any portion of the audio profile sub-system (stream endpoint or remote control profile). This function should be called in response to the receipt of an etAUD_Open_Request_Indication event.

Prototype:

int BTPSAPI **AUD_Open_Request_Response**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, AUD_Connection_Request_Type_t ConnectionRequestType, Boolean_t AcceptConnection)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Specifies the Bluetooth device address of the device that is connecting.
ConnectionRequestType	Specifies the connection request that is being responded. Valid values for this parameter are defined as follows: acrStream acrRemoteControl
AcceptConnection	Specifies whether to accept the pending connection request.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION

Possible Events:

etAUD_Stream_Open_Indication

Notes:

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

AUD_Open_Remote_Stream

The following function is responsible for opening an audio streaming endpoint on the specified remote device. This function accepts the remote Bluetooth device address as well as the local audio stream type to connect.

Prototype:

int BTPSAPI **AUD_Open_Remote_Stream**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Address of the remote Bluetooth device to connect.
StreamType	Local audio stream type (not remote audio stream type) of the remote device to connect. Values for this parameter are as follows: astSNK astSRC

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_STREAM_ALREADY_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED
BTAUD_ERROR_UNABLE_TO_CONNECT_REMOTE_STREAM
BTAUD_ERROR_STREAM_CONNECTION_IN_PROGRESS

Possible Events:

etAUD_Stream_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.

AUD_Close_Stream

The following function is responsible for closing a currently open audio stream endpoint on the local device. This function accepts the local stream endpoint type of the currently active audio stream.

Prototype:

int BTPSAPI **AUD_Close_Stream**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth device address of the remote device that the stream to close is located.

StreamType	The local audio stream endpoint type to close. Values for this parameter are as follows: astSNK astSRC

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED

Notes:

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

AUD_Open_Remote_Control

The following function is responsible for opening a remote control connection to the specified remote device. This function accepts the remote Bluetooth device address to connect.

Notes:

This function **DOES NOT** open the audio stream, it **ONLY** makes an remote control (if one is not already established).

This function should ***ONLY*** be used if the caller wishes to create a remote control connection to the remote device but ***DOES NOT*** want to create a stream connection. The reason is that this module will automatically create a remote control connection to the remote device when a stream connection is made (if the remote device supports a remote control connection).

If this function is called to establish a remote control connection (and the connection is successful), the caller ***MUST*** call the **AUD_Close_Remote_Control**() function to disconnect the remote control connection.

Prototype:

int BTPSAPI AUD_Open_Remote_Control(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR)

Parameters:

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

BD_ADDR	Address of the remote Bluetooth device to connect.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_REMOTE_CONTROL_NOT_INITIALIZED
BTAUD_ERROR_REMOTE_CONTROL_ALREADY_CONNECTED
BTAUD_ERROR_REMOTE_CONTROL_NOT_CONNECTED
BTAUD_ERROR_REMOTE_CONTROL_CONNECTION_IN_PROGRESS

Possible Events:

etAUD_Remote_Control_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.

AUD_Close_Remote_Control

The following function is responsible for closing a currently open remote control connection on the local device. This function accepts the Bluetoth device address of the remote control connection to disconnect.

Notes:

This function should only be called if the local device previously issued a call to the **AUD_Open_Remote_Control**() function and a remote control connection was successfully established.

Prototype:

int BTPSAPI **AUD_Close_Remote_Control**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth device address of the remote device that is to have the remote control connection closed.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_REMOTE_CONTROL_NOT_CONNECTED
BTAUD_ERROR_REMOTE_CONTROL_NOT_INITIALIZED

Notes:

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

AUD_Open_Browsing_Channel

The following function is responsible for initiating a Browsing Channel connection to a remote device. It will try to establish L2CAP channel if no channel exists to the remote device.

Notes:

A Browsing Channel can *ONLY* be added if there already exists an on-going AVCTP connection between the device and the remote device already.

The Browsing Channel cannot exist without corresponding AVCTP connection. This means that if the AVCTP connection is terminated, the Browsing Channel connection will be terminated as well.

Prototype:

int BTPSAPI **AUD_Open_Browsing_Channel**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR)

Parameters:

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

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION

Notes:

1. 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.

AUD_Close_Browsing_Channel

The following function is responsible for disconnecting any Opened Browsing channel to the specified remote device.

Prototype:

int BTPSAPI **AUD_Close_Browsing_Channel**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR)

Parameters:

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

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION

Notes:

1. 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.

AUD_Change_Stream_State

The following function is responsible for Changing the stream state of a currently opened audio stream endpoint on the local device.

Prototype:

int BTPSAPI **AUD_Change_Stream_State**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType, AUD_Stream_State_t StreamState)

Parameters:

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

BD_ADDR	Bluetooth device address of the remote device that the stream to change the state of is located.
StreamType	The local audio stream endpoint type to change the state of. Values for this parameter are as follows: astSNK astSRC
StreamState	The new audio stream endpoint state. Values for this parameter are as follows: astStreamStopped astStreamStarted

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED
BTAUD_ERROR_STREAM_STATE_CHANGE_IN_PROGRESS
BTAUD_ERROR_STREAM_FORMAT_CHANGE_IN_PROGRESS
BTAUD_ERROR_STREAM_STATE_ALREADY_CURRENT
BTAUD_ERROR_UNABLE_TO_CHANGE_STREAM_STATE

Possible Events:

etAUD_Stream_Close_Indication
etAUD_Stream_State_Change_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.

AUD_Query_Stream_State

This function is responsible for querying the current audio stream state of a currently opened audio stream endpoint on the local device.

Prototype:

int BTPSAPI **AUD_Query_Stream_State**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType,
AUD_Stream_State_t *StreamState)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth device address of the remote device that the stream to query the state of is located.
StreamType	The local audio stream endpoint type to query the state of. Values for this parameter are as follows: astSNK astSRC
StreamState	Pointer to a buffer that is receive the current stream endpoint state.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED

Possible Events:

etAUD_Stream_Close_Indication

Notes:

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

AUD_Change_Stream_Format

This function is responsible for changing the stream format of a currently opened stream endpoint on the local device. This function can only be called when the state of the stream is suspended.

Prototype:

int BTPSAPI **AUD_Change_Stream_Format**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType,
AUD_Stream_Format_t *StreamFormat)

Parameters:

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

BD_ADDR — Bluetooth device address of the remote device that the stream to change the format of is located.

StreamType — The local audio stream endpoint type to change the format of. Values for this parameter are as follows:

astSNK
astSRC

StreamFormat — The new audio stream endpoint format. The structure of this parameter is defined as follows:

```
typedef struct
{
  unsigned long   SampleFrequency;
  unsigned int    NumberChannels;
  unsigned long   FormatFlags;
} AUD_Stream_Format_t;
```

where, FormatFlags is a bitmask of one or more of the following:

AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_BIT_MASK
AUD_STREAM_FORMAT_FLAGS_SCMS_T_PROTECTION_SUPPORTED

Valid values for the Codec Type Bit Mask are:

AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_SBC
AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_AAC
AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_MP3

Note, that depending upon the Codec type, the remaining flags can be one or more of the following:

AUD_STREAM_FORMAT_FLAGS_DUAL_CHANNEL_NOT_SUPPORTED
AUD_STREAM_FORMAT_FLAGS_JOINT_STEREO_NOT_SUPPORTED

for the SBC and MP3 Codec Types, and:

AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LC
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LTP
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_SCALABLE
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_VBR

for the AAC Codec, and:

AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_1
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_2
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_3
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_CRC_PROTECTION
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_PAYLOAD_FORMAT_2
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_VBR

for the MP3 Codec.

Also note that the SCMS-T content protection flag will only be valid if the sub-system was initialized specifying SCMS-T copy protection support AND the remote device supports SCMS-T copy protection.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED
BTAUD_ERROR_STREAM_IS_ACTIVE
BTAUD_ERROR_STREAM_STATE_CHANGE_IN_PROGRESS
BTAUD_ERROR_STREAM_FORMAT_CHANGE_IN_PROGRESS
BTAUD_ERROR_STREAM_FORMAT_ALREADY_CURRENT
BTAUD_ERROR_UNABLE_TO_CHANGE_STREAM_FORMAT
BTAUD_ERROR_UNSUPPORTED_FORMAT

Possible Events:

etAUD_Stream_Close_Indication
etAUD_Stream_Format_Change_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.

AUD_Query_Stream_Format

This function is responsible for querying the current audio stream format of a currently opened stream endpoint on the local device.

Prototype:

int BTPSAPI **AUD_Query_Stream_Format**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType,
AUD_Stream_Format_t *StreamFormat)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth device address of the remote device that the stream to query the format of is located.

StreamType	The local audio stream endpoint type to query the format of. Values for this parameter are as follows: astSNK astSRC
StreamFormat	Pointer to a buffer that is receive the current stream endpoint audio format.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED

Possible Events:

etAUD_Stream_Close_Indication

Notes:

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

AUD_Query_Stream_Configuration

This function is responsible for querying the current stream configuration of a currently opened stream endpoint on the local device. This function can be used to determine the current low-level A2DP configuration parameters of the active audio stream. These parameters can then be applied to the host encoding/decoding process.

Prototype:

int BTPSAPI **AUD_Query_Stream_Configuration**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType, AUD_Stream_Configuration_t *StreamConfiguration)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth device address of the remote device that the stream to query the configuration of is located.
StreamType	The stream endpoint type to query the configuration of. Values for this parameter are as follows:

astSNK
astSRC

StreamConfiguration — Pointer to a buffer that is receive the current stream endpoint configuration. This pointer must point to a buffer that is defined to have the following format:

```
typedef struct
{
   unsigned int              MediaMTU;
   AUD_Stream_Format_t       StreamFormat;
   unsigned int              MediaCodecType;
   unsigned int              MediaCodecInfoLength;
   unsigned char             MediaCodecInformation[
                               AUD_MEDIA_CODEC_
                               INFORMATION_SIZE_
                               MAXIMUM_SIZE];
} AUD_Stream_Configuration_t;
```

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_NOT_INITIALIZED

Possible Events:

etAUD_Stream_Close_Indication

Notes:

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

AUD_Send_Remote_Control_Command

This function is responsible for sending the specified AVRCP remote control command to the specified remote Device. This function is only applicable for a local audio SRC device. This means that if there is no actively connected audio SNK device then this call will fail.

Prototype:

int BTPSAPI **AUD_Send_Remote_Control_Command**(unsigned int BluetoothStackID,
BD_ADDR_t BD_ADDR,
AUD_Remote_Control_Command_Data_t *RemoteControlCommandData,
unsigned long ResponseTimeout)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	The Bluetooth device address of the device that is to receive the remote control command.
RemoteControlCommandData	The remote control command information. This parameter has the following format:

```
typedef struct
{
  AVRCP_Message_Type_t MessageType;
  union
  {
    AVRCP_Unit_Info_Command_Data_t
                UnitInfoCommandData;
    AVRCP_Subunit_Info_Command_Data_t
                SubunitInfoCommandData;
    AVRCP_Pass_Through_Command_Data_t
                PassThroughCommandData;
    AVRCP_Vendor_Dependent_Generic_Command_Data_t
                VendorDependentGenericCommandData;
    AVRCP_Browsing_Channel_Generic_Message_Data_t
                BrowsingChannelGenericMessageData;
    AVRCP_Group_Navigation_Command_Data_t
                GroupNavigationCommandData;
    AVRCP_Get_Capabilities_Command_Data_t
                GetCapabilitiesCommandData;
    AVRCP_List_Player_Application_Setting_Values_
                Command_Data_t
                ListPlayerApplicationSettingValuesCommandD
                ata;
    AVRCP_Get_Current_Player_Application_Setting_
                Value_Command_Data_t
                GetCurrentPlayerApplicationSetting
                ValueCommandData;
    AVRCP_Set_Player_Application_Setting_Value_
                Command_Data_t
                SetPlayerApplicationSettingValueCommandD
                ata;
    AVRCP_Get_Player_Application_Setting_Attribute_
                Text_Command_Data_t
                GetPlayerApplicationSettingAttribute
                TextCommandData;
    AVRCP_Get_Player_Application_Setting_Value_
                Text_Command_Data_t
                GetPlayerApplicationSettingValue
                TextCommandData;
    AVRCP_Inform_Displayable_Character_Set_Command_
                Data_t
```

```
                  InformDisplayableCharacterSet
                  CommandData;
            AVRCP_Inform_Battery_Status_Of_CT_Command_
                  Data_t
                  InformBatteryStatusOfCTCommandData;
            AVRCP_Get_Element_Attributes_Command_Data_t
                  GetElementAttributesCommandData;
            AVRCP_Request_Continuing_Response_Command_
                  Data_t
                  RequestContinuingResponseCommandData;
            AVRCP_Abort_Continuing_Response_Command_
                  Data_t
                  AbortContinuingResponseCommandData;
            AVRCP_Get_Play_Status_Response_Data_t
                  GetPlayStatusResponseData;
            AVRCP_Register_Notification_Command_Data_t
                  RegisterNotificationCommandData;
            AVRCP_Set_Absolute_Volume_Command_Data_t
                  SetAbsoluteVolumeCommandData;
            AVRCP_Set_Addressed_Player_Command_Data_t
                  SetAddressedPlayerCommandData;
            AVRCP_Play_Item_Command_Data_t
                  PlayItemCommandData;
            AVRCP_Add_To_Now_Playing_Command_Data_t
                  AddToNowPlayingCommandData;
            AVRCP_Set_Browsed_Player_Command_Data_t
                  SetBrowsedPlayerCommandData;
            AVRCP_Change_Path_Command_Data_t
                  ChangePathCommandData;
            AVRCP_Get_Item_Attributes_Command_Data_t
                  GetItemAttributesCommandData;
            AVRCP_Search_Command_Data_t
                  SearchCommandData;
            AVRCP_Search_Response_Data_t
                  SearchResponseData;
            AVRCP_Get_Folder_Items_Command_Data_t
                  GetFolderItemsCommandData;
         } MessageData;
      } AUD_Remote_Control_Command_Data_t;
```

ResponseTimeout	The Timeout value (in milliseconds) to wait for a response (confirmation) from the remote device. Note that if this value is specified as zero then no confirmation will be issued as the audio sub-system will not track the remote control command internally.

Return:

Positive (non-zero) return value if successful which represents the internal transaction ID of the remote control command. This can be used to match up the remote control command confirmation result.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_UNABLE_TO_SEND_REMOTE_CONTROL_COMMAND
BTAUD_ERROR_REMOTE_CONTROL_NOT_CONNECTED
BTAUD_ERROR_INVALID_REMOTE_CONTROL_DATA
BTAUD_ERROR_REMOTE_DEVICE_NOT_CONNECTED

Possible Events:

etAUD_Remote_Control_Command_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.

AUD_Send_Remote_Control_Response

This function is responsible for sending the specified remote control response to the remote device that originated a remote control command.

Prototype:

int BTPSAPI **AUD_Send_Remote_Control_Response**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, unsigned int TransactionID, AUD_Remote_Control_Response_Data_t *RemoteControlResponseData)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	The Bluetooth device address of the remote device that sent the remote control command.
TransactionID	The transaction ID of the original remote control transaction.
RemoteControlResponseData	The remote control response information. This parameter has the following format: typedef struct { AVRCP_Message_Type_t MessageType; union { AVRCP_Unit_Info_Response_Data_t UnitInfoResponseData; AVRCP_Subunit_Info_Response_Data_t SubunitInfoResponseData;

```
AVRCP_Pass_Through_Response_Data_t
                PassThroughResponseData;
AVRCP_Vendor_Dependent_Generic_Response_Data_t
                VendorDependentGenericResponseData;
AVRCP_Group_Navigation_Response_Data_t
                GroupNavigationResponseData;
AVRCP_Get_Capabilities_Response_Data_t
                GetCapabilitiesResponseData;
AVRCP_List_Player_Application_Setting_Attributes_
                Response_Data_t
                ListPlayerApplicationSettingAttributes
                ResponseData;
AVRCP_List_Player_Application_Setting_Values_
                Response_Data_t
                ListPlayerApplicationSettingValues
                ResponseData;
AVRCP_Get_Current_Player_Application_Setting_Value_
                Response_Data_t
                GetCurrentPlayerApplicationSetting
                ValueResponseData;
AVRCP_Set_Player_Application_Setting_Value_Response_
                Data_t
                SetPlayerApplicationSettingValue
                ResponseData;
AVRCP_Get_Player_Application_Setting_Attribute_Text_
                Response_Data_t
                GetPlayerApplicationSettingAttribute
                TextResponseData;
AVRCP_Get_Player_Application_Setting_Value_Text_
                Response_Data_t
                GetPlayerApplicationSettingValueText
                ResponseData;
AVRCP_Inform_Displayable_Character_Set_Response_
                Data_t
                InformDisplayableCharacterSetResponse
                Data;
AVRCP_Inform_Battery_Status_Of_CT_Response_Data_t
                InformBatteryStatusOfCTResponseData;
AVRCP_Get_Element_Attributes_Response_Data_t
                GetElementAttributesResponseData;
AVRCP_Get_Play_Status_Response_Data_t
                GetPlayStatusResponseData;
AVRCP_Register_Notification_Response_Data_t
                RegisterNotificationResponseData;
AVRCP_Abort_Continuing_Response_Response_Data_t
                AbortContinuingResponseResponseData;
AVRCP_Set_Absolute_Volume_Response_Data_t
                SetAbsoluteVolumeResponseData;
AVRCP_Set_Addressed_Player_Response_Data_t
                SetAddressedPlayerResponseData;
```

```
      AVRCP_Play_Item_Response_Data_t
                  PlayItemResponseData;
      AVRCP_Add_To_Now_Playing_Response_Data_t
                  AddToNowPlayingResponseData;
      AVRCP_Command_Reject_Response_Data_t
                  CommandRejectResponseData;
      AVRCP_Set_Browsed_Player_Response_Data_t
                  SetBrowsedPlayerResponseData;
      AVRCP_Change_Path_Response_Data_t
                  ChangePathResponseData;
      AVRCP_Get_Item_Attributes_Response_Data_t
                  GetItemAttributesResponseData;
      AVRCP_Search_Response_Data_t
                  SearchResponseDat;
      AVRCP_Get_Folder_Items_Response_Data_t
                  GetFolderItemsResponseData;
      AVRCP_General_Reject_Response_Data_t
                  GeneralRejectResponseData;
   } MessageData;
} AUD_Remote_Control_Response_Data_t;
```

Return:

Zero if successful.

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

```
BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_UNABLE_TO_SEND_REMOTE_CONTROL_
      RESPONSE
BTAUD_ERROR_REMOTE_CONTROL_NOT_CONNECTED
BTAUD_ERROR_INVALID_REMOTE_CONTROL_DATA
BTAUD_ERROR_REMOTE_DEVICE_NOT_CONNECTED
```

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.

AUD_Send_Encoded_Audio_Data

This function is responsible for sending the specified encoded audio data to the remote SNK (local SRC stream endpoint).

Notes:

This is a low level function that exists for applications that would like to encode the audio data themselves (as opposed to having this module encode and send the data). The caller can determine the current configuration of the stream by calling the **AUD_Query_Stream_Configuration**() function.

The data that is sent *MUST* contain the AVDTP Header Information (i.e. the first byte of the data *MUST* be a valid AVDTP Header byte).

This function assumes the specified data is being sent at real time pacing, and the data is queued to be sent immediately.

Prototype:

int BTPSAPI **AUD_Send_Encoded_Audio_Data**(unsigned int BluetoothStackID, unsigned int RawAudioDataFrameLength, unsigned char *RawAudioDataFrame)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
RawAudioDataFrameLength	The number of bytes of raw, encoded, audio frame information.
RawAudioDataFrame	The raw, encoded, audio data to send.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_INITIALIZED
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_IS_NOT_ACTIVE
BTAUD_ERROR_UNABLE_TO_SEND_STREAM_DATA

Possible Events:

etAUD_Stream_Close_Indication

Notes:

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

AUD_Send_RTP_Encoded_Audio_Data

This function is responsible for sending the specified encoded audio data to the remote SNK (local SRC stream endpoint). This function differs from the **AUD_Send_Encoded_Audio_Data**() function in that this function allows the RTP payload information to specified for the data.

Notes:

This is a low level function that exists for applications that would like to encode the audio data themselves (as opposed to having this module encode and send the data). The caller can determine the current configuration of the stream by calling the **AUD_Query_Stream_Configuration**() function.

The data that is sent *MUST* contain the AVDTP Header Information (i.e. the first byte of the data *MUST* be a valid AVDTP Header byte).

This function assumes the specified data is being sent at real time pacing, and the data is queued to be sent immediately.

Prototype:

int BTPSAPI **AUD_Send_RTP_Encoded_Audio_Data**(unsigned int BluetoothStackID, unsigned int RawAudioDataFrameLength, unsigned char *RawAudioDataFrame, unsigned long Flags, AUD_RTP_Header_Info_t *RTPHeaderInfo)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
RawAudioDataFrameLength	The number of bytes of raw, encoded, audio frame information.
RawAudioDataFrame	The raw, encoded, audio data to send.
Flags	Optional flags that specify RTP options. Currently there are no defined values for this parameter and zero should be passed for this parameter.
RTPHeaderInfo	The RTP information that should be sent with this packet of audio data. This structure has the following definition:

```
typedef struct
{
  Word_t      SequenceNumber;
  DWord_t     TimeStamp;
  Byte_t      PayloadType;
  Boolean_t   Marker;
} AUD_RTP_Header_Info_t;
```

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED
BTAUD_ERROR_INVALID_OPERATION
BTAUD_ERROR_STREAM_NOT_INITIALIZED
BTAUD_ERROR_STREAM_NOT_CONNECTED
BTAUD_ERROR_STREAM_IS_NOT_ACTIVE
BTAUD_ERROR_UNABLE_TO_SEND_STREAM_DATA

Possible Events:

etAUD_Stream_Close_Indication

Notes:

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

AUD_Get_Server_Connection_Mode

This function is responsible for retrieving the current audio sub-system server connection mode.

Prototype:

int BTPSAPI **AUD_Get_Server_Connection_Mode**(unsigned int BluetoothStackID,
AUD_Server_Connection_Mode_t *ServerConnectionMode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServerConnectionMode	Pointer to a server connection mode variable which will receive the current server connection mode.

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED

Notes:

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

AUD_Set_Server_Connection_Mode

This function is responsible for setting the current audio sub-system server connection mode.

Prototype:

int BTPSAPI **AUD_Set_Server_Connection_Mode**(unsigned int BluetoothStackID, AUD_Server_Connection_Mode_t ServerConnectionMode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServerConnectionMode	The new server connection mode to set the Server to use. This parameter must be one of the following: ausAutomaticAccept ausAutomaticReject ausManualAccept

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED

Notes:

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

AUD_Query_Stream_Channel_Information

This function is responsible for querying the channel information for the Media channel that is associated with the specified stream type.

Prototype:

int BTPSAPI **AUD_Query_Stream_Channel_Information** (unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, AUD_Stream_Type_t StreamType, AUD_Stream_Channel_Info_t *ChannelInformation)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth device address of the remote device that the stream to query the channel information of is located.

StreamType	The local audio stream type to query the channel information for. Values for this parameter are as follows: astSNK astSRC
ChannelInformation	Pointer to a variable to receive the Channel Information. This structure is defined as follows: typedef struct { Word_t InMTU; Word_t OutMTU; Word_t LocalCID; Word_t RemoteCID; } AUD_Stream_Channel_Info_t;

Return:

Zero if successful.

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

BTAUD_ERROR_INVALID_PARAMETER
BTAUD_ ERROR_INVALID_BLUETOOTH_STACK_ID
BTAUD_ERROR_NOT_INITIALIZED

Notes:

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

2.2 Audio Profile Sub-system Event Callback Prototypes

The event callback functions, mentioned in the audio profile sub-system registration or connection functions, all accept the callback function described by the following prototype.

AUD_Event_Callback_t

Prototype of callback function passed to the **AUD_Initialize**() function. The function that is registered will receive ALL audio profile sub-system asynchronous events.

Prototype:

void (BTPSAPI ***AUD_Event_Callback_t**)(unsigned int BluetoothStackID,
AUD_Event_Data_t *AUD_Event_Data, unsigned long CallbackParameter)

Parameters:

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

```
typedef struct
{
  AUD_Event_Type_t  Event_Data_Type;
  Word_t            Event_Data_Size;
  union
  {
    AUD_Open_Request_Indication_Data_t
          *AUD_Open_Request_Indication_Data;
    AUD_Stream_Open_Indication_Data_t
          *AUD_Stream_Open_Indication_Data;
    AUD_Stream_Open_Confirmation_Data_t
          *AUD_Stream_Open_Confirmation_Data;
    AUD_Stream_Close_Indication_Data_t
          *AUD_Stream_Close_Indication_Data;
    AUD_Stream_State_Change_Indication_Data_t
          *AUD_Stream_State_Change_Indication_Data;
    AUD_Stream_State_Change_Confirmation_Data_t
          *AUD_Stream_State_Change_Confirmation_Data;
    AUD_Stream_Format_Change_Indication_Data_t
          *AUD_Stream_Format_Change_Indication_Data;
    AUD_Stream_Format_Change_Confirmation_Data_t
          *AUD_Stream_Format_Change_Confirmation_Data;
    AUD_Encoded_Audio_Data_Indication_Data_t
          *AUD_Encoded_Audio_Data_Indication_Data;
    AUD_Remote_Control_Command_Indication_Data_t
          *AUD_Remote_Control_Command_Indication_Data;
    AUD_Remote_Control_Command_Confirmation_Data_t
          *AUD_Remote_Control_Command_Confirmation_Data;
    AUD_Remote_Control_Open_Indication_Data_t
          *AUD_Remote_Control_Open_Indication_Data;
    AUD_Remote_Control_Open_Confirmation_Data_t
          *AUD_Remote_Control_Open_Confirmation_Data;
    AUD_Remote_Control_Close_Indication_Data_t
          *AUD_Remote_Control_Close_Indication_Data;
    AUD_Signalling_Channel_Open_Indication_Data_t
          *AUD_Signalling_Channel_Open_Indication_Data;
    AUD_Signalling_Channel_Close_Indication_Data_t
          *AUD_Signalling_Channel_Close_Indication_Data;
    AUD_Browsing_Channel_Open_Indication_Data_t
          *AUD_Browsing_Channel_Open_Indication_Data;
    AUD_Browsing_Channel_Open_Confirmation_Data_t
          *AUD_Browsing_Channel_Open_Confirmation_Data;
    AUD_Browsing_Channel_Close_Indication_Data_t
          *AUD_Browsing_Channel_Close_Indication_Data;
  } Event_Data;
} AUD_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.

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 Audio Profile Sub-system Profile Events

The possible audio profile sub-system events from the Bluetooth stack is listed in the table below and are described in the text that follows:

Event	Description
etAUD_Open_Request_Indication	Event that signals a remote audio stream or remote control profile is attempting a connection
etAUD_Stream_Open_Indication	Event that signals that the local audio stream (SRC/SNK) now has an active connection
etAUD_Stream_Open_Confirmation	Event that signals the result of a local audio stream (SRC/SNK) connection to a remote device
etAUD_Stream_Close_Indication	Event that signals that a currently active audio stream (SRC/SNK) is no longer active
etAUD_Stream_State_Change_Indication	Event that signals that streaming state of a local audio stream (SRC/SNK) has changed
etAUD_Stream_State_Change_Confirmation	Event that signals the result of a local audio stream (SRC/SNK) state change request
etAUD_Stream_Format_Change_Indication	Event that signals that stream format of a local audio stream (SRC/SNK) has changed
etAUD_Stream_Format_Change_Confirmation	Event that signals the result of a local audio stream (SRC/SNK) format change request
etAUD_Encoded_Audio_Data_Indication	Event that signals that encoded audio data has been received on the local audio SNK from the currently connected remote audio SRC
etAUD_Remote_Control_Command_Indication	Event that signals that a AVRCP remote

	control command has been received by a remote device
etAUD_Remote_Control_Command_Confirmation	Event that signals the result of a local remote control command request
etAUD_Remote_Control_Open_Indication	Event that signals that a remote control connection has been established from a specific remote device
etAUD_Remote_Control_Open_Confirmation	Event that signals the result of an outgoing remote control connection
etAUD_Remote_Control_Close_Indication	Event that signal that a remote control connection is no longer established to a specific remote device
etAUD_Signalling_Channel_Open_Indication	Event that signals that a signalling channel has been established to/from a specific remote device
etAUD_Signalling_Channel_Close_Indication	Event that signals that a signaling channel is no longer established to a specific remote device
etAUD_Browsing_Channel_Open_Indication	Event that signals that a remote Browsing service is connected to the local Browsing service.
etAUD_Browsing_Channel_Open_Confirmation	Event that signals that an attempt to connect to a remote AUD Browsing Channel is complete.
etAUD_Browsing_Channel_Close_Indication	Event that signals that remote device disconnects the Browsing Channel from the Local service.

etAUD_Open_Request_Indication

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

Return Structure:

```
typedef struct
{
  BD_ADDR_t                    BD_ADDR;
  AUD_Connection_Request_Type_t ConnectionRequestType;
} AUD_Open_Request_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the Bluetooth device making the request.

ConnectionRequestType	Specifies the connection request that is being requested. Valid values for this parameter are defined as follows: acrStream acrRemoteControl

etAUD_Stream_Open_Indication

This event is dispatched when a remote audio device connects to the specified local audio stream.

Return Structure:

```
typedef struct
{
  BD_ADDR_t              BD_ADDR;
  unsigned int           MediaMTU;
  AUD_Stream_Type_t      StreamType;
  AUD_Stream_Format_t    StreamFormat;
} AUD_Stream_Open_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the Bluetooth device making the connection.
MediaMTU	Specifies the GAVD/AVDTP /Media MTU (in bytes) of the active stream connection.
StreamType	Audio stream type (local audio stream type) of the currently active stream. Values for this parameter are as follows: astSNK astSRC
StreamFormat	Specifies the current stream format that is configured for the local stream.

etAUD_Stream_Open_Confirmation

This event is dispatched to indicate the result (success or failure) of a previously submitted connection request via the **AUD_Open_Remote_Stream**() function.

Return Structure:

```
typedef struct
{
  BD_ADDR_t              BD_ADDR;
  unsigned int           OpenStatus;
  unsigned int           MediaMTU;
  AUD_Stream_Type_t      StreamType;
  AUD_Stream_Format_t    StreamFormat;
} AUD_Stream_Open_Confirmation_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote device that was connected.

OpenStatus	Specifies the connection status of the connection attempt. This will be one of the following values: AUD_STREAM_OPEN_CONFIRMATION_STATUS_SUCCESS AUD_STREAM_OPEN_CONFIRMATION_STATUS_CONNECTION_TIMEOUT AUD_STREAM_OPEN_CONFIRMATION_STATUS_CONNECTION_REFUSED AUD_STREAM_OPEN_CONFIRMATION_STATUS_UNKNOWN_ERROR
MediaMTU	Specifies the GAVD/AVDTP /Media MTU (in bytes) of the active stream connection.
StreamType	Audio stream type (local audio stream type) of the currently active stream. Values for this parameter are as follows: astSNK astSRC
StreamFormat	Specifies the current stream format that is configured for the local stream.

etAUD_Stream_Close_Indication

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

Return Structure:

```
typedef struct
{
  BD_ADDR_t                BD_ADDR;
  AUD_Stream_Type_t        StreamType;
  AUD_Disconnect_Reason_t  DisconnectReason;
} AUD_Stream_Close_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote device that disconnected.
StreamType	Audio stream type (local audio stream type) of the disconnected audio stream. Values for this parameter are as follows: astSNK astSRC
DisconnectReason	Reason for the disconnection. This value will be one of the following: adrRemoteDeviceDisconnect adrRemoteDeviceLinkLoss adrRemoteDeviceTimeout

etAUD_Stream_State_Change_Indication

This event is dispatched when an audio stream endpoint state changes (asynchronously).

Return Structure:

```
typedef struct
{
  BD_ADDR_t             BD_ADDR;
  AUD_Stream_Type_t     StreamType;
  AUD_Stream_State_t    StreamState;
} AUD_Stream_State_Change_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote Bluetooth device that the audio stream is present.
StreamType	Audio stream type (local audio stream type) of the audio stream. Values for this parameter are as follows: astSNK astSRC
StreamState	Current audio stream state. This value will be one of the following: astStreamStopped astStreamStarted

etAUD_Stream_State_Change_Confirmation

This event is dispatched to reflect the status of a prior audio stream endpoint state change (via the **AUD_Change_Stream_State**() function).

Return Structure:

```
typedef struct
{
  BD_ADDR_t             BD_ADDR;
  Boolean_t             Successful;
  AUD_Stream_Type_t     StreamType;
  AUD_Stream_State_t    StreamState;
} AUD_Stream_State_Change_Confirmation_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote Bluetooth device that the audio stream is present.
Successful	Boolean value that specifies whether or not the stream state change was successful (TRUE) or not (FALSE).
StreamType	Audio stream type (local audio stream type) of the audio stream. Values for this parameter are as follows: astSNK

astSRC

StreamState — The current audio stream state. This value will be one of the following:

astStreamStopped
astStreamStarted

etAUD_Stream_Format_Change_Indication

This event is dispatched when an audio stream endpoint format changes (asynchronously).

Return Structure:

```
typedef struct
{
  BD_ADDR_t              BD_ADDR;
  AUD_Stream_Type_t      StreamType;
  AUD_Stream_Format_t    StreamFormat;
} AUD_Stream_Format_Change_Indication_Data_t;
```

Event Parameters:

BD_ADDR — Address of the remote Bluetooth device that the audio stream is present.

StreamType — Audio stream type (local audio stream type) of the audio stream. Values for this parameter are as follows:

astSNK
astSRC

StreamFormat — The new audio stream endpoint format. The structure of this parameter is defined as follows:

```
typedef struct
{
  unsigned long  SampleFrequency;
  unsigned int   NumberChannels;
  unsigned long  FormatFlags;
} AUD_Stream_Format_t;
```

where, FormatFlags is a bitmask of one or more of the following:

AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_BIT_MASK
AUD_STREAM_FORMAT_FLAGS_SCMS_T_PROTECTION_SUPPORTED

Valid values for the Codec Type Bit Mask are:

AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_SBC
AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_AAC
AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_MP3

Note, that depending upon the Codec type, the remaining flags can be one or more of the following:

AUD_STREAM_FORMAT_FLAGS_DUAL_CHANNEL_NOT_SUPPORTED

AUD_STREAM_FORMAT_FLAGS_JOINT_STEREO_NOT_SUPPORTED

for the SBC and MP3 Codec Types, and:

AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LC
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LTP
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_SCALABLE
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_VBR

for the AAC Codec, and:

AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_1
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_2
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_3
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_CRC_PROTECTION
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_PAYLOAD_FORMAT_2
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_VBR

for the MP3 Codec.

Also note that the SCMS-T content protection flag will only be valid if the sub-system was initialized specifying SCMS-T copy protection support AND the remote device supports SCMS-T copy protection.

etAUD_Stream_Format_Change_Confirmation

This event is dispatched to reflect the status of a prior audio stream endpoint format change (via the **AUD_Change_Stream_Format**() function).

Return Structure:

```
typedef struct
{
   BD_ADDR_t              BD_ADDR;
   Boolean_t              Successful;
   AUD_Stream_Type_t      StreamType;
   AUD_Stream_Format_t    StreamFormat;
} AUD_Stream_Format_Change_Confirmation_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote Bluetooth device that the audio stream is present.
Successful	Boolean value that specifies whether or not the stream format change was successful (TRUE) or not (FALSE).
StreamType	Audio stream type (local audio stream type) of the audio stream. Values for this parameter are as follows: astSNK

astSRC

StreamFormat

The current audio stream endpoint format. The structure of this parameter is defined as follows:

```
typedef struct
{
  unsigned long  SampleFrequency;
  unsigned int   NumberChannels;
  unsigned long  FormatFlags;
} AUD_Stream_Format_t;
```

where, FormatFlags is a bitmask of one or more of the following:

AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_BIT_MASK
AUD_STREAM_FORMAT_FLAGS_SCMS_T_PROTECTION_SUPPORTED

valid values for the Codec Type Bit Mask are:

AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_SBC
AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_AAC
AUD_STREAM_FORMAT_FLAGS_CODEC_TYPE_MP3

Note, that depending upon the Codec type, the remaining flags can be one or more of the following:

AUD_STREAM_FORMAT_FLAGS_DUAL_CHANNEL_NOT_SUPPORTED
AUD_STREAM_FORMAT_FLAGS_JOINT_STEREO_NOT_SUPPORTED

for the SBC and MP3 Codec Types, and:

AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LC
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_LTP
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_MPEG4_SCALABLE
AUD_STREAM_FORMAT_FLAGS_AAC_SUPPORT_VBR

for the AAC Codec, and:

AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_1
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_2
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_LAYER_3
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_CRC_PROTECTION
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_PAYLOAD_FORMAT_2
AUD_STREAM_FORMAT_FLAGS_MP3_SUPPORT_VBR

for the MP3 Codec.

Also note that the SCMS-T content protection flag will only be valid if the sub-system was initialized specifying SCMS-T copy protection support AND the remote device supports SCMS-T copy protection.

etAUD_Encoded_Audio_Data_Indication

This event is dispatched when audio data is received on a SNK stream endpoint.

Notes:

This is low level event data that exists for applications that would like to decode the audio data themselves (as opposed to having this module decode the audio data). The caller can determine the current configuration of the stream by calling the **AUD_Query_Stream_Configuration**() function.

The data that is received *WILL* contain the AVDTP Header Information (i.e. the first byte of the data *WILL* be a valid AVDTP Header byte).

This event is dispatched at real time pacing (i.e. as soon as it has been received).

Return Structure:

```
typedef struct
{
  BD_ADDR_t                 BD_ADDR;
  unsigned int              RawAudioDataFrameLength;
  unsigned char            *RawAudioDataFrame;
  AUD_RTP_Header_Info_t    *RTPHeaderInfo;
} AUD_Encoded_Audio_Data_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote Bluetooth device that the audio stream is present.
RawAudioDataFrameLength	The number of bytes of raw, encoded, audio frame information.
RawAudioDataFrame	The raw, encoded, audio data that was received.
RTPHeaderInfo	The RTP information that was received with this packet of audio data. This structure has the following definition:

```
typedef struct
{
  Word_t       SequenceNumber;
  DWord_t      TimeStamp;
  Byte_t       PayloadType;
  Boolean_t    Marker;
} AUD_RTP_Header_Info_t;
```

etAUD_Remote_Control_Command_Indication

This event is dispatched when a remote control command is received to from a remote device (asynchronously).

Return Structure:

```
typedef struct
{
  BD_ADDR_t                              BD_ADDR;
  unsigned int                           TransactionID;
  AUD_Remote_Control_Command_Data_t      RemoteControlCommandData;
} AUD_Remote_Control_Command_Indication_Data_t;
```

EventParamerers:

BD_ADDR	Address of the remote Bluetooth device that the remote control command was received from.
TransactionID	AVRCP transaction ID of the received command (used when sending a response).
RemoteControlCommandData	The remote control command data that was received. This structure has the following format:

```
typedef struct
{
  AVRCP_Message_Type_t MessageType;
  union
  {
    AVRCP_Unit_Info_Command_Data_t
                UnitInfoCommandData;
    AVRCP_Subunit_Info_Command_Data_t
                SubunitInfoCommandData;
    AVRCP_Pass_Through_Command_Data_t
                PassThroughCommandData;
    AVRCP_Vendor_Dependent_Generic_Command_Data_t
                VendorDependentGenericCommandData;
    AVRCP_Browsing_Channel_Generic_Message_Data_t
                BrowsingChannelGenericMessageData;
    AVRCP_Group_Navigation_Command_Data_t
                GroupNavigationCommandData;
    AVRCP_Get_Capabilities_Command_Data_t
                GetCapabilitiesCommandData;
    AVRCP_List_Player_Application_Setting_Values_
                Command_Data_t
                ListPlayerApplicationSettingValuesCommandD
                ata;
    AVRCP_Get_Current_Player_Application_Setting_
                Value_Command_Data_t
                GetCurrentPlayerApplicationSetting
                ValueCommandData;
    AVRCP_Set_Player_Application_Setting_Value_
                Command_Data_t
                SetPlayerApplicationSettingValueCommandD
                ata;
    AVRCP_Get_Player_Application_Setting_Attribute_
                Text_Command_Data_t
```

```
                  GetPlayerApplicationSettingAttribute
                  TextCommandData;
            AVRCP_Get_Player_Application_Setting_Value_
                  Text_Command_Data_t
                  GetPlayerApplicationSettingValue
                  TextCommandData;
            AVRCP_Inform_Displayable_Character_Set_Command_
                  Data_t
                  InformDisplayableCharacterSet
                  CommandData;
            AVRCP_Inform_Battery_Status_Of_CT_Command_
                  Data_t
                  InformBatteryStatusOfCTCommandData;
            AVRCP_Get_Element_Attributes_Command_Data_t
                  GetElementAttributesCommandData;
            AVRCP_Request_Continuing_Response_Command_
                  Data_t
                  RequestContinuingResponseCommandData;
            AVRCP_Abort_Continuing_Response_Command_
                  Data_t
                  AbortContinuingResponseCommandData;
            AVRCP_Get_Play_Status_Response_Data_t
                  GetPlayStatusResponseData;
            AVRCP_Register_Notification_Command_Data_t
                  RegisterNotificationCommandData;
            AVRCP_Set_Absolute_Volume_Command_Data_t
                  SetAbsoluteVolumeCommandData;
            AVRCP_Set_Addressed_Player_Command_Data_t
                  SetAddressedPlayerCommandData;
            AVRCP_Play_Item_Command_Data_t
                  PlayItemCommandData;
            AVRCP_Add_To_Now_Playing_Command_Data_t
                  AddToNowPlayingCommandData;
            AVRCP_Set_Browsed_Player_Command_Data_t
                  SetBrowsedPlayerCommandData;
            AVRCP_Change_Path_Command_Data_t
                  ChangePathCommandData;
            AVRCP_Get_Item_Attributes_Command_Data_t
                  GetItemAttributesCommandData;
            AVRCP_Search_Command_Data_t
                  SearchCommandData;
            AVRCP_Search_Response_Data_t
                  SearchResponseData;
            AVRCP_Get_Folder_Items_Command_Data_t
                  GetFolderItemsCommandData;
         } MessageData;
      } AUD_Remote_Control_Command_Data_t;
```

etAUD_Remote_Control_Command_Confirmation

This event is dispatched when a remote control confirmation is received from a remote device (previously issued via the **AUD_Send_Remote_Control_Command()**).

Return Structure:

```
typedef struct
{
  BD_ADDR_t                 BD_ADDR;
  unsigned int              TransactionID;
  unsigned int              ConfirmationStatus;
  AUD_Remote_Control_Response_Data_t RemoteControlResponseData;
} AUD_Remote_Control_Command_Confirmation_Data_t;
```

EventParamerers:

BD_ADDR	Address of the remote Bluetooth device that the remote control command was received from.
TransactionID	AVRCP transaction ID of the original command (return value from the **AUD_Send_Remote_Control_Command()** function).
ConfirmationStatus	Confirmation Status of the remote control command. This value will be one of the following: AUD_REMOTE_CONTROL_COMMAND_CONFIRMATION_STATUS_SUCCESS AUD_REMOTE_CONTROL_COMMAND_CONFIRMATION_STATUS_TIMEOUT AUD_REMOTE_CONTROL_COMMAND_CONFIRMATION_STATUS_UNKNOWN_ERROR
RemoteControlResponseData	The remote control response information. This parameter has the following format:

```
typedef struct
{
  AVRCP_Message_Type_t MessageType;
  union
  {
    AVRCP_Unit_Info_Response_Data_t
                UnitInfoResponseData;
    AVRCP_Subunit_Info_Response_Data_t
                SubunitInfoResponseData;
    AVRCP_Pass_Through_Response_Data_t
                PassThroughResponseData;
    AVRCP_Vendor_Dependent_Generic_Response_Data_t
                VendorDependentGenericResponseData;
    AVRCP_Group_Navigation_Response_Data_t
                GroupNavigationResponseData;
    AVRCP_Get_Capabilities_Response_Data_t
                GetCapabilitiesResponseData;
    AVRCP_List_Player_Application_Setting_Attributes_
                Response_Data_t
```

ListPlayerApplicationSettingAttributes
ResponseData;
AVRCP_List_Player_Application_Setting_Values_
Response_Data_t
ListPlayerApplicationSettingValues
ResponseData;
AVRCP_Get_Current_Player_Application_Setting_
Value_Response_Data_t
GetCurrentPlayerApplicationSetting
ValueResponseData;
AVRCP_Set_Player_Application_Setting_Value_
Response_Data_t
SetPlayerApplicationSettingValue
ResponseData;
AVRCP_Get_Player_Application_Setting_Attribute_
Text_Response_Data_t
GetPlayerApplicationSettingAttribute
TextResponseData;
AVRCP_Get_Player_Application_Setting_Value_Text_
Response_Data_t
GetPlayerApplicationSettingValueText
ResponseData;
AVRCP_Inform_Displayable_Character_Set_Response_
Data_t
InformDisplayableCharacterSetResponse
Data;
AVRCP_Inform_Battery_Status_Of_CT_Response_
Data_t
InformBatteryStatusOfCTResponseData;
AVRCP_Get_Element_Attributes_Response_Data_t
GetElementAttributesResponseData;
AVRCP_Get_Play_Status_Response_Data_t
GetPlayStatusResponseData;
AVRCP_Register_Notification_Response_Data_t
RegisterNotificationResponseData;
AVRCP_Abort_Continuing_Response_Response_Data_t
AbortContinuingResponseResponseData;
AVRCP_Set_Absolute_Volume_Response_Data_t
SetAbsoluteVolumeResponseData;
AVRCP_Set_Addressed_Player_Response_Data_t
SetAddressedPlayerResponseData;
AVRCP_Play_Item_Response_Data_t
PlayItemResponseData;
AVRCP_Add_To_Now_Playing_Response_Data_t
AddToNowPlayingResponseData;
AVRCP_Command_Reject_Response_Data_t
CommandRejectResponseData;
AVRCP_Set_Browsed_Player_Response_Data_t
SetBrowsedPlayerResponseData;
AVRCP_Change_Path_Response_Data_t
ChangePathResponseData;

```
      AVRCP_Get_Item_Attributes_Response_Data_t
                  GetItemAttributesResponseData;
      AVRCP_Search_Response_Data_t
                  SearchResponseDat;
      AVRCP_Get_Folder_Items_Response_Data_t
                  GetFolderItemsResponseData;
      AVRCP_General_Reject_Response_Data_t
                  GeneralRejectResponseData;
   } MessageData;
} AUD_Remote_Control_Response_Data_t;
```

etAUD_Remote_Control_Open_Indication

This event is dispatched when a remote audio device connects to the specified local audio stream.

Return Structure:

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

Event Parameters:

BD_ADDR	Address of the Bluetooth device making the connection.

etAUD_Remote_Control_Open_Confirmation

This event is dispatched to indicate the result (success or failure) of a previously submitted connection request via the **AUD_Open_Remote_Control**() function.

Return Structure:

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

Event Parameters:

BD_ADDR	Address of the remote device that was connected.
OpenStatus	Specifies the connection status of the connection attempt. This will be one of the following values: AUD_REMOTE_CONTROL_OPEN_CONFIRMATION_STATUS_SUCCESS AUD_REMOTE_CONTROL_OPEN_CONFIRMATION_STATUS_CONNECTION_TIMEOUT AUD_REMOTE_CONTROL_OPEN_CONFIRMATION_STATUS_CONNECTION_REFUSED AUD_REMOTE_CONTROL_OPEN_CONFIRMATION_STATUS_UNKNOWN_ERROR

etAUD_Remote_Control_Close_Indication

This event is dispatched when a remote control disconnection occurs.

Return Structure:

```
typedef struct
{
  BD_ADDR_t                BD_ADDR;
  AUD_Disconnect_Reason_t  DisconnectReason;
} AUD_Remote_Control_Close_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Address of the remote device that disconnected.
DisconnectReason	Reason for the disconnection. This value will be one of the following: adrRemoteDeviceDisconnect adrRemoteDeviceLinkLoss adrRemoteDeviceTimeout

etAUD_Signalling_Channel_Open_Indication

This event is dispatched when the AVDTP/GAVD signalling channel is established with a remote device.

Return Structure:

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

EventParamerers:

BD_ADDR	The Bluetooth address of the remote device which has connected the AVDTP/GAVD signalling channel to the local device.

etAUD_Signalling_Channel_Close_Indication

This event is dispatched when the AVDTP/GAVD signalling channel is disconnected from a remote device.

Return Structure:

```
typedef struct
{
  BD_ADDR_t                BD_ADDR;
  AUD_Disconnect_Reason_t  DisconnectReason;
} AUD_Signalling_Channel_Close_Indication_Data_t;
```

EventParamerers:

BD_ADDR	The Bluetooth address of the remote device which has disconnected the AVDTP/GAVD signalling channel from the local device.

DisconnectReason	Reason for the disconnection. This value will be one of the following: adrRemoteDeviceDisconnect adrRemoteDeviceLinkLoss adrRemoteDeviceTimeout

etAUD_Browsing_Channel_Open_Indication

This event is dispatched when a remote Browsing service gets connected to the local Browsing service.

Return Structure:

```
typedef struct
{
  BD_ADDR_t    BD_ADDR;
  Word_t       MTU;
} AUD_Browsing_Channel_Open_Indication_Data_t;
```

EventParamerers:

BD_ADDR	The Bluetooth address of the remote device that is connected.
MTU	Specifies the maximum allowable packet data payload that can be sent (Maximum Transmission Unit).

etAUD_Browsing_Channel_Open_Confirmation

This event is dispatched to the application when an attempt to connect to a remote AUD Browsing Channel is complete.

Return Structure:

```
typedef struct
{
  BD_ADDR_t    BD_ADDR;
  unsigned int OpenStatus;
  Word_t       MTU;
} AUD_Browsing_Channel_Open_Confirmation_Data_t;
```

EventParamerers:

BD_ADDR	The Bluetooth address of the remote device that the connection was attempted.
OpenStatus	Result of the connection zero if successful or a negative error code in case of failure.
MTU	Specifies the maximum allowable packet data payload that can be sent (Maximum Transmission Unit).

etAUD_Browsing_Channel_Close_Indication

This event is dispatched to the application when the remote device disconnects the Browsing Channel from the Local service.

Return Structure:

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

EventParamerers:

BD_ADDR	The Bluetooth address of the Remote Device that has disconnected the Browsing Channel.

3. File Distributions

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

File	Contents/Description
AUDAPI.h	Bluetooth Audio profile sub-system API definitions
SS1BTAUD.h	Bluetooth Audio profile sub-system Include file