Audio/Video Control Transport Protocol (AVCTP)

Application Programming Interface Reference Manual

Profile Version: 1.4

Release: 4.0.1
January 10, 2014

Louisville, KY www.stonestreetone.com

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

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/Video Control Protocol provided by Bluetopia. Chapter 2 contains a description of the programming interfaces for this. Chapter 3 contains the header file name list for the Bluetooth AVCTP. This is extensively used in the programming of upper profile layer like AVRCP.

1.1 Scope

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

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

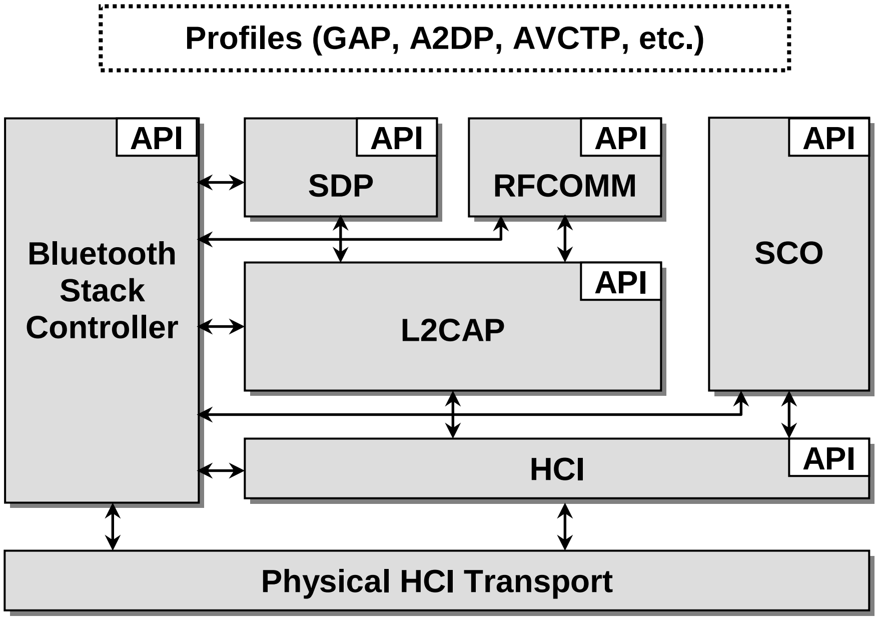

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

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, Volumes 0-4*, version 2.1 + EDR, July 26, 2007.
2. *Specification of the Bluetooth System, Bluetooth Core Specification Addendum 1,* June 26, 2008.
3. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 3.0+HS, April 21, 2009.
4. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 3.0+HS, April 21, 2009.
5. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 3.0+HS, April 21, 2009.
6. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 3.0+HS, April 21, 2009.
7. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 3.0+HS, April 21, 2009.
8. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 3.0+HS, April 21, 2009.
9. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 4.0, June 30, 2010.
10. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 4.0, June 30, 2010.
11. *Specification of the Bluetooth System, Volume 2, Core System Package [BR/EDR Controller Volume]*, version 4.0, June 30, 2010.
12. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 4.0, June 30, 2010.
13. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 4.0, June 30, 2010.
14. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 4.0, June 30, 2010.
15. *Specification of the Bluetooth System, Volume 6, Core System Package [Low Energy Controller Volume]*, version 4.0, June 30, 2010.
16. *Bluetooth Assigned Numbers*, Bluetooth.org, version 2.25, May 24th, 2004.
17. *Audio/Video Control Transport Protocol Specification*, version 1.3, June 26, 2008.
18. *Audio/Video Remote Control Profile*, version 1.4, June 26, 2008.

19. *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
AVRCP	Audio/Video Remote Control Profile (Bluetooth Profile)
API	Application Programming Interface
BD_ADDR	Bluetooth Device Address
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Data Rate
GAP	Generic Access Profile (Bluetooth Profile)
AVCTP	Audio/Video Control Transport Protocol
HS	High Speed
LE	Low Energy
LSB	Least Significant Bit
MSB	Most Significant Bit
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. AVCTP Programming Interface

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

2.1 AVCTP Commands

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

Function	Description
AVCTP_Initialize	This function is responsible for initializing the Audio/Video Control Transport Protocol.
AVCTP_Cleanup	The following function is responsible for cleaning up a previously initialized AVCTP instance.
AVCTP_Enable_Browsing_Channel_Support	This function is responsible for instructing the AVCTP that it is to support the Browsing Channel.
AVCTP_Connect_Request_Response	This function is responsible for responding to an individual request to connect to a local AVCTP server.
AVCTP_Register_Profile	This function will register a Local Profile so that remote Profiles/Applications can connect to this.
AVCTP_UnRegister_Profile	This function will un-register a Local Profile.
AVCTP_Register_Profile_SDP_Record	This function adds a Profile Role SDP Service Record to the SDP Database.
AVCTP_UnRegister_Profile_SDP_Record	This is a utility MACRO that deletes a registered Profile SDP Service Record from the SDP Database.
AVCTP_Connect_Device	This function is responsible for initiating a connection to a remote device.
AVTCP_Connect_Browsing_Channel	This function is responsible for initiating a Browsing Channel connection to a remote device.
AVCTP_Close_Connection	This function is responsible for disconnecting a connection to a remote device.
AVCTP_Close_Browsing_Channel	This function is responsible for disconnecting any connected Browsing Channel to the specified remote device.

AVCTP_Send_Message	This function is used by a profile to send a message to a remote profile.
AVCTP_Send_Browsing_Channel_Message	This function is used by a profile to send a message to a remote profile over an established Browsing Channel.
AVCTP_Get_Profile_Server_Connection_Mode	This function is responsible for retrieving the current AVCTP Server Connection Mode.
AVCTP_Set_Profile_Server_Connection_Mode	This function is responsible for setting the AVCTP Server Connection Mode.

AVCTP_Initialize

This function is responsible for initializing the Audio/Video Control Transport Protocol. This function must be called before any other Profile may use this protocol. This function can only be called once per Bluetooth Stack Instance.

Prototype:

int BTPSAPI **AVCTP_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:

BTAVCTP_ERROR_INSUFFICIENT_RESOURCES
BTAVCTP_ERROR_CONTEXT_ALREADY_EXISTS
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_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.

AVCTP_Cleanup

The following function is responsible for cleaning up a previously initialized AVCTP instance.

Note:

1. This function does not delete any SDP Service Record Handles (i.e., added via a call to the AVCTP_Register_Profile_SDP_Record() function).

Prototype:

int BTPSAPI **AVCTP_Cleanup**(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:

BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID

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.

AVCTP_Enable_Browsing_Channel_Support

This function is responsible for instructing the AVCTP module that it is to support the Browsing Channel. This function must be called after a successful call to AVCTP_Initialize() and before any profiles are registered.

Notes:

1. Once the Browsing Channel is enabled it cannot be disabled.

Prototype:

int BTPSAPI **AVCTP_Enable_Browsing_Channel_Support** (
    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:

BTAVCTP_ERROR_PROFILES_REGISTERED
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_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.

AVCTP_Connect_Request_Response

This function is responsible for responding to an individual request to connect to a local AVCTP Server. This function should be called in response to the receipt of an etAVCTP_Connect_Request_Indication event.

Notes:

1. The connection to the server is not established until an etAVCTP_Connect_Indication event has occurred.
2. The etAVCTP_Connect_Request_Indication event will only be dispatched if the server mode was explicitly set to asmManualAccept via the AVCTP_Set_Profile_Server_Connection_Mode() function. If this mode is set, ONLY the callback that was specified with the AVCTP_Initialize() function will receive this event.

Prototype:

int BTPSAPI **AVCTP_Connect_Request_Response**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, Boolean_t AcceptConnection)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
BD_ADDR	Bluetooth Device Address of the AVCTP connection for which a connection request was received.
AcceptConnection	Specifies whether to accept the pending connection request or reject the request.

Return:

Zero if successful.

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

BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_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.

AVCTP_Register_Profile

This function will register a Local Profile so that remote Profiles/Applications can connect to it.

Prototype:

int BTPSAPI **AVCTP_Register_Profile**(unsigned int BluetoothStackID, UUID_16_t ProfileUUID, AVCTP_Event_Callback_t EventCallback, unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ProfileUUID	UUID of the Profile that is using the AVCTP transport (can be the AVRCP UUID).
EventCallback	Function that is called whenever any event occurs on this profile.
CallbackParameter	A user-defined parameter (e.g. a tag value) that will be passed back to the user in the callback function with each event callback.

Return:

A Positive Profile Identifier if successful.

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

BTAVCTP_ERROR_INSUFFICIENT_RESOURCES
BTAVCTP_ERROR_AVCTP_CONNECTED
BTAVCTP_ERROR_PROFILE_ALREADY_REGISTERED
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_ERROR_INVALID_PARAMETER

Possible Events:

etAVCTP_Connect_Indication
etAVCTP_Connect_Confirmation
etAVCTP_Disconnect_Indication
etAVCTP_Message_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.

AVCTP_UnRegister_Profile

The following function is responsible for unregistering a profile from a particular Bluetooth Stack. The stack will respond with invalid-profile for any attempts by a remote device to connect to this profile after it is unregistered.

Prototype:

int BTPSAPI **AVCTP_UnRegister_Profile**(unsigned int BluetoothStackID, unsigned int AVCTPProfileID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile to be unregistered.

Return:

Zero if successful.

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

BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_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.

AVCTP_Register_ Profile_SDP_Record

This function adds an AVCTP Profile Service Record to the SDP database.

Notes:

1. The Service Record Handle that is returned from this function will remain in the SDP Record Database until it is deleted by calling the SDP_Delete_Service_Record() function. A macro is provided to delete the Service Record from the SDP Database. This macro maps the AVCTP_Un_Register_SDP_Record() to SDP_Delete_Service_Record(), and is defined as follows:

   AVCTP_UnRegister_Profile_SDP_Record (__BluetoothStackID, __SDPRecordHandle) (SDP_Delete_Service_Record(__BluetoothStackID, __SDPRecordHandle))

2. Any Protocol Information that is specified will be added in the protocol attribute after the default protocol list of L2CAP and AVCTP.

3. 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.
4. At least one Service Class (UUID) must be specified in the SDP Service Record structure.
5. The ProtocolList and ProfileList members of the SDP Service Record structure are optional (if specified as NULL). The Protocol List information must be a Data Element Sequence, and the information contained in this sequence is added after the AVCTP and L2CAP Protocol information. The ProfileList must also be a Data Element Sequence, however this information is added as-is (nothing is added other than this information).

Prototype:

int BTPSAPI **AVCTP_Register_Profile_SDP_Record**(unsigned int BluetoothStackID, AVCTP_SDP_Service_Record_t *SDPServiceRecord, char *ServiceName, char *ProviderName, DWord_t *SDPServiceRecordHandle)

Parameters:

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

SDPServiceRecord Specifies additional SDP information to add to the record. This is defined by the following structure:

```
typedef struct
{
   unsigned int          NumberServiceClassUUID;
   SDP_UUID_Entry_t   *SDPUUIDEntries;
   SDP_Data_Element_t *ProtocolList;
   SDP_Data_Element_t *ProfileList;
} AVCTP_SDP_Service_Record_t;
```

ServiceName Name to appear in the SDP Database for this service.

Provider Name Name of the provider 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:

BTFTP_ERROR_NOT_INITIALIZED
BTFTP_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.

AVCTP_Connect_Device

This function is responsible for initiating a connection to a remote device. It will try to establish an L2CAP channel if no channel exists to the remote device.

Prototype:

int BTPSAPI **AVCTP_Connect_Device**(unsigned int BluetoothStackID, unsigned int AVCTPProfileID, BD_ADDR_t RemoteBD_ADDR)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile that initiates the connection. This is the ID that was returned when this profile was registered.
RemoteBD_ADDR	Address of the Bluetooth device to connect with.

Return:

Zero if successful.

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

BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID D
BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_ERROR_PROFILE_BUSY
BTAVCTP_AWAITING_DISCONNECTION
BTAVCTP_ERROR_ALREADY_CONNECTED
BTAVCTP_ERROR_ALREADY_CONNECTING
BTAVCTP_ERROR_INSUFFICIENT_RESOURCES

Possible Events:

etAVCTP_Connect_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.

AVCTP_Connect_Browsing_Channel

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

Notes:

1. A Browsing Channel can ONLY be added if there already exists an on-going AVCTP connection between the local device and the remote device already.
2. The Browsing Channel cannot exist without a corresponding AVCTP connection. This means that if the AVCTP connection is terminated, the Browsing Channel connection will be terminated as well.

Prototype:

int BTPSAPI **AVCTP_Connect_Browsing_Channel** (unsigned int BluetoothStackID, unsigned int AVCTPProfileID, BD_ADDR_t RemoteBD_ADDR, Word_t MTUSize)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile that initiates the connection. This is the ID that was returned when this profile was registered.
RemoteBD_ADDR	Address of the Bluetooth device to connect with.
MTUSize	Specifies the MTU (Maximum Transmission Unit) size to use for the Browsing Channel connection.

Return:

Zero if successful.

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

BTAVCTP_ERROR_INSUFFICIENT_RESOURCES
BTAVCTP_ERROR_AWAITING_DISCONNECTION
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_CONNECTION_NOT_INITIATED
BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_ERROR_ALREADY_CONNECTED
BTAVCTP_ERROR_ALREADY_CONNECTING

Possible Events:

etAVCTP_Browsing_Channel_Connect_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.

AVCTP_Close_Connection

This function is responsible for disconnecting a connection to a remote device. The L2CAP channel is disconnected only if this profile has initiated this connection.

Prototype:

int BTPSAPI **AVCTP_Close_Connection**(unsigned int BluetoothStackID, unsigned int AVCTPProfileID, BD_ADDR_t RemoteBD_ADDR)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile wishes to disconnect the connection. This is the ID that was returned when this profile was registered.
RemoteBD_ADDR	The Bluetooth Device Address of the remote device to disconnect.

Return:

Zero if successful.

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

BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_IDD
BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_ERROR_CONNECTION_NOT_INITIATED
BTAVCTP_ERROR_INVALID_CONNECTION

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.

AVCTP_Close_Browsing_Channel

This function is responsible for disconnecting any connected Browsing Channel connection to a remote device. The L2CAP channel is disconnected only if this profile has initiated this connection.

Prototype:

int BTPSAPI **AVCTP_Close_Browsing_Channel** (unsigned int BluetoothStackID, unsigned int AVCTPProfileID, BD_ADDR_t RemoteBD_ADDR)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile wishes to disconnect the connection. This is the ID that was returned when this profile was registered.
RemoteBD_ADDR	The Bluetooth Device Address of the remote device to disconnect.

Return:

Zero if successful.

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

BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_ERROR_CONNECTION_NOT_INITIATED
BTAVCTP_ERROR_INVALID_CONNECTION

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.

AVCTP_Send_Message

This function is used by a profile to send a message to a remote profile.

Prototype:

int BTPSAPI **AVCTP_Send_Message**(unsigned int BluetoothStackID, unsigned int AVCTPProfileID, BD_ADDR_t RemoteBD_ADDR, Byte_t TransactionID, Boolean_t ResponseMessage, unsigned int DataLength, Byte_t *DataBuffer)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile wishes to send data to the remote device. This is the ID that was returned when this profile was registered.
RemoteBD_ADDR	Address of the Bluetooth device to send the message to.

TransactionID	A number (1-15) that identifies this transaction.
ResponseMessage	Flag indicating if this is a response message or not.
DataLength	Specifies the length of the payload. This parameter specifies the length (in bytes) of the payload data that is to be written.
DataBuffer	Points to the payload data. This parameter is a pointer to the payload data to be written to the specified stream endpoint. This pointer must point to at least the number of bytes specified by the DataLength parameter.

Return:

Zero if successful.

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

BTAVCTP_ERROR_INSUFFICIENT_RESOURCES
BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_INVALID_CONNECTION
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_MESSAGE_TOO_LONG

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.

AVCTP_Send_Browsing_Channel_Message

This function is used by a profile to send a message to a remote profile over an established Browsing Channel.

Prototype:

int BTPSAPI **AVCTP_Send_Browsing_Channel_Message** (
unsigned int BluetoothStackID, unsigned int AVCTPProfileID,
BD_ADDR_t RemoteBD_ADDR, Byte_t TransactionID, Boolean_t ResponseMessage,
unsigned int DataLength, Byte_t *DataBuffer)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
AVCTPProfileID	The ID of the profile wishes to send data to the remote device. This is the ID that was returned when this profile was registered.
RemoteBD_ADDR	Address of the Bluetooth device to send the message to.
TransactionID	A number (1-15) that identifies this transaction.

ResponseMessage	Flag indicating if this is a response message or not.
DataLength	Specifies the length of the payload. This parameter specifies the length (in bytes) of the payload data that is to be written.
DataBuffer	Points to the payload data. This parameter is a pointer to the payload data to be written to the specified stream endpoint. This pointer must point to at least the number of bytes specified by the DataLength parameter.

Return:

Zero if successful.

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

BTAVCTP_ERROR_INSUFFICIENT_RESOURCES
BTAVCTP_ERROR_PROFILE_NOT_FOUND
BTAVCTP_ERROR_INVALID_CONNECTION
BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_ERROR_INVALID_PARAMETER
BTAVCTP_ERROR_BROWSING_CHANNEL_MTU_EXCEEDED

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.

AVCTP_Get_Profile_Server_Connection_Mode

This function is responsible for retrieving the current AVCTP Server Connection Mode.

Notes:

1. The default Server Connection Mode is asmAutomaticAccept.
2. This function is used for AVCTP Servers which use Bluetooth Security Mode 2.

Prototype:

int BTPSAPI **AVCTP_Get_Profile_Server_Connection_Mode** (
unsigned int BluetoothStackID,
AVCTP_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:

BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_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.

AVCTP_Set_Profile_Server_Connection_Mode

This function is responsible for setting the AVCTP Server Connection Mode.

Notes:

1. The default Server Connection Mode is asmAutomaticAccept.
2. This function is used for AVCTP Servers which use Bluetooth Security Mode 2.

Prototype:

int BTPSAPI **AVCTP_Set_Profile_Server_Connection_Mode** (
unsigned int BluetoothStackID,
AVCTP_Server_Connection_Mode_t ServerConnectionMode,
AVCTP_Event_Callback_t EventCallback, unsigned long CallbackParameter)

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.
EventCallback	An AVCTP Event Callback which will receive notifications of a Bluetooth Connection Request.
CallbackParameter	A user-defined parameter (e.g. a tag value) that will be passed back to the user in the callback function with each event callback.

Return:

Zero if successful.

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

BTAVCTP_ERROR_NOT_INITIALIZED
BTAVCTP_ERROR_INVALID_BLUETOOTH_STACK_ID
BTAVCTP_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 AVCTP Events

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

Event	Description
etAVCTP_Connect_Indication	A remote device has connected to local AVCTP instance.
etAVCTP_Connect_Confirmation	Confirms that the connection attempt to a remote profile that was initiated by a local profile has ended and informs if it was successful or not.
etAVCTP_Disconnect_Indication	A remote device that was connected to a local AVCTP profile has disconnected.
etAVCTP_Message_Indication	A local registered and connected profile has received a message/response from a remote device.
etAVCTP_Connect_Request_Indication	A remote service is requesting a connection to the local service.
etAVCTP_Browsing_Channel_Connect_Indication	A remote Browsing service has connected to the local Browsing service.
etAVCTP_Browsing_Channel_Connect_confirmation	A previously outstanding attempt to connect to a remote AVCTP Browsing Channel is complete.
etAVCTP_Browsing_Channel_Disconnect_Indication	A remote device has disconnected from the Browsing Channel of the local service.
etAVCTP_Browsing_Channel_Message_Indication	The local Browsing Service received data from a remote Browsing Service that is connected to the local device.

etAVCTP_Connect_Indication

A remote device has connected to a local AVCTP instance.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
} AVCTP_Connect_Indication_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the Remote device that connected to the local AVCTP server.

etAVCTP_Connect_Confirmation

Confirms that the connection attempt to a remote profile that was initiated by a local profile has ended and informs if it was successful or not.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
    int             Status;
    int             Result;
} AVCTP_Connect_Confirmation_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the Remote device to which the connection was attempted.
Status	AVCTP_OPEN_STATUS_SUCCESS AVCTP_OPEN_STATUS_CONNECTION_TIMEOUT AVCTP_OPEN_STATUS_CONNECTION_REFUSED AVCTP_OPEN_STATUS_UNKNOWN_ERROR
Result	Result code returned by this operation.

etAVCTP_Disconnect_Indication

A remote device has disconnected from a local AVCTP instance.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
} AVCTP_Disconnect_Indication_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the Remote device that disconnected from a local AVCTP instance.

etAVCTP_Message_Indication

A local registered and connected profile has received a message/response from a remote device.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
    Byte_t          MessageType;
    Byte_t          TransactionID;
    BOOLEAN         InvalidProfileID;
    unsigned int    DataLength;
    Byte_t          *DataBuffer;
} AVCTP_Message_Indication_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the Remote device to which the connection was attempted.
MessageType	Type of message. This is set for response and not set for a command message.
TransactionID	Identifier for this transaction. Used by the application if needed.
InvalidProfileID	InvalidProfileID is set if the local application tried to send a message to a profile that was not registered with remote AVCTP. This would result in local AVCTP receiving a message with the IPID bit set. This will be conveyed to the local application through InvalidProfileID. If InvalidProfileID is set to TRUE, Datalength will be zero and Databuffer will point to NULL.

DataLength	Specifies the length of the received data. This value represents the size (in bytes) of the data that is pointed to by the DataBuffer member.
DataBuffer	Pointer to the incoming Data. Use only if DataLength is positive.

etAVCTP_Connect_Request_Indication

A remote service is requesting a connection to the local service.

Return Structure:

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

Event Parameters:

BD_ADDR	Bluetooth Address of the Remote device that is requesting a connection to the local AVCTP server.

etAVCTP_Browsing_Channel_Connect_Indication

A remote Browsing service has connected to the local Browsing service.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
    Word_t          MTU;
} AVCTP_Browsing_Channel_Connect_Indication_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the Remote device that connected to the local AVCTP server Browsing Channel.
MTU	Maximum Transmission Unit specified for this Browsing Channel.

etAVCTP_Browsing_Channel_Connect_Confirmation

A previously outstanding attempt to connect to a remote AVCTP Browsing Channel is complete.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
    int             Status;
    int             Result;
    Word_t          MTU;
} AVCTP_Browsing_Channel_Connect_Confirmation_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the remote device that was connected to by the local AVCTP server Browsing Channel.
Status	AVCTP_OPEN_STATUS_SUCCESS AVCTP_OPEN_STATUS_CONNECTION_TIMEOUT AVCTP_OPEN_STATUS_CONNECTION_REFUSED AVCTP_OPEN_STATUS_UNKNOWN_ERROR
Result	Result code returned by this operation.
MTU	Maximum Transmission Unit specified for this Browsing Channel.

etAVCTP_Browsing_Channel_Disconnect_Indication

A remote device has disconnected from the Browsing Channel of the local service.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
} AVCTP_Browsing_Channel_Disconnect_Indication_Data_t;
```

Event Parameters:

AVCTPProfileID	The Profile ID of the AVCTP profile receiving this event.
BD_ADDR	Bluetooth Address of the Remote device that disconnected from a local AVCTP instance Browsing Channel.

etAVCTP_Browsing_Channel_Message_Indication

The local Browsing Service received data from a remote Browsing Service that is connected to the local device.

Return Structure:

```
typedef struct
{
    unsigned int    AVCTPProfileID;
    BD_ADDR_t       BD_ADDR;
    Byte_t          MessageType;
    Byte_t          TransactionID;
    Boolean_t       InvalidProfileID;
    unsigned int    DataLength;
    Byte_t          *DataBuffer;
} AVCTP_Browsing_Channel_Message_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Bluetooth Address of the Remote device which has sent the incoming message.
MessageType	Type of message. This is set for response and not set for a command message.
TransactionID	Identifier for this transaction. Used by the application if needed.
InvalidProfileID	InvalidProfileID is set if the local application tried to send a message to a profile that was not registered with remote AVCTP. This would result in local AVCTP receiving a message with the IPID bit set. This will be conveyed to the local application through InvalidProfileID. If InvalidProfileID is set to TRUE, Datalength will be zero and Databuffer will point to NULL.
DataLength	Specifies the length of the received data. This value represents the size (in bytes) of the data that is pointed to by the DataBuffer member.
DataBuffer	Pointer to the incoming Data. Use only if DataLength is positive.

3. File Distributions

The header files that are distributed with the Bluetooth AVCTP are listed in the table below.

File	Contents/Description
AVCTPAPI.h	Bluetooth AVCTP API definitions
SS1BTAVC.h	Bluetooth AVCTP Include file
AVCTypes.h	Bluetooth AVCTP type definitions