Generic Attribute Profile (GATT)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
January 10, 2014

Louisville, KY www.stonestreetone.com

Table of Contents

1. Introduction

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.

The remainder of this chapter has sections on the scope of this document, other documents applicable to this document, and a listing of acronyms and abbreviations. Chapter 2 is the API reference that contains a description of all programming interfaces for the Bluetooth Generic Attribute (GATT) Profile provided by Bluetopia. Chapter 3 contains the header file name list for the Bluetooth Generic Attribute Profile library.

1.1 Scope

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

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

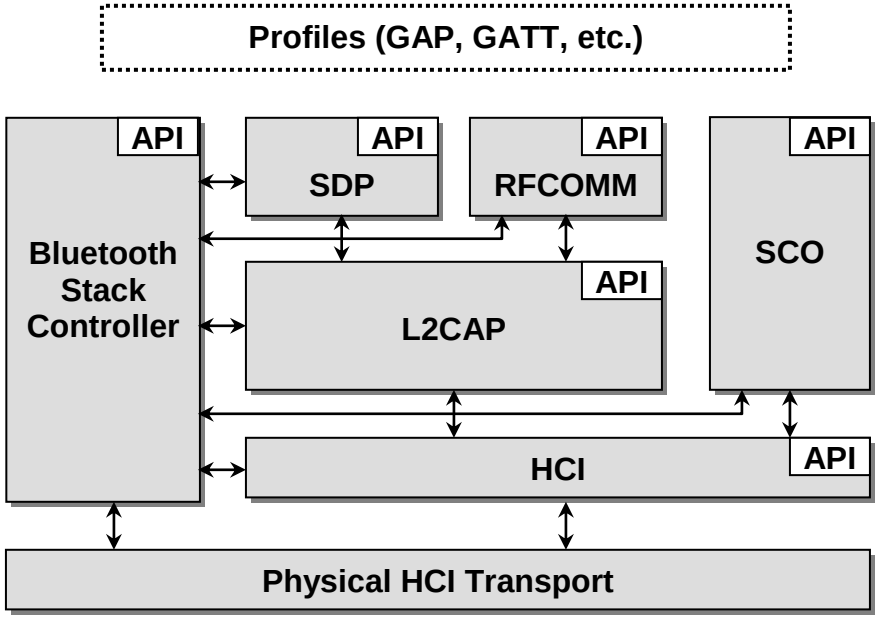

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

1.2 Applicable Documents

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

1. *Specification of the Bluetooth System, Volume 1, Core*, version 1.1, February 22, 2001.
2. *Specification of the Bluetooth System, Volume 2, Profiles*, version 1.1, February 22, 2001.
3. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 2.0 + EDR, November 4, 2004.
4. *Specification of the Bluetooth System, Volume 2, Core System Package*, version 2.0 + EDR, November 4, 2004.
5. *Specification of the Bluetooth System, Volume 3, Core System Package*, version 2.0 + EDR, November 4, 2004.
6. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 2.1+EDR, July 26, 2007.
7. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 2.1+EDR, July 26, 2007.
8. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 2.1+EDR, July 26, 2007.
9. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 2.1+EDR, July 26, 2007.
10. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 2.1+EDR, July 26, 2007.
11. *Specification of the Bluetooth System, Bluetooth Core Specification Addendum 1*, June 26, 2008.
12. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 3.0+HS, April 21, 2009.
13. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 3.0+HS, April 21, 2009.
14. *Specification of the Bluetooth System, Volume 2, Core System Package [Controller Volume]*, version 3.0+HS, April 21, 2009.
15. *Specification of the Bluetooth System, Volume 3, Core System Package [Host Volume]*, version 3.0+HS, April 21, 2009.
16. *Specification of the Bluetooth System, Volume 4, Host Controller Interface [Transport Layer]*, version 3.0+HS, April 21, 2009.
17. *Specification of the Bluetooth System, Volume 5, Core System Package [AMP Controller Volume]*, version 3.0+HS, April 21, 2009.

18. *Specification of the Bluetooth System, Volume 0, Master Table of Contents & Compliance Requirements*, version 4.0, June 30, 2010.

19. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 4.0, June 30, 2010.

20. *Specification of the Bluetooth System, Volume 2, Core System Package [BR/EDR Controller Volume]*, version 4.0, June 30, 2010.

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

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

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

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

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

26. *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.

27. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual,* version 4.0.1, January 10, 2013.

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

1.3 Acronyms and Abbreviations

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

Term	Meaning
API	Application Programming Interface
ATT	Attribute Protocol
BD_ADDR	Bluetooth Device Address
BR	Basic Rate
BT	Bluetooth
EDR	Enhanced Data Rate
GATT	Generic Attribute Profile
HCI	Host Controller Interface

Term	Meaning
HS	High Speed
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy
RFCOMM	Radio Frequency serial COMMunications – Serial cable emulation protocol based on ETSI TS 07.10
SCO link	Synchronous Connection-Oriented Link – Supports time-bounded information like voice. (Master to single slave)
SDP	Service Discovery Protocol
SPP	Serial Port Protocol

2. Generic Attribute Profile Programming Interfaces

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

2.1 Generic Attribute Profile Commands

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

Function	Description
GATT_Initialize	This function is responsible for initializing the Generic Attribute Profile module.
GATT_Cleanup	This function is responsible for cleaning up a previously initialized Generic Attribute Profile module.
GATT_Register_Connection_Events	Register a connection event callback to receive connection status information.
GATT_Un_Register_Connection_Events	Un-register a previously registered connection event callback.
GATT_Connection_Request_Response	Respond to an incoming BR/EDR GATT connection request.
GATT_Connect	Connect to a remote BR/EDR GATT device.
GATT_Disconnect	Disconnect from a currently connected BR/EDR GATT device.
GATT_Get_Incoming_Connection_Mode	Query the current BR/EDR GATT incoming connection mode.
GATT_Set_Incoming_Connection_Mode	Configure the current BR/EDR GATT incoming connection mode.
GATT_Register_SDP_Record	Register a generic GATT SDP Record.
GATT_Register_Service	Registers a GATT service with the local GATT database.
GATT_Un_Register_Service	Un-register a previously registered GATT service from the local GATT database.
GATT_Register_Service_SDP_Record	Registers a SDP Record for a GATT Service that

	supports BR/EDR.
GATT_Read_Response	Respond with a successful response to a received GATT read request.
GATT_Write_Response	Respond with a successful response to a received GATT write request.
GATT_Execute_Write_Response	Respond with a successful response to a received GATT execute write request.
GATT_Error_Response	Respond with an error response to received GATT request.
GATT_Handle_Value_Indication	Send a handle/value indication to a connected GATT client.
GATT_Handle_Value_Notification	Send a handle/value notification to a connected GATT client.
GATT_Verify_Signature	Verify signed write request that was received from a remote GATT client.
GATT_Service_Changed_Read_Response	Used to respond to a Service Changed read request.
GATT_Service_Changed_CCCD_Read_Response	Used to respond to a Service Changed CCCD read request.
GATT_Service_Changed_Indication	Used to send a Service Changed indication.
GATT_Exchange_MTU_Request	Request a change in the ATT MTU for a connection to a remote LE GATT server.
GATT_Discover_Services	Discover primary services on a remote, connected, GATT server.
GATT_Discover_Services_By_UUID	Discover services with a specific UUID on a remote, connected, GATT server.
GATT_Discover_Included_Services	Discover all included services on a remote, connected, GATT server.
GATT_Discover_Characteristics	Discover characteristics on a remote, connected, GATT server.
GATT_Discover_Characteristic_Descriptors	Discover characteristic descriptors on a remote, connected, GATT server.
GATT_Read_Value_Request	Read a value from a remote, connected, GATT server.
GATT_Read_Long_Value_Request	Read a long value from a remote, connected, GATT server.
GATT_Read_Value_By_UUID_Request	Read a value with a specific UUID from a remote, connected, GATT server.

GATT_Read_Multiple_Values_Request	Read multiple values from a remote, connected, GATT server.
GATT_Write_Request	Write a value to a remote, connected, GATT server (and wait for a response).
GATT_Write_Without_Response_ Request	Write a value to a remote, connected, GATT server (and do not wait for (or request) a response).
GATT_Signed_Write_Without_ Response	Write a value (with specified signing information) to a remote, connected, GATT server (and do not wait for (or request) a response).
GATT_Prepare_Write_Request	Prepare a write operation of one (or more) values to be written atomically to a remote, connected, GATT server.
GATT_Execute_Write_Request	Execute/commit a previously prepared write operation on a remote, connected, GATT server.
GATT_Handle_Value_Confirmation	Send a handle/value confirmation response to a remote, connected, GATT server.
GATT_Start_Service_Discovery	Used to start a service discovery operation that will discover services and information about the discovered services.
GATT_Start_Service_Discovery_Handle_ Range	Used to start a service discovery operation at a specific handle range of the remote GATT database that will discover services and information about the discovered services.
GATT_Stop_Service_Discovery	Used to stop a service discovery operation that was previously started with GATT_Start_Service_Discovery() or GATT_Start_Service_Discovery_Handle_Range () API.
GATT_Cancel_Transaction	Attempt to cancel a currently queued transaction.
GATT_Query_Maximum_Supported_MTU	Allows a mechanism of querying the maximum supported GATT MTU.
GATT_Change_Maximum_Supported_MTU	Allows a mechanism of changing the maximum supported GATT MTU.
GATT_Query_Connection_MTU	Allows a mechanism to query the MTU for a specified connection.
GATT_Query_Connection_ID	Allows a mechanism to query the Connection ID for a specified connection.
GATT_Query_Transaction_Opcode	Allows a mechanism to query the Attribute Protocol Opcode for a specified transaction.

GATT_Set_Queuing_Parameters	Allows a mechanism of changing the queuing parameters that are used to limit the number of un-acknowledged packets that are queued internally.
GATT_Get_Queuing_Parameters	Allows a mechanism of querying the current queuing parameters that are currently being used to limit the number of un-acknowledged packets that are queued internally.
GATT_Query_Service_Range_Availability	Allows a mechanism of determining if a specified handle range is available to be used to register a service in.

GATT_Initialize

This function is responsible for initializing the GATT profile. This function must be called before any other GATT profile function may be called. This function can only be called once per stack instanced. This function accepts a mandatory connection callback function that is used to monitor GATT connections (for both BR/EDR and LE). This callback is equivalent to a callback that is registered with the GATT_Register_Connection_Events() function, except that the registered function is the ONLY function that will receive BR/EDR incoming connection requests when in manual accept mode.

Notes:

The callback function specified in this function IS required and cannot be NULL.

The registered connection callback will also receive GATT server initiated events as well.

Prototype:

int BTPSAPI **GATT_Initialize**(unsigned int BluetoothStackID, unsigned long Flags,
GATT_Connection_Event_Callback_t ConnectionEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
Flags	Initialization flags bit-mask. This value must be one (or more) of the following bit-mask constant flags: GATT_INITIALIZATION_FLAGS_SUPPORT_LE GATT_INITIALIZATION_FLAGS_SUPPORT_BR_EDR
ConnectionEventCallback	Callback function that is registered to receive connection events.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each connection event.

Return:

Zero if successful.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_CONTEXT_ALREADY_EXISTS
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Device_Connection_Request
etGATT_Connection_Device_Connection
etGATT_Connection_Device_Disconnection
etGATT_Connection_Server_Indication
etGATT_Connection_Server_Notification

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.

GATT_Cleanup

This function is responsible for cleaning up and freeing all resources associated with a GATT instance. After this function is called, no other GATT profile function can be called until after a successful call to the GATT_Initialize() function is performed.

Prototype:

int BTPSAPI **GATT_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:

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Register_Connection_Events

This function is provided to allow a mechanism to register a connection event callback to receive GATT connection events. This registered callback will receive all of the same events as the connection event callback that was registered with the GATT_Initialize() function EXCEPT for the BR/EDR only etGATT_Connection_Device_Connection_Request event.

Notes:

This function only needs to be called if an additional connection event callback functions are required for monitoring GATT connection events. Under most circumstances, calling this function will not be required. It should be noted that if this function is called with the same exact connection callback event function as the function passed to the GATT_Initialize() function, then the function will physically be called twice for the shared events (passing the respective callback parameter to each invocation).

Prototype:

int BTPSAPI **GATT_Register_Connection_Events**(unsigned int BluetoothStackID,
GATT_Connection_Event_Callback_t ConnectionEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionEventCallback	Callback function that is registered to receive connection events.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function with each connection event.

Return:

Positive, non-zero value if successful. This value represents the event connection callback ID value that can be passed to the connection event un-registration function to un-register the callback.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Device_Connection
etGATT_Connection_Device_Disconnection
etGATT_Connection_Server_Indication
etGATT_Connection_Server_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.

GATT_Un_Register_Connection_Events

This function is responsible for un-registering a connection event callback that was registered via a successful call to the GATT_Register_Connection_Events() function.

Prototype:

int BTPSAPI **GATT_Un_Register_Connection_Events**(unsigned int BluetoothStackID, unsigned int EventCallbackID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
EventCallbackID	Event callback ID of the event callback to un-register. This value was obtained via the successful return value from calling the GATT_Register_Connection_Events() function.

Return:

Zero if successful.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Connection_Request_Response

This function is provided to allow a mechanism to respond to an incoming BR/EDR only GATT connection from a specified remote BR/EDR device. This function allows the ability to accept or reject the incoming BR/EDR connection from the specified Bluetooth device.

Notes:

This function is ONLY applicable to BR/EDR connections.

This function should only be called in response to receiving the:

etGATT_Connection_Device_Connection_Request

event. Note that this event is only dispatched when the incoming connection mode is set to:

gimManualAccept

Prototype:

int BTPSAPI **GATT_Connection_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 remote BR/EDR Bluetooth device that is attempting to connect with the local device.
AcceptConnection	Specifies whether to accept the incoming BR/EDR connection (TRUE) or reject the incoming BR/EDR connection (FALSE).

Return:

Zero value if successful.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Device_Connection_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.

GATT_Connect

This function is provided to allow a mechanism to create a BR/EDR only GATT connection to the specified remote BR/EDR device. Due to the differences between BR/EDR and LE regarding the mechanisms for connection establishment, the connection event callback that is specified for this connection is only used to dispatch the connection confirmation event (etGATT_Connection_Device_Connection_Confirmation). This allows the caller the ability to determine the status of the connection attempt. The return value of this function represents the connection ID that can be used in functions that require a connection ID to send data to a connected remote device.

Notes:

This function is ONLY applicable to BR/EDR connections.

The event callback function will only receive a single event. This event is the etGATT_Connection_Device_Connection_Confirmation event. If the caller requires other connection events it must either register a separate connection event handler or monitor the connection event handler that was registered when GATT was initialized.

Prototype:

int BTPSAPI **GATT_Connect**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR,
GATT_Connection_Event_Callback_t ConnectionEventCallback,
unsigned long CallbackParameter)

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 BR/EDR Bluetooth GATT server to connect.
ConnectionEventCallback	Callback function that is registered to receive the connection confirmation event (which contains the connection status).
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the connection ID of the connection. This value can be passed to functions that require a connection ID to send data to a remote GATT server (or disconnect).

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Device_Connection_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.

GATT_Disconnect

This function is responsible for disconnecting a currently connected BR/EDR GATT connection (either initiated locally or remotely).

Prototype:

int BTPSAPI **GATT_Disconnect**(unsigned int BluetoothStackID,
unsigned int ConnectionID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID that identifies the currently connected BR/EDR GATT connection that is to be disconnected.

Return:

Zero if successful.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Get_Incoming_Connection_Mode

This function allows a mechanism to query the current BR/EDR incoming GATT connection mode.

Prototype:

int BTPSAPI **GATT_Get_Incoming_Connection_Mode**(unsigned int BluetoothStackID,
GATT_Incoming_Connection_Mode_t *IncomingConnectionMode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
IncomingConnectionMode	Pointer to a buffer that is to receive the currently configured BR/EDR GATT incoming connection mode. This value will be one of the following: gimAutomaticAccept gimAutomaticReject gimManualAccept

Return:

Zero if successful.

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

BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Set_Incoming_Connection_Mode

This function allows a mechanism to change the current BR/EDR incoming GATT connection mode.

Prototype:

int BTPSAPI **GATT_Set_Incoming_Connection_Mode**(unsigned int BluetoothStackID, GATT_Incoming_Connection_Mode_t IncomingConnectionMode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
IncomingConnectionMode	New BR/EDR GATT incoming connection mode. This value must be one of the following: gimAutomaticAccept gimAutomaticReject gimManualAccept

Return:

Zero if successful.

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

BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Register_SDP_Record

This function provides a means to add a generic GATT SDP 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 GATT_Un_Register_SDP_Record**( )** to SDP_Delete_Service_Record(), and is defined as follows:

GATT_Un_Register_SDP_Record(__BluetoothStackID, __SDPRecordHandle)

2. If no UUID information is specified in the SDPServiceRecord Parameter, then the default GATT Service Class is added.

Prototype:

int BTPSAPI **GATT_Register_SDP_Record**(unsigned int BluetoothStackID,
GATT_SDP_Service_Record_t *SDPServiceRecord,
DWord_t *SDPServiceRecordHandle)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize
SDPServiceRecord	Any additional Service Discovery Protocol information to be added to the record for the GATT SDP record. This is structure defined as:

```
typedef struct
{
  unsigned int          NumberServiceClassUUID;
  SDP_UUID_Entry_t      *SDPUUIDEntries;
} GATT_SDP_Service_Record_t;
```

SDPServiceRecordHandle	Returned handle to the SDP Database entry which 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:

BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Register_Service

This function allows a GATT service to be registered with the local GATT server. This function will register a service with the specified service attributes with the local GATT server. This function will return the unique service ID which is used to identify the service as well as the starting and ending attribute handles of the service in the local GATT service.

Notes:

1. A callback function is required to be specified for the registered service. This callback will be called by the local GATT server when a request arrives from a remote GATT client (for example, reading an attribute value).
2. If the GATT_SERVICE_FLAGS_BR_EDR_SERVICE bit is set in the ServiceFlags parameter, it is the responsibility of the application to call **GATT_Register_Service_SDP_Record**() passing in the handle range returned from a successful call to this function to register an SDP Record for the BR/EDR service.

Prototype:

int BTPSAPI **GATT_Register_Service**(unsigned int BluetoothStackID,
Byte_t ServiceFlags, unsigned int NumberOfServiceAttributeEntries,
GATT_Service_Attribute_Entry_t *ServiceTable,
GATT_Attribute_Handle_Group_t *ServiceHandleGroupResult,
GATT_Server_Event_Callback_t ServerEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServiceFlags	Specifies the current service flags to apply to the registered service. The value of this parameter is a bit-mask of zero or more of the following values: GATT_SERVICE_FLAGS_LE_SERVICE GATT_SERVICE_FLAGS_BR_EDR_SERVICE
NumberOfServiceAttributeEntries	Specifies the total number of service attribute entries that are contained in the ServiceTable parameter.
ServiceTable	Pointer to an array of service attribute entries that specify all of the service attributes for the registered service. Each member in this array is defined by the following structure:

```
typedef struct
{
  Byte_t                          Attribute_Flags;
  GATT_Service_Attribute_Entry_Type_t
                                  Attribute_Entry_Type;
  void                           *Attribute_Value;
} GATT_Service_Attribute_Entry_t;
```

where, Attribute_Flags is a bit list. Possible bit values are:

GATT_ATTRIBUTE_FLAGS_READABLE
GATT_ATTRIBUTE_FLAGS_WRITABLE
GATT_ATTRIBUTE_FLAGS_HIDDEN
GATT_ATTRIBUTE_FLAGS_READABLE_WRITABLE

and, the Attribute_Entry_Type is defined to be one of the following values:

aetPrimaryService16
aetPrimaryService128
aetSecondaryService16
aetSecondaryService128
aetIncludeDefinition
aetCharacteristicDeclaration16
aetCharacteristicDeclaration128
aetCharacteristicValue16
aetCharacteristicValue128
aetCharacteristicDescriptor16
aetCharacteristicDescriptor128

and the Attribute_Value member is a pointer to a buffer that contains the correct data type for the specified attribute entry type:

Attribute Entry Type	Attribute Value Data
aetPrimaryService16	GATT_Primary_Service_16_Entry_t
aetPrimaryService128	GATT_Primary_Service_128_Entry_t
aetSecondaryService16	GATT_Secondary_Service_16_Entry_t
aetSecondaryService128	GATT_Secondary_Service_128_Entry_t
aetIncludeDefinition	GATT_Include_Definition_Entry_t
aetCharacteristicDeclaration16	GATT_Characteristic_Declaration_16_Entry_t
aetCharacteristicDeclaration128	GATT_Characteristic_Declaration_128_Entry_t
aetCharacteristicValue16	GATT_Characteristic_Value_16_Entry_t
aetCharacteristicValue128	GATT_Characteristic_Value_128_Entry_t
aetCharacteristicDescriptor16	GATT_Characteristic_Descriptor_16_Entry_t

aetCharacteristicDescriptor128	GATT_Characteristic_Descriptor_128_Entry_t

where, the structures above are defined as:

```
typedef struct
{
  UUID_16_t      Service_UUID;
} GATT_Primary_Service_16_Entry_t;

typedef struct
{
  UUID_128_t     Service_UUID;
} GATT_Primary_Service_128_Entry_t;

typedef struct
{
  UUID_16_t      Service_UUID;
} GATT_Secondary_Service_16_Entry_t;

typedef struct
{
  UUID_128_t     Service_UUID;
} GATT_Secondary_Service_128_Entry_t;

typedef struct
{
  unsigned int ServiceID;
} GATT_Include_Definition_Entry_t;

typedef struct
{
  Byte_t         Properties;
  UUID_16_t      Characteristic_Value_UUID;
} GATT_Characteristic_Declaration_16_Entry_t;

typedef struct
{
  Byte_t         Properties;
  UUID_128_t     Characteristic_Value_UUID;
} GATT_Characteristic_Declaration_128_Entry_t;

typedef struct
{
  UUID_16_t       Characteristic_Value_UUID;
  unsigned int    Characteristic_Value_Length;
  Byte_t         *Characteristic_Value;
} GATT_Characteristic_Value_16_Entry_t;

typedef struct
{
  UUID_128_t      Characteristic_Value_UUID;
  unsigned int    Characteristic_Value_Length;
  Byte_t         *Characteristic_Value;
} GATT_Characteristic_Value_128_Entry_t;

typedef struct
```

```
{
  UUID_16_t       Characteristic_Descriptor_UUID;
  unsigned int    Characteristic_Descriptor_Length;
  Byte_t          *Characteristic_Descriptor;
} GATT_Characteristic_Descriptor_16_Entry_t;

typedef struct
{
  UUID_128_t      Characteristic_Descriptor_UUID;
  unsigned int    Characteristic_Descriptor_Length;
  Byte_t          *Characteristic_Descriptor;
} GATT_Characteristic_Descriptor_128_Entry_t;
```

ServiceHandleGroupResult	This parameter is both an input and output parameter. On input this parameter can be used to request that GATT place the service at a specific handle range in the GATT database. This is accomplished by setting the Starting_Handle and Ending_Handle members of this structure to the requested handle range in the GATT database. If these members are zero, or otherwise invalid, on input then the GATT layer will place the service in the first available region in the GATT database. On output GATT will return the handle range of the registered service if this function returns success. This structure is declared as follows:

```
typedef struct
{
  Word_t Starting_Handle;
  Word_t Ending_Handle;
} GATT_Attribute_Handle_Group_t;
```

ServerEventCallback	Callback function that is registered to receive events that are associated with the specified service.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the service ID that uniquely identifies the service in the local GATT database.

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

BTGATT_ERROR_INVALID_SERVICE_TABLE_FORMAT
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_NOT_INITIALIZED

Possible Events:

etGATT_Server_Device_Connection

etGATT_Server_Device_Disconnection
etGATT_Server_Read_Request
etGATT_Server_Write_Request
etGATT_Server_Signed_Write_Request
etGATT_Server_Execute_Write_Request

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.

GATT_Un_Register_Service

This function allows a previously registered GATT service to be removed from the local GATT server. This function will free all resources that are being utilized by the service being removed from the GATT database.

Prototype:

void BTPSAPI **GATT_Un_Register_Service**(unsigned int BluetoothStackID,
unsigned int ServiceID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServiceID	Specifies the service ID of the service that is to be removed. This value is the successful return value from the call to GATT_Register_Service.

Return:

Possible Events:

Notes:

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

GATT_Register_Service_SDP_Record

This function provides a means to add a service GATT SDP 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 GATT_Un_Register_Service_SDP_Record**( )** to SDP_Delete_Service_Record(), and is defined as follows:

GATT_Un_Register_Service_SDP_Record(__BluetoothStackID, __SDPRecordHandle)

Prototype:

int BTPSAPI **GATT_Register_Service_SDP_Record**(unsigned int BluetoothStackID,
GATT_SDP_Service_Record_t *SDPServiceRecord,

GATT_Attribute_Handle_Group_t *ServiceHandleRange,

DWord_t *SDPServiceRecordHandle)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize
SDPServiceRecord	Service Discovery Protocol information to be added to the record for the GATT SDP record. This is structure defined as: typedef struct { unsigned int NumberServiceClassUUID; SDP_UUID_Entry_t *SDPUUIDEntries; } **GATT_SDP_Service_Record_t**;
ServiceHandleRange	Service Handle Range that is returned from a successful call to **GATT_Register_Service**(). This structure is declared as follows: typedef struct { Word_t Starting_Handle; Word_t Ending_Handle; } **GATT_Attribute_Handle_Group_t**;
SDPServiceRecordHandle	Returned handle to the SDP Database entry which 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:

BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Read_Response

This function is provided to allow a mechanism for a service handler to successfully respond to a received GATT/ATT read request (etGATT_Server_Read_Request event).

Notes:

This function only allows a successful response to be sent. If an error response is required, then the GATT_Error_Response() function should be used to respond with the error information.

Prototype:

int BTPSAPI **GATT_Read_Response**(unsigned int BluetoothStackID,
unsigned int TransactionID, unsigned int DataLength, Byte_t *Data)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the original read request. This value was received in the etGATT_Server_Read_Request event.
DataLength	Specifies the amount of data to return. This is the amount of data (in bytes) pointed to by the Data parameter.
Data	Specifies the buffer that contains the data to return in the read response. This buffer must point to a buffer that contains (at least) as many bytes as specified by the DataLength parameter.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Write_Response

This function is provided to allow a mechanism for a service handler to successfully respond to a received GATT/ATT write request (etGATT_Server_Write_Request event).

Notes:

This function only allows a successful response to be sent. If an error response is required, then the GATT_Error_Response() function should be used to respond with the error information.

Prototype:

int BTPSAPI **GATT_Write_Response**(unsigned int BluetoothStackID,
unsigned int TransactionID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the original write request. This value was received in the etGATT_Server_Write_Request event.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Execute_Write_Response

This function is provided to allow a mechanism for a service handler to successfully respond to a received GATT/ATT execute write request (etGATT_Server_Execute_Write_Request).

Notes:

This function only allows a successful response to be sent. If an error response is required, then the GATT_Error_Response() function should be used to respond with the error information.

Prototype:

int BTPSAPI **GATT_Execute_Write_Response**(unsigned int BluetoothStackID,
unsigned int TransactionID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the original execute write request. This value was received in the etGATT_Server_Execute_Write_Request event.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Error_Response

This function is provided to allow a mechanism for a service handler to respond to a received GATT/ATT request with an error response.

Prototype:

int BTPSAPI **GATT_Error_Response**(unsigned int BluetoothStackID,
unsigned int TransactionID, Word_t AttributeOffset, Byte_t ErrorCode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the original request. This value was received in the GATT request event.
AttributeOffset	Attribute offset of the first attribute that causes the error. This value will be greater than or equal to zero (specifies the very first attribute in the service) and less than the maximum number of attributes contained in the service.

ErrorCode — Error code to return as a response to the request. This may be one of the following values:

- ATT_PROTOCOL_ERROR_CODE_INVALID_HANDLE
- ATT_PROTOCOL_ERROR_CODE_READ_NOT_PERMITTED
- ATT_PROTOCOL_ERROR_CODE_WRITE_NOT_PERMITTED
- ATT_PROTOCOL_ERROR_CODE_INVALID_PDU
- ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_AUTHENTICATION
- ATT_PROTOCOL_ERROR_CODE_REQUEST_NOT_SUPPORTED
- ATT_PROTOCOL_ERROR_CODE_INVALID_OFFSET
- ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_AUTHORIZATION
- ATT_PROTOCOL_ERROR_CODE_PREPARE_QUEUE_FULL
- ATT_PROTOCOL_ERROR_CODE_ATTRIBUTE_NOT_FOUND
- ATT_PROTOCOL_ERROR_CODE_ATTRIBUTE_NOT_LONG
- ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_ENCRYPTION_KEY_SIZE
- ATT_PROTOCOL_ERROR_CODE_INVALID_ATTRIBUTE_VALUE_LENGTH
- ATT_PROTOCOL_ERROR_CODE_UNLIKELY_ERROR
- ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_ENCRYPTION
- ATT_PROTOCOL_ERROR_CODE_UNSUPPORTED_GROUP_TYPE
- ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_RESOURCES

In addition to the above, application specific error codes can be defined. These codes will be within the range:

- ATT_PROTOCOL_ERROR_CODE_APPLICATION_ERROR_START
- ATT_PROTOCOL_ERROR_CODE_APPLICATION_ERROR_END

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Handle_Value_Indication

This function is provided to allow a mechanism for a service handler to send a GATT attribute handle/value indication to a remote, connected, GATT client.

Notes:

Indications require the client to acknowledge that the indication was received. This will be signified by the reception of the etGATT_Server_Confirmation_Response event which will also include the total number of bytes that were indicated.

Prototype:

int BTPSAPI **GATT_Handle_Value_Indication**(unsigned int BluetoothStackID, unsigned int ServiceID, unsigned int ConnectionID, Word_t AttributeOffset, Word_t AttributeValueLength, Byte_t *AttributeValue)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServiceID	Service ID of the service that is sending the indication.
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value indication.
AttributeOffset	Attribute offset of the attribute that is being indicated. This value will be greater than or equal to zero (specifies the very first attribute in the service) and less than the maximum number of attributes contained in the service.
AttributeValueLength	Length (in bytes) of the attribute value data that is to be indicated.
AttributeValue	Buffer that contains the value data to be indicated. This buffer must contain (at least) the number of bytes specified by the AttributeValueLength parameter.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the handle/value indication transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the indication (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Server_Confirmation_Response

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.

GATT_Handle_Value_Notification

This function is provided to allow a mechanism for a service handler to send a GATT attribute handle/value notification to a remote, connected, GATT client.

Notes:

Notifications do not require the client to acknowledge that the notification was received.

Prototype:

int BTPSAPI **GATT_Handle_Value_Notification**(unsigned int BluetoothStackID, unsigned int ServiceID, unsigned int ConnectionID, Word_t AttributeOffset, Word_t AttributeValueLength, Byte_t *AttributeValue)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServiceID	Service ID of the service that is sending the notification.
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value notification.
AttributeOffset	Attribute offset of the attribute that is being notified. This value will be greater than or equal to zero (specifies the very first attribute in the service) and less than the maximum number of attributes contained in the service.
AttributeValueLength	Length (in bytes) of the attribute value data that is to be notified.
AttributeValue	Buffer that contains the value data to be notified. This buffer must contain (at least) the number of bytes specified by the AttributeValueLength parameter.

Return:

Positive, non-zero value if successful. This value represents the number of attribute value bytes that will be sent in the notification.

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

Note that if this function returns:

BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

then this is a signal to the caller that the requested data could NOT be sent because the requested data could not be queued in the outgoing L2CAP Queue (i.e. queuing criteria was not met). The caller then must wait for the:

etGATT_Server_Device_Buffer_Empty

event before trying to send any more data. When this event is signaled, another attempt can be made to send the data to the remote device.

Possible Events:

Notes:

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

GATT_Verify_Signature

This function is provided to allow a mechanism for a service handler verify if the data received in a signed write request (etGATT_Server_Signed_Write_Request event) is correctly signed.

Prototype:

Boolean_t BTPSAPI **GATT_Verify_Signature**(unsigned int BluetoothStackID, unsigned int ServiceID, Word_t AttributeOffset, unsigned int AttributeValueLength, Byte_t *AttributeValue, ATT_Authentication_Signature_t *ReceivedSignature, Encryption_Key_t *CSRK)

Parameters:

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

ServiceID	Service ID of the service that is sending the notification.
AttributeOffset	Attribute offset of the attribute that is being written. This value will be greater than or equal to zero (specifies the very first attribute in the service) and less than the maximum number of attributes contained in the service.
AttributeValueLength	Length (in bytes) of the attribute value data that is to be verified/written.
AttributeValue	Buffer that contains the value data to be verified/written. This buffer must contain (at least) the number of bytes specified by the AttributeValueLength parameter.
ReceivedSignature	Pointer to the GATT/ATT signature that was received in the write request event.
CSRK	Pointer to the connection signature resolving key (CSRK) that is to be used to verify the received signature.

Return:

Boolean TRUE if the verification was successful.

Boolean FALSE if the verification was not successful (or unable to be performed)

Possible Events:

Notes:

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

GATT_Service_Changed_Read_Response

This function is provided to allow a mechanism for a connection handler to successfully respond to a received GATT Service Changed read request (etGATT_Connection_Service_Changed_Read_Request event).

Notes:

This function only allows a successful response to be sent. If an error response is required, then the GATT_Error_Response() function should be used to respond with the error information.

Prototype:

int BTPSAPI **GATT_Service_Changed_Read_Response**(unsigned int BluetoothStackID, unsigned int TransactionID, GATT_Service_Changed_Data_t *Service_Changed_Data)

Parameters:

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

TransactionID	Transaction ID of the original Service Changed read request. This value was received in the etGATT_Connection_Service_Changed_Read_Request event.
Service_Changed_Data	Specifies a pointer to the data to respond to the Service Changed Read Request with. This is structure defined as:

```
typedef struct
{
  Word_t Affected_Start_Handle;
  Word_t Affected_End_Handle;
} GATT_Service_Changed_Data_t;
```

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Service_Changed_CCCD_Read_Response

This function is provided to allow a mechanism for a connection handler to successfully respond to a received GATT Service Changed CCCD read request (etGATT_Connection_Service_Changed_CCCD_Read_Request event).

Notes:

It is the responsibility of the application to respond with the unique CCCD value for each client.

This function only allows a successful response to be sent. If an error response is required, then the GATT_Error_Response() function should be used to respond with the error information.

Prototype:

int BTPSAPI **GATT_Service_Changed_CCCD_Read_Response**(unsigned int BluetoothStackID,
unsigned int TransactionID, Word_t CCCD)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the original Service Changed read request. This value was received in the etGATT_Connection_Service_Changed_CCCD_Read_Request event.
CCCD	Value of the Client's CCCD for the Service Changed characteristic to respond to the request with.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Service_Changed_Indication

This function is provided to allow a mechanism for a connection handler to send a GATT Service Changed Indication to a remote, connected, GATT client.

Notes:

Indications require the client to acknowledge that the indication was received. This will be signified by the reception of the etGATT_Connection_Service_Changed_Confirmation event.

Prototype:

int BTPSAPI **GATT_Service_Changed_Indication**(unsigned int BluetoothStackID, unsigned int ConnectionID, GATT_Service_Changed_Data_t *Service_Changed_Data)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote client device to send the Service Changed indication.

Service_Changed_Data — Specifies a pointer to the Service Changed data to indicate. This is structure defined as:

```
typedef struct
{
  Word_t Affected_Start_Handle;
  Word_t Affected_End_Handle;
} GATT_Service_Changed_Data_t;
```

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the Service Changed indication transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the indication (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Service_Changed_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.

GATT_Exchange_MTU_Request

This function is provided to allow a mechanism for a GATT client to request a change in the ATT MTU for a connected LE device. This function accepts the MTU to request from the remote connected LE device.

This function is ONLY applicable to LE connections.

Prototype:

int BTPSAPI **GATT_Exchange_MTU_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t RequestedMTU,
GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

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

ConnectionID	Connection ID of the currently connected remote GATT server device.
RequestedMTU	The MTU to request from the remote, connected, LE device. This value must be between the following values: ATT_PROTOCOL_MTU_MINIMUM_LE + 1 GATT_MAXIMUM_SUPPORTED_STACK_MTU
ClientEventCallback	Callback function that is registered to receive the exchange MTU response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the exchange MTU request transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Exchange_MTU_Response

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.

GATT_Discover_Services

This function is provided to allow a mechanism for a GATT client to discover the services on a remote, connected GATT server. This function accepts the starting and ending handle ranges to search for services on.

Notes:

To discover all services on a remote GATT server this function should be called with the starting and ending handles set to:

ATT_PROTOCOL_HANDLE_MINIMUM_VALUE
ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE

respectively. The etGATT_Client_Service_Discovery_Response event will specify the services found in the specified range. The client can then call this function again with the starting handle set to one greater than the ending handle returned in the event. This process should be repeated to discover all services on a remote GATT server.

Prototype:

int BTPSAPI **GATT_Discover_Services**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t StartingHandle, Word_t EndingHandle,
GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
StartingHandle	Starting attribute handle to use to begin the search range. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
EndingHandle	Ending attribute handle to use to end the search range. This value must be between (and at least one value larger than the starting attribute handle): ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ClientEventCallback	Callback function that is registered to receive the discover services response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the discover services transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Service_Discovery_Response

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.

GATT_Discover_Services_By_UUID

This function is provided to allow a mechanism for a GATT client to discover the services on a remote, connected GATT server that match the specified UUID. This function accepts the starting and ending handle ranges to search for services on. This function allows the ability to search for a specific service instead of searching for all services.

Notes:

To discover a service on a remote GATT server this function should be called with the starting and ending handles set to:

ATT_PROTOCOL_HANDLE_MINIMUM_VALUE
ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE

respectively. The etGATT_Client_Service_Discovery_By_UUID_Response event will specify if the service was found in the specified range. The client can then call this function again with the starting handle set to one greater than the ending handle returned in the event. This process should be repeated to discover all service of the specified type on a remote GATT server.

Prototype:

int BTPSAPI **GATT_Discover_Services_By_UUID**(unsigned int BluetoothStackID, unsigned int ConnectionID, Word_t StartingHandle, Word_t EndingHandle, GATT_UUID_t *UUID, GATT_Client_Event_Callback_t ClientEventCallback, unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
StartingHandle	Starting attribute handle to use to begin the search range. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
EndingHandle	Ending attribute handle to use to end the search range. This value must be between (and at least one value larger than the starting attribute handle): ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE

UUID	Contains the service UUID to search for. This structure has the following format: typedef struct { GATT_UUID_Type_t UUID_Type; union { UUID_16_t UUID_16; UUID_128_t UUID_128; } UUID; } **GATT_UUID_t**; where, UUID_Type is defined to be one of the following: guUUID_16 guUUID_128
ClientEventCallback	Callback function that is registered to receive the discover services by UUID response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the discover services transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Service_Discovery_By_UUID_Response

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.

GATT_Discover_Included_Services

This function is provided to allow a mechanism for a GATT client to discover any included services for a specific service on a remote, connected GATT server. This function accepts the starting and ending handle range of the service to search.

Notes:

The starting and ending handle values that are passed to this function should specify the starting and ending handles of a single service. This will allow the ability to discern which services are included with the specific service referenced by the starting and ending handles.

Prototype:

int BTPSAPI **GATT_Discover_Included_Services**(unsigned int BluetoothStackID, unsigned int ConnectionID, Word_t ServiceStartingHandle, Word_t ServiceEndingHandle, GATT_Client_Event_Callback_t ClientEventCallback, unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
ServiceStartingHandle	Starting service attribute handle to use to begin the search range. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ServiceEndingHandle	Ending service attribute handle to use to end the search range. This value must be between (and at least one value larger than the starting attribute handle): ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ClientEventCallback	Callback function that is registered to receive the included services response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the discover included services transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Included_Services_Discovery_Response

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.

GATT_Discover_Characteristics

This function is provided to allow a mechanism for a GATT client to discover any characteristics of a specific service on a remote, connected GATT server. This function accepts the starting and ending handle range of the service to search.

Notes:

The starting and ending handle values that are passed to this function should specify the starting and ending handles of a single service. This will allow the ability to discern which characteristics are included with the specific service referenced by the starting and ending handles.

Prototype:

int BTPSAPI **GATT_Discover_Characteristics**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t ServiceStartingHandle,
Word_t ServiceEndingHandle, GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
ServiceStartingHandle	Starting service attribute handle to use to begin the search range. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ServiceEndingHandle	Ending service attribute handle to use to end the search range. This value must be between (and at least one value larger than the starting attribute handle): ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ClientEventCallback	Callback function that is registered to receive the included characteristics response event.

CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the discover characteristics transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Characteristic_Discovery_Response

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.

GATT_Discover_Characteristic_Descriptors

This function is provided to allow a mechanism for a GATT client to discover any characteristic descriptors of a specific characteristic of a specific service on a remote, connected GATT server. This function accepts the starting and ending handle range of the characteristic to search.

Notes:

The starting and ending handle values that are passed to this function should specify the starting and ending handles of a single characteristic. This will allow the ability to discern which characteristic descriptors are included with the specific characteristic referenced by the starting and ending handles.

Prototype:

```
int BTPSAPI GATT_Discover_Characteristic_Descriptors(
    unsigned int BluetoothStackID, unsigned int ConnectionID,
    Word_t CharacteristicStartingHandle, Word_t CharacteristicEndingHandle,
    GATT_Client_Event_Callback_t ClientEventCallback,
    unsigned long CallbackParameter)
```

Parameters:

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

ConnectionID	Connection ID of the currently connected remote GATT server device.
CharacteristicStartingHandle	Starting characteristic attribute handle to use to begin the search range. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
CharacteristicEndingHandle	Ending characteristic attribute handle to use to end the search range. This value must be between (and at least one value larger than the starting attribute handle): ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ClientEventCallback	Callback function that is registered to receive the included characteristic descriptor response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the discover characteristic descriptors transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Characteristic_Descriptor_Discovery_Response

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.

GATT_Read_Value_Request

This function is provided to allow a mechanism for a GATT client to issue a read value request to a connected, remote GATT server.

Prototype:

int BTPSAPI **GATT_Read_Value_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t AttributeHandle,

GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeHandle	Attribute handle of the attribute that is to be read. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ClientEventCallback	Callback function that is registered to receive the included read value response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the read value transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Read_Response

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.

GATT_Read_Long_Value_Request

This function is provided to allow a mechanism for a GATT client to issue a read long value request to a connected, remote GATT server.

Prototype:

int BTPSAPI **GATT_Read_Long_Value_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t AttributeHandle, Word_t AttributeOffset,
GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeHandle	Attribute handle of the attribute that is to be read. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
AttributeOffset	Starting offset (in bytes) of the attribute value data to read.
ClientEventCallback	Callback function that is registered to receive the included read value response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the read long value transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Long_Read_Response

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.

GATT_Read_Value_By_UUID_Request

This function is provided to allow a mechanism for a GATT client to issue a read value request for a specific UUID attribute for a specific service to a connected, remote GATT server.

Notes:

The starting and ending handle values that are passed to this function should specify the starting and ending handles of a single service. This will allow the ability to discern which attribute value is associated with the specific service referenced by the starting and ending handles.

Prototype:

int BTPSAPI **GATT_Read_Value_By_UUID_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, GATT_UUID_t *AttributeUUID,
Word_t ServiceStartHandle, Word_t ServiceEndHandle,
GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeUUID	Contains the attribute UUID to search for. This structure has the following format: typedef struct { GATT_UUID_Type_t UUID_Type; union { UUID_16_t UUID_16; UUID_128_t UUID_128; } UUID; } **GATT_UUID_t**; where, UUID_Type is defined to be one of the following: guUUID_16 guUUID_128
ServiceStartingHandle	Starting service attribute handle to use to begin the search range. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ServiceEndingHandle	Ending service attribute handle to use to end the search range. This value must be between (and at least one value larger than the starting attribute handle):

ATT_PROTOCOL_HANDLE_MINIMUM_VALUE
ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE

ClientEventCallback	Callback function that is registered to receive the included read attribute value by UUID response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the read attribute value by UUID transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Read_By_UUID_Response

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.

GATT_Read_Multiple_Values_Request

This function is provided to allow a mechanism for a GATT client to issue a read value request for a list of specific attributes to a connected, remote GATT server.

Prototype:

int BTPSAPI **GATT_Read_Multiple_Values_Request**(unsigned int BluetoothStackID, unsigned int ConnectionID, Word_t NumberOfHandles, Word_t *AttributeHandleList, GATT_Client_Event_Callback_t ClientEventCallback, unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
NumberOfHandles	Specifies the total number of attribute handle entries that are contained in the AttributeHandleList parameter.

AttributeHandleList	Pointer to an array of attribute handle entries that specify all of the attributes that should be read. Each member in this array must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
ClientEventCallback	Callback function that is registered to receive the included read multiple values response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the read multiple values transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Read_Multiple_Response

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.

GATT_Write_Request

This function is provided to allow a mechanism for a GATT client to issue a write value request for a specific attribute to a connected, remote GATT server.

Notes:

This function will not write a value with a length greater than the current MTU minus 3. If the value to be written is larger than this then the GATT_Prepare_Write_Request() function should be used.

It is possible that this function can write less data than specified (due to the MTU and packet header overhead). The write response event (etGATT_Client_Write_Response) will contain the total number of bytes that were able to be written.

Prototype:

int BTPSAPI **GATT_Write_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t AttributeHandle, Word_t AttributeLength,
void *AttributeValue, GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeHandle	Attribute handle of the attribute to write. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
AttributeLength	Length (in bytes) of the actual attribute value data to write to the specified attribute.
AttributeValue	Buffer that contains (at least) as many bytes a specified by the AttributeLength parameter.
ClientEventCallback	Callback function that is registered to receive the included the write value response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the write value transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Write_Response

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.

GATT_Write_Without_Response_Request

This function is provided to allow a mechanism for a GATT client to issue a write value request for a specific attribute to a connected, remote GATT server. This differs from the GATT_Write_Request() function in that there is no response from the server about the write request. This means that the client is not able to tell how much (if any) of the data was actually processed by the remote GATT server.

Notes:

This function will not write a value with a length greater than the current MTU minus 3.

It is possible that this function can write less data than specified (due to the MTU and packet header overhead). The return value will indicate the total number of bytes that will be written.

Prototype:

int BTPSAPI **GATT_Write_Without_Response_Request**(unsigned int BluetoothStackID, unsigned int ConnectionID, Word_t AttributeHandle, Word_t AttributeLength, void *AttributeValue)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeHandle	Attribute handle of the attribute to write. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
AttributeLength	Length (in bytes) of the actual attribute value data to write to the specified attribute.
AttributeValue	Buffer that contains (at least) as many bytes a specified by the AttributeLength parameter.

Return:

Positive, non-zero value if successful. This value represents the amount of data that will be written to the remote device.

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

Note that if this function returns:

BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

then this is a signal to the caller that the requested data could NOT be sent because the requested data could not be queued in the outgoing L2CAP Queue (i.e. queuing criteria was not met). The caller then must wait for the:

etGATT_Connection_Device_Buffer_Empty

event before trying to send any more data. When this event is signaled, another attempt can be made to send the data to the remote device.

Possible Events:

Notes:

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

GATT_Signed_Write_Without_Response_Request

This function is provided to allow a mechanism for a GATT client to issue a signed write value request for a specific attribute to a connected, remote GATT server. Because there is no response to this function, the client is not able to tell how much (if any) of the data was actually processed by the remote GATT server.

Notes:

This function will not write a value with a length greater than the current MTU minus 3.

It is possible that this function can write less data than specified (due to the MTU and packet header overhead). The return value will indicate the total number of bytes that will be written.

Prototype:

int BTPSAPI **GATT_Signed_Write_Without_Response_Request**(unsigned int BluetoothStackID, unsigned int ConnectionID, Word_t AttributeHandle, Word_t AttributeLength, void *AttributeValue, Encryption_Key_t *CSRK)

Parameters:

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

ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeHandle	Attribute handle of the attribute to write. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
AttributeLength	Length (in bytes) of the actual attribute value data to write to the specified attribute.
AttributeValue	Buffer that contains (at least) as many bytes a specified by the AttributeLength parameter.
CSRK	Pointer to the connection signature resolving key (CSRK) that will be used to sign the data that is to be sent.

Return:

Positive, non-zero value if successful. This value represents the amount of data that will be written to the remote device.

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

Note that if this function returns:

BTPS_ERROR_INSUFFICIENT_BUFFER_SPACE

then this is a signal to the caller that the requested data could NOT be sent because the requested data could not be queued in the outgoing L2CAP Queue (i.e. queuing criteria was not met). The caller then must wait for the:

etGATT_Connection_Device_Buffer_Empty

event before trying to send any more data. When this event is signaled, another attempt can be made to send the data to the remote device.

Possible Events:

Notes:

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

GATT_Prepare_Write_Request

This function is provided to allow a mechanism for a GATT client to issue a write long value request for a specific attribute to a connected, remote GATT server. This function differs from the GATT_Write_Request() function in that this function can be used to write values that span multiple PDU's. Once all of the data has been prepared (i.e. sent successfully) the client can issue the GATT_Execute_Write_Request() function to commit the value data in a single, atomic, transaction (and receive a status response).

Notes:

The response event (etGATT_Client_Prepare_Write_Response) will signify to the client how much data was sent. The client can then use this data to determine the new offset of data to write and call this function again. This process should be repeated until either all of the data has been sent or an error occurred.

The GATT_Execute_Write_Request() function can be called to actually write/commit the data to the remote GATT server after all of the data value has been transmitted successfully.

Prototype:

int BTPSAPI **GATT_Write_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Word_t AttributeHandle, Word_t AttributeLength,
Word_t AttributeValueOffset, void *AttributeValue,
GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
AttributeHandle	Attribute handle of the attribute to write. This value must be between: ATT_PROTOCOL_HANDLE_MINIMUM_VALUE ATT_PROTOCOL_HANDLE_MAXIMUM_VALUE
AttributeLength	Total length (in bytes) of the actual attribute value data to write to the specified attribute.
AttributeValueOffset	Offset (in bytes) of the attribute value to write. This value must be smaller than the AttributeLength parameter.
AttributeValue	Buffer that contains (at least) as many bytes a specified by the AttributeLength parameter minus the AttributeValueOffset parameter.
ClientEventCallback	Callback function that is registered to receive the included the prepare write value response event.

CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the prepare write value transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_HANDLE_VALUE
BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Prepare_Write_Response

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.

GATT_Execute_Write_Request

This function is provided to allow a mechanism for a GATT client to issue an execute write long value request for a specific attribute to a connected, remote GATT server. The write that is to be executed must have been prepared by calling the GATT_Prepare_Write() function one or more times. This function should be called once all of the value data has been prepared (i.e. sent successfully). The client can then issue this function to commit the value data in a single, atomic, transaction (and receive a status response).

Notes:

This function can also be used to cancel any prior writes that were issued via one or more successful calls to the GATT_Prepare_Write_Request() function.

The GATT_Execute_Write_Request() function can be called to actually write/commit the data to the remote GATT server after all of the data value has been transmitted successfully. See the GATT_Prepare_Write_Request() function for more information.

Prototype:

int BTPSAPI **GATT_Write_Request**(unsigned int BluetoothStackID,
unsigned int ConnectionID, Boolean_t CancelWrite,
GATT_Client_Event_Callback_t ClientEventCallback,
unsigned long CallbackParameter)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
CancelWrite	Boolean flag that specifies whether or not to cancel (TRUE) the prepared write requests, or to commit/execute the prepared write requests (FALSE).
ClientEventCallback	Callback function that is registered to receive the included the prepare write value response event.
CallbackParameter	A user-defined parameter (e.g., a tag value) that will be passed back to the user in the callback function.

Return:

Positive, non-zero value if successful. This value represents the transaction ID of the execute prepared write transaction. This value can be passed to the GATT_Cancel_Transaction() function to cancel the transaction (if required).

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

BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Execute_Write_Response

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.

GATT_Handle_Value_Confirmation

This function is provided to allow a mechanism for a GATT client to issue an acknowledgement for a received handle/value indication event (etGATT_Connection_Server_Indication). Note that this event is dispatched via either the callback registered with the GATT_Initialize() function or a callback registered via the GATT_Register_Connection_Events() function.

Notes:

The connection ID and transaction ID values that are passed to this function should be the values that were contained in the handle/value indication event (etGATT_Connection_Server_Indication).

Prototype:

int BTPSAPI **GATT_Handle_Value_Confirmation** (unsigned int BluetoothStackID, unsigned int ConnectionID, unsigned int TransactionID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device.
TransactionID	Transaction ID of the received handle/value indication that is being acknowledged.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Start_Service_Discovery

This function provides a mechanism of performing a service discovery operation that will automatically discover all included services, characteristics and characteristic descriptors for either all services supported by a remote device or all services of a specified UUID that are supported by a remote device. This function is provided to simplify the GATT service discovery process.

Notes:

The NumberOfUUID and UUIDList parameters are optional and may be set to 0 and NULL respectively. If these parameters are not specified then all services on the specified remote device will be discovered.

Only 1 service discovery operation per remote device can be outstanding at a time.

Prototype:

int BTPSAPI **GATT_Start_Service_Discovery(**unsigned int BluetoothStackID, unsigned int ConnectionID, unsigned int NumberOfUUID, GATT_UUID_t *UUIDList, GATT_Service_Discovery_Event_Callback_t ServiceDiscoveryCallback, unsigned long CallbackParameter**)**

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device to perform service discovery on.
NumberOfUUID	Option parameter that, if specified, contains the number of UUIDs that are contained in the UUIDList parameter.
UUIDList	Optional list of Service UUIDs to attempt to discover on the specified remote device.
ServiceDiscoveryCallback	Callback function that will be called with the result of the service discovery operation.
CallbackParameter	Callback parameter that will be passed to the specified ServiceDiscoveryCallback when called with the result of the service discovery operation.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
BTGATT_ERROR_OUTSTANDING_REQUEST_EXISTS

Possible Events:

etGATT_Service_Discovery_Indication
etGATT_Service_Discovery_Complete

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.

GATT_Start_Service_Discovery_Handle_Range

This function provides a mechanism of performing a service discovery operation that will automatically discover all included services, characteristics and characteristic descriptors for either all services supported by a remote device or all services of a specified UUID that are supported by a remote device in a specific handle range of the remote GATT database. This function is provided to simplify the GATT service discovery process.

Notes:

The DiscoveryHandleRange parameter is optional. However if it is specified the handle range must be valid (i.e. Start and End Handle must be non-zero and Start Handle must be less than or equal to End Handle).

The NumberOfUUID and UUIDList parameters are optional and may be set to 0 and NULL respectively. If these parameters are not specified then all services on the specified remote device will be discovered.

Only 1 service discovery operation per remote device can be outstanding at a time.

Prototype:

```
int BTPSAPI GATT_Start_Service_Discovery_Handle_Range(
    unsigned int BluetoothStackID, unsigned int ConnectionID,
    GATT_Attribute_Handle_Group_t *DiscoveryHandleRange,
    unsigned int NumberOfUUID, GATT_UUID_t *UUIDList,
    GATT_Service_Discovery_Event_Callback_t ServiceDiscoveryCallback,
    unsigned long CallbackParameter)
```

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device to perform service discovery on.
DiscoveryHandleRange	Handle range of the GATT database on the remote device to perform the discovery procedure on.
NumberOfUUID	Option parameter that, if specified, contains the number of UUIDs that are contained in the UUIDList parameter.
UUIDList	Optional list of Service UUIDs to attempt to discover on the specified remote device.
ServiceDiscoveryCallback	Callback function that will be called with the result of the service discovery operation.
CallbackParameter	Callback parameter that will be passed to the specified ServiceDiscoveryCallback when called with the result of the service discovery operation.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
BTGATT_ERROR_OUTSTANDING_REQUEST_EXISTS

Possible Events:

etGATT_Service_Discovery_Indication
etGATT_Service_Discovery_Complete

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.

GATT_Stop_Service_Discovery

This function provides a mechanism of stopping a service discovery operation that was previously started using the GATT_Start_Service_Discovery() or GATT_Start_Service_Discovery_Handle_Range() API.

Prototype:

int BTPSAPI **GATT_Stop_Service_Discovery(**unsigned int BluetoothStackID,
unsigned int ConnectionID**)**

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the currently connected remote GATT server device that has a service discovery operation outstanding.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_CONNECTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Cancel_Transaction

This function is provided to allow a mechanism for a GATT client to cancel a currently queued transaction.

Notes:

If the transaction ID specifies a transaction that has already been sent to the remote device then there is really way no way to cancel the transaction as it cannot be purged from the queue.

Prototype:

int BTPSAPI **GATT_Cancel_Transaction**(unsigned int BluetoothStackID,
unsigned int TransactionID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the transaction that is to be cancelled.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Query_Maximum_Supported_MTU

This function is provided to allow a mechanism for querying the maximum supported GATT MTU of the GATT layer.

Prototype:

int BTPSAPI **GATT_Query_Maximum_Supported_MTU**(unsigned int BluetoothStackID, Word_t *MTU)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MTU	Pointer to return the maximum supported MTU.

Return:

Zero if successful.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Change_Maximum_Supported_MTU

This function is provided to allow a mechanism for changing the maximum supported GATT MTU of the GATT layer.

Notes:

This API can only be used if there are NO active GATT connections.

Prototype:

int BTPSAPI **GATT_Change_Maximum_Supported_MTU** (unsigned int BluetoothStackID, Word_t MTU)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MTU	MTU to configure as the maximum supported for the GATT layer.

Return:

Zero if successful.

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

BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Query_Connection_MTU

This function is provided to allow a mechanism for querying the MTU of a specified connection.

Prototype:

int BTPSAPI **GATT_Query_Connection_MTU**(unsigned int BluetoothStackID, unsigned int ConnectionID, Word_t *MTU)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionID	Connection ID of the connection to query the MTU for.
MTU	Pointer to return the MTU for the connection.

Return:

Zero if successful.

An error code if negative; one of the following values:
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Query_Connection_ID

This function is provided to allow a mechanism for querying the connection identifier of a specified connection.

Prototype:

int BTPSAPI **GATT_Query_Connection_ID**(unsigned int BluetoothStackID, GATT_Connection_Type_t ConnectionType, BD_ADDR_t BD_ADDR, unsigned int *ConnectionID)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ConnectionType	Identifies the type of connection to query the Connection ID. This value must be one of the following: gctLE gctBR_EDR
BD_ADDR	Specifies the address of the remote Bluetooth device to query the Connection ID for.
ConnectionID	Pointer to return the Connection ID for the connection if successful.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Query_Transaction_Opcode

This function is provided to allow a mechanism for querying the Attribute Protocol Opcode of a specified transaction.

Prototype:

int BTPSAPI **GATT_Query_Transaction_Opcode**(unsigned int BluetoothStackID, unsigned int TransactionID, Byte_t *Opcode)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	Transaction ID of the transaction to query the Attribute Protocol Opcode for.
Opcode	Pointer to return the Opcode for the specified transaction.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Set_Queuing_Parameters

This function is provided to allow a mechanism of setting the queuing parameters that are used to limit the number of un-acknowledged packets that are queued internally.

Notes:

Setting both the MaximumNumberDataPackets and QueuedDataPacketsThreshold parameters to zero will disable the queuing mechanism. This means that the number of un-acknowledged packets that will only limited by the amount of available RAM.

Only un-acknowledged transactions are affected by the queuing. Acknowledged transactions are never affected. The following APIs (and only the following) are affected by the queuing mechanism:

- GATT_Write_Without_Response_Request
- GATT_Signed_Write_Without_Response_Request
- GATT_Handle_Value_Notification

Prototype:

int BTPSAPI **GATT_Set_Queuing_Parameters**(unsigned int BluetoothStackID, unsigned int MaximumNumberDataPackets, unsigned int QueuedDataPacketsThreshold, Boolean_t DiscardOldest)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
MaximumNumberDataPackets	Maximum number of un-acknowledged packets that may be queued internally.
QueuedDataPacketsThreshold	The lower threshold limit that the lower layer should call back to signify that it can queue more data packets for transmission.
DiscardOldest	Boolean that specifies if the oldest packets should be discarded when a buffer full condition occurs (if TRUE). If FALSE no packets will be discarded when the buffer is full. This can be useful to isochronous-like applications.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Get_Queuing_Parameters

This function is provided to allow a mechanism of getting the queuing parameters that are currently being used to limit the number of un-acknowledged packets that are queued internally.

Prototype:

int BTPSAPI **GATT_Get_Queuing_Parameters**(unsigned int BluetoothStackID, unsigned int *MaximumNumberDataPackets, unsigned int *QueuedDataPacketsThreshold, Boolean_t *DiscardOldest)

Parameters:

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

MaximumNumberDataPackets	Pointer to return the maximum number of un-acknowledged packets that may be queued internally.
QueuedDataPacketsThreshold	Pointer to return the the lower threshold limit that the lower layer should call back to signify that it can queue more data packets for transmission.
DiscardOldest	Pointer to return the boolean that specifies if the oldest packets should be discarded when a buffer full condition occurs (if TRUE). If FALSE no packets will be discarded when the buffer is full. This can be useful to isochronous-like applications.

Return:

Zero if successful.

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

BTGATT_ERROR_INVALID_TRANSACTION_ID
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Notes:

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

GATT_Query_Service_Range_Availability

This function is provided to allow a mechanism of determining if a specified handle range is available in the GATT database for a service to be registered in.

Prototype:

Boolean_t BTPSAPI **GATT_Query_Service_Range_Availability**(
unsigned int BluetoothStackID, GATT_Attribute_Handle_Group_t *ServiceHandleGroup)

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
ServiceHandleGroup	Pointer to a structure containing the start and end handle of a region in the GATT database to determine the availability of.

Return:

TRUE if the specified handle range in the GATT database is available (i.e. no other service is using any handles in the specified range).

FALSE if the specified region of the GATT database is not available.

Possible Events:

Notes:

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

2.2 Generic Attribute Profile Event Callback Prototypes

2.2.1 Connection Event Callback

The event callback function mentioned in the GATT_Initialize(), GATT_Register_Connection_Events(), and GATT_Connect() functions accept the callback function described by the following prototype.

GATT_Connection_Event_Callback_t

Prototype of callback function passed in the functions listed above.

Prototype:

void (BTPSAPI ***GATT_Connection_Event_Callback_t**)(unsigned int BluetoothStackID,
GATT_Connection_Event_Data_t *GATT_Connection_Event_Data,
unsigned long CallbackParameter)

Parameters:

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

```
typedef struct
{
  GATT_Connection_Event_Type_t  Event_Data_Type;
  Word_t                        Event_Data_Size;
  union
  {
    GATT_Device_Connection_Request_Data_t
          *GATT_Device_Connection_Request_Data;
    GATT_Device_Connection_Data_t
          *GATT_Device_Connection_Data;
    GATT_Device_Connection_Confirmation_Data_t
          *GATT_Device_Connection_Confirmation_Data;
    GATT_Device_Disconnection_Data_t
          *GATT_Device_Disconnection_Data;
  GATT_Device_Buffer_Empty_Data_t
          *GATT_Device_Buffer_Empty_Data;
```

```
            GATT_Server_Notification_Data_t
                  *GATT_Server_Notification_Data;
            GATT_Server_Indication_Data_t
                  *GATT_Server_Indication_Data;
            GATT_Device_Connection_MTU_Update_Data_t
                  *GATT_Device_Connection_MTU_Update_Data;
            GATT_Connection_Service_Database_Update_Data_t
                  *GATT_Connection_Service_Database_Update_Dat
                  a;
            GATT_Connection_Service_Changed_Read_Data_t
                  *GATT_Connection_Service_Changed_Read_Data;
            GATT_Connection_Service_Changed_Confirmation_Data
                  _t
                  *GATT_Connection_Service_Changed_Confirmatio
                  n_Data;
            GATT_Connection_Service_Changed_CCCD_Read_Data
                  _t
                  *GATT_Connection_Service_Changed_CCCD_Rea
                  d_Data;
            GATT_Connection_Service_Changed_CCCD_Update_Da
                  ta_t
                  *GATT_Connection_Service_Changed_CCCD_Up
                  date_Data;
         } Event_Data;
      } GATT_Connection_Event_Data_t;
```

where, Event_Data_Type is one of the enumerations of the event types listed in the table in section 2.3.1, and each data structure in the union is described with its event in that section as well.

CallbackParameter	User-defined parameter (e.g., tag value) that was defined in the callback registration.

Return:

Notes:

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

2.2.2 Server Event Callback

The event callback function mentioned in the GATT_Register_Service() function accepts the callback function described by the following prototype.

GATT_Server_Event_Callback_t

Prototype of callback function passed in the function listed above.

Prototype:

void (BTPSAPI * **GATT_Server_Event_Callback_t**)(unsigned int BluetoothStackID,
GATT_Server_Event_Data_t *GATT_Server_Event_Data,
unsigned long CallbackParameter)

Parameters:

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

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

```
typedef struct
{
  GATT_Server_Event_Type_t    Event_Data_Type;
  Word_t                      Event_Data_Size;
   union
  {
    GATT_Device_Connection_Data_t
                *GATT_Device_Connection_Data;
    GATT_Device_Disconnection_Data_t
                *GATT_Device_Disconnection_Data;
    GATT_Device_Buffer_Empty_Data_t
                *GATT_Device_Buffer_Empty_Data;
    GATT_Device_Connection_MTU_Update_Data_t
                *GATT_Device_Connection_MTU_Update_Data;
    GATT_Read_Request_Data_t
                *GATT_Read_Request_Data;
    GATT_Write_Request_Data_t
                *GATT_Write_Request_Data;
    GATT_Signed_Write_Request_Data_t
                *GATT_Signed_Write_Request_Data;
    GATT_Execute_Write_Request_Data_t
                *GATT_Execute_Write_Request_Data;
    GATT_Execute_Write_Confirmation_Data_t
                *GATT_Execute_Write_Confirmation_Data;
    GATT_Confirmation_Data_t
                *GATT_Confirmation_Data;
  } Event_Data;
} GATT_Server_Event_Data_t;
```

where, Event_Data_Type is one of the enumerations of the event types listed in the table in section 2.3.2, and each data structure in the union is described with its event in that section as well.

CallbackParameter User-defined parameter (e.g., tag value) that was defined in the callback registration.

Return:

Notes:

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

2.2.3 Client Event Callback

The event callback function mentioned in the client request functions accepts the callback function described by the following prototype.

GATT_Client_Event_Callback_t

Prototype of callback function passed in the client request functions.

Prototype:

void (BTPSAPI * **GATT_Client_Event_Callback_t**)(unsigned int BluetoothStackID,
GATT_Client_Event_Data_t *GATT_Client_Event_Data,
unsigned long CallbackParameter)

Parameters:

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

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

```
typedef struct
{
  GATT_Client_Event_Type_t   Event_Data_Type;
  Word_t                     Event_Data_Size;
   union
  {
  GATT_Request_Error_Data_t
     *GATT_Request_Error_Data;
  GATT_Service_Discovery_Response_Data_t
     *GATT_Service_Discovery_Response_Data;
  GATT_Service_Discovery_By_UUID_Response_Data_t
     *GATT_Service_Discovery_By_UUID_Response_Data;
  GATT_Included_Services_Discovery_Response_Data_t
     *GATT_Included_Services_Discovery_Response_Data;
  GATT_Characteristic_Discovery_Response_Data_t
     *GATT_Characteristic_Discovery_Response_Data;
  GATT_Characteristic_Descriptor_Discovery_Response_Data_t
     *GATT_Characteristic_Descriptor_Discovery_Response_Data;
  GATT_Read_Response_Data_t
     *GATT_Read_Response_Data;
  GATT_Read_By_UUID_Response_Data_t
     *GATT_Read_By_UUID_Response_Data;
  GATT_Read_Long_Response_Data_t
     *GATT_Read_Long_Response_Data;
```

```
      GATT_Read_Multiple_Response_Data_t
         *GATT_Read_Multiple_Response_Data;
      GATT_Write_Response_Data_t
         *GATT_Write_Response_Data;
      GATT_Prepare_Write_Response_Data_t
         *GATT_Prepare_Write_Response_Data;
      GATT_Execute_Write_Response_Data_t
         *GATT_Execute_Write_Response_Data;
      GATT_Exchange_MTU_Response_Data_t
         *GATT_Exchange_MTU_Response_Data;
      } Event_Data;
   } GATT_Client_Event_Data_t;
```

where, Event_Data_Type is one of the enumerations of the event types listed in the table in section 2.3.3, and each data structure in the union is described with its event in that section as well.

CallbackParameter — User-defined parameter (e.g., tag value) that was defined in the callback registration.

Return:

Notes:

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

2.2.4 Service Discovery Event Callback

The event callback function mentioned in the GATT_Start_Service_Discovery() function accepts the callback function described by the following prototype.

GATT_Service_Discovery_Event_Callback_t

Prototype of callback function passed in the GATT_Start_Service_Discovery() function.

Prototype:

```
void (BTPSAPI *GATT_Service_Discovery_Event_Callback_t)(
    unsigned int BluetoothStackID,
    GATT_Service_Discovery_Event_Data_t *GATT_Service_Discovery_Event_Data,
    unsigned long CallbackParameter)
```

Parameters:

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

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

```
typedef struct
{
  GATT_Service_Discovery_Event_Type_t  Event_Data_Type;
  Word_t                               Event_Data_Size;
   union
  {
  GATT_Service_Discovery_Indication_Data_t
     * GATT_Service_Discovery_Indication_Data;
  GATT_Service_Discovery_Complete_Data_t
     *GATT_Service_Discovery_Complete_Data;
  } Event_Data;
} GATT_Service_Discovery_Event_Data_t;
```

where, Event_Data_Type is one of the enumerations of the event types listed in the table in section 2.3.4, and each data structure in the union is described with its event in that section as well.

CallbackParameter — User-defined parameter (e.g., tag value) that was defined in the callback registration.

Return:

Notes:

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

2.3 Generic Attribute Profile Events

The Generic Attribute Profile contains events that are received based upon the type of callback (connection, server, and client request). The following sections detail those events.

2.3.1 Generic Attribute Profile Connection Events

The possible GATT connection events from the Bluetooth stack are listed in the table below and are described in the text which follows:

Event	Description
etGATT_Connection_Device_Connection_Request	Dispatched when a remote BR/EDR device is attempting a GATT connection to the local GATT server.
etGATT_Connection_Device_Connection	Dispatched when a remote Bluetooth device is connected to the GATT profile.
etGATT_Connection_Device_Connection_Confirmation	Dispatched when an out-going (client) BR/EDR connection is complete.

etGATT_Connection_Device_Disconnection	Dispatched when a remote Bluetooth device is disconnected from the GATT profile.
etGATT_Connection_Server_Indication	Dispatched when a remote Bluetooth device sends a GATT server indication to the local GATT profile.
etGATT_Connection_Server_Notification	Dispatched when a remote Bluetooth device sends a GATT server notification to the local GATT profile.
etGATT_Connection_Device_Connection_MTU_Update	Dispatched when the MTU for a remote Bluetooth LE device is changed.
etGATT_Connection_Service_Database_Update	Dispatched when the GATT database has been changed due to the addition/removal of a GATT Service.
etGATT_Connection_Service_Changed_Read_Request	Dispatched when a remote Bluetooth device attempts to read it's Service Changed value.
etGATT_Connection_Service_Changed_Confirmation	Dispatched when a confirmation is received from a GATT Service Changed indication that was sent by the local Bluetooth Device.
etGATT_Connection_Device_Buffer_Empty	Dispatched when the buffer for the specified device connection becomes empty.
etGATT_Connection_Service_Changed_CCCD_Read_Request	Dispatched when a Client attempts to read its unique Client Characteristic Configuration Descriptor (CCCD) value for the Service Changed characteristic.
etGATT_Connection_Service_Changed_CCCD_Update	Dispatched when a Client updates its unique CCCD value for the Service Changed characteristic.

etGATT_Connection_Device_Connection_Request

This event is dispatched when a remote BR/EDR device is requesting a connection to the local GATT profile server.

Notes:

This event is ONLY dispatched to the connection event callback function that was registered via the call to GATT_Initialize(). This event is also ONLY dispatched when the incoming BR/EDR connection mode is set to:

gimManualAccept

Return Structure:

```
typedef struct
{
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
} GATT_Device_Connection_Request_Data_t;
```

Event Parameters:

ConnectionType	Identifies the type of remote Bluetooth device that is requesting the connection. Currently this value will always be: gctBR_EDR
RemoteDevice	Specifies the address of the client Bluetooth device that has connected to the local GATT profile.

etGATT_Connection_Device_Connection

This event is dispatched whenever a remote LE or BR/EDR device is connected to the local GATT profile.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
  Word_t                    MTU;
} GATT_Device_Connection_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that is now connected to the local GATT profile.
MTU	Specifies the largest negotiated maximum transmission unit (MTU) that can be used when communicating over this connection.

etGATT_Connection_Device_Connection_Confirmation

This event is dispatched to signify the connection status of an out-going BR/EDR GATT connection.

Notes:

This event is ONLY dispatched to the connection event callback function that was registered via the call to the GATT_Connect() function. The event callback that is registered with this function will only receive this event.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  unsigned int              ConnectionStatus;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
  Word_t                    MTU;
} GATT_Device_Connection_Confirmation_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionStatus	Specifies the status of the connection. This value will be one of the following: GATT_CONNECTION_CONFIRMATION_STATUS_SUCCESS GATT_CONNECTION_CONFIRMATION_STATUS_CONNECTION_TIMEOUT GATT_CONNECTION_CONFIRMATION_STATUS_CONNECTION_REFUSED GATT_CONNECTION_CONFIRMATION_STATUS_UNKNOWN_ERROR
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device was the target of the original out-going connection.
MTU	Specifies the largest negotiated maximum transmission unit (MTU) that can be used when communicating over this connection.

etGATT_Connection_Device_Disconnection

This event is dispatched whenever a remote LE or BR/EDR device is no longer connected to the local GATT profile.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
} GATT_Device_Disconnection_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is no longer connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that is no longer connected to the local GATT profile.

etGATT_Connection_Server_Indication

This event is dispatched whenever a remote LE or BR/EDR device sends a GATT server indication to the local GATT profile.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      AttributeHandle;
  Word_t                      AttributeValueLength;
  Byte_t                     *AttributeValue;
} GATT_Server_Indication_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication. This value is used when sending a confirmation acknowledgement for the indication.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that has sent a server indication to the local GATT profile.

AttributeHandle	Attribute handle of the attribute value that is being indicated.
AttributeValueLength	Specifies the length (in bytes) of the AttributeValue buffer.
AttributeValue	Pointer to the buffer that contains the indicated data. The length of this data will be given by the attribute value length parameter.

etGATT_Connection_Server_Notification

This event is dispatched whenever a remote LE or BR/EDR device sends a GATT server notification to the local GATT profile.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      AttributeHandle;
  Word_t                      AttributeValueLength;
  Byte_t                     *AttributeValue;
} GATT_Server_Notification_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that has sent a server notification to the local GATT profile.
AttributeHandle	Attribute handle of the attribute value that is being notified.
AttributeValueLength	Specifies the length (in bytes) of the AttributeValue buffer.
AttributeValue	Pointer to the buffer that contains the notification data. The length of this data will be given by the attribute value length parameter.

etGATT_Connection_Device_Connection_MTU_Update

This event is dispatched whenever the MTU for a remote LE device is changed.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      MTU;
} GATT_Device_Connection_MTU_Update_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device who's MTU has been updated.
MTU	Specifies the largest negotiated maximum transmission unit (MTU) that can be used when communicating over this connection.

etGATT_Connection_Service_Database_Update

This event is dispatched whenever the GATT database on the local Bluetooth Device is modified due to the addition/removal of a GATT Service.

Return Structure:

```
typedef struct
{
  Boolean_t                   ServiceAdded;
  GATT_Service_Changed_Data_t ServiceChangedData;
} GATT_Connection_Service_Database_Update_Data_t;
```

Event Parameters:

ServiceAdded	Boolean which specifies if the local GATT database changed due to the addition (TRUE) or removal (FALSE) of a GATT Service.
ServiceChangedData	Specifies the region of the local GATT database that has been affected by the addition/removal of a GATT Service. This is structure defined as: typedef struct { Word_t Affected_Start_Handle; Word_t Affected_End_Handle; } **GATT_Service_Changed_Data_t**;

etGATT_Connection_Service_Changed_Read_Request

This event is dispatched whenever a remote LE or BR/EDR device is attempting to read it's GATT Service Changed value.

Return Structure:

```
typedef struct
{
  unsigned int                 ConnectionID;
  unsigned int                 TransactionID;
  GATT_Connection_Type_t       ConnectionType;
  BD_ADDR_t                    RemoteDevice;
} GATT_Connection_Service_Changed_Read_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the Service Changed read request. This value is used when responding to the Service Changed read request.
ConnectionType	Identifies the type of remote Bluetooth device that is performing the read. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that has sent the request.

etGATT_Connection_Service_Changed_Confirmation

This event is dispatched whenever the local GATT server received an acknowledgement for a Service Changed indication from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                 ConnectionID;
  unsigned int                 TransactionID;
  GATT_Connection_Type_t       ConnectionType;
  BD_ADDR_t                    RemoteDevice;
  Byte_t                       Status;
} GATT_Connection_Service_Changed_Confirmation_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication that generated the confirmation response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following:

gctLE
gctBR_EDR

RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
Status	Specifies the status of the confirmation response. This value will be one of the following: GATT_CONFIRMATION_STATUS_SUCCESS GATT_CONFIRMATION_STATUS_TIMEOUT

etGATT_Connection_Device_Buffer_Empty

This event is dispatched when the buffer for the specified device connection becomes empty.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t ConnectionType;
  BD_ADDR_t                   RemoteDevice;
} GATT_Device_Buffer_Empty_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device whose buffer is no longer full. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device whose buffer is no longer full.

etGATT_Connection_Service_Changed_CCCD_Read_Request

This event is dispatched when a remote client is attempting to read its unique Service Changed CCCD value.

Notes:

It is the responsibility of the application, when responding to this request, to respond with the unique CCCD for the client that is performing the request.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t ConnectionType;
  BD_ADDR_t                   RemoteDevice;
} GATT_Connection_Service_Changed_CCCD_Read_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the Service Changed CCCD read request. This value is used when responding to the Service Changed CCCD read request.
ConnectionType	Identifies the type of remote Bluetooth device that is performing the read. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that has sent the request.

etGATT_Connection_Service_Changed_CCCD_Update

This event is dispatched when a remote client has updated its unique CCCD value for the Service Changed characteristic.

Notes:

It is the responsibility of the application to store this value persistently, across connections, only for bonded devices.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      CCCD;
} GATT_Connection_Service_Changed_CCCD_Update_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device who has updated its CCCD. This value will be one of the following: gctLE gctBR_EDR

RemoteDevice	Specifies the address of the Bluetooth device who has updated its CCCD.
CCCD	Value of the CCCD for the Service Changed characteristic that the specified client has written.

2.3.2 Generic Attribute Profile Server Events

The possible GATT server events from the Bluetooth stack are listed in the table below and are described in the text which follows:

Event	Description
etGATT_Server_Device_Connection	Dispatched when a remote Bluetooth device is connected to the GATT profile.
etGATT_Server_Device_Disconnection	Dispatched when a remote Bluetooth device is disconnected from the GATT profile.
etGATT_Server_Device_Connection_MTU_Update	Dispatched when the MTU for a remote Bluetooth LE device is changed.
etGATT_Server_Read_Request	Dispatched when a read value request is received by the GATT server from a connected Bluetooth device.
etGATT_Server_Write_Request	Dispatched when a write value request is received by the GATT server from a connected Bluetooth device.
etGATT_Server_Signed_Write_Request	Dispatched when a signed write value request is received by the GATT server from a connected Bluetooth device.
etGATT_Server_Execute_Write_Request	Dispatched when an execute write request is received by the GATT server from a connected Bluetooth device.
etGATT_Server_Execute_Write_Confirmation	Dispatched to inform services when it is safe to commit prepared writes that have been processed.
etGATT_Server_Confirmation_Response	Dispatched when an indication acknowledgment is received from a connected Bluetooth device.
etGATT_Connection_Device_Buffer_Empty	Dispatched when the buffer for the specified device connection becomes empty.

etGATT_Server_Device_Connection

This event is dispatched whenever a remote LE or BR/EDR device is connected to the local GATT profile.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      MTU;
} GATT_Device_Connection_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that is now connected to the local GATT profile.
MTU	Specifies the largest negotiated maximum transmission unit (MTU) that can be used when communicating over this connection.

etGATT_Server_Device_Disconnection

This event is dispatched whenever a remote LE or BR/EDR device is no longer connected to the local GATT profile.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
} GATT_Device_Disconnection_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is no longer connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that is no longer connected to the local GATT profile.

etGATT_Server_Device_Connection_MTU_Update

This event is dispatched whenever the MTU for a remote LE device is changed.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
  Word_t                    MTU;
} GATT_Device_Connection_MTU_Update_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device who's MTU has been updated.
MTU	Specifies the largest negotiated maximum transmission unit (MTU) that can be used when communicating over this connection.

etGATT_Server_Read_Request

This event is dispatched whenever the local GATT server received a read request from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  unsigned int              TransactionID;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
  unsigned int              ServiceID;
  Word_t                    AttributeOffset;
  Word_t                    AttributeValueOffset;
} GATT_Read_Request_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication. This value is used when sending a response for the request.

ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
ServiceID	Specifies the unique service ID of the service that is receiving the request. This value is returned from a successful call to GATT_Register_Service.
AttributeOffset	Specifies the offset in the service table, which was registered in a successful call to GATT_Register_Service, of the attribute that the request is being made to.
AttributeValueOffset	Specifies the offset into the attribute value that the request is being made to.

etGATT_Server_Write_Request

This event is dispatched whenever the local GATT server received a write request from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  unsigned int              TransactionID;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
  unsigned int              ServiceID;
  Word_t                    AttributeOffset;
  Word_t                    AttributeValueLength;
  Word_t                    AttributeValueOffset;
  Byte_t                   *AttributeValue;
  Boolean_t                 DelayWrite;
} GATT_Write_Request_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication. This value is used when sending a response for the request.
ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following: gctLE gctBR_EDR

RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
ServiceID	Specifies the unique service ID of the service that is receiving the request. This value is returned from a successful call to GATT_Register_Service.
AttributeOffset	Specifies the offset in the service table, which was registered in a successful call to GATT_Register_Service, of the attribute that the request is being made to.
AttributeValueLength	Specifies the length of the data contained in the write request.
AttributeValueOffset	Specifies the offset into the attribute value that the request is being made to.
AttributeValue	Pointer to the buffer that contains the data to write. The length of this data will be given by the attribute value length parameter.
DelayWrite	Boolean flag that specifies whether or not to queue (TRUE) the write request, or to commit the write request (FALSE) immediately. If the Boolean flag specifies that the write request should be queued (TRUE) then the application must wait for the following event to be received with a successful status before committing the write: etGATT_Server_Confirmation_Response The status of the above event will be the following if it is successful and all of the queued writes for the specified connection must be committed: GATT_EXECUTE_WRITE_CONFIRMATION_STATUS_NO_ERROR

etGATT_Server_Signed_Write_Request

This event is dispatched whenever the local GATT server received a signed write request from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                    ConnectionID;
  unsigned int                    TransactionID;
  GATT_Connection_Type_t          ConnectionType;
  BD_ADDR_t                       RemoteDevice;
  unsigned int                    ServiceID;
  Word_t                          AttributeOffset;
  Word_t                          AttributeValueLength;
  Byte_t                         *AttributeValue;
  ATT_Authentication_Signature_t  AuthenticationSignature;
} GATT_Signed_Write_Request_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication. This value is used when sending a response for the request.
ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
ServiceID	Specifies the unique service ID of the service that is receiving the request. This value is returned from a successful call to GATT_Register_Service.
AttributeOffset	Specifies the offset in the service table, which was registered in a successful call to GATT_Register_Service, of the attribute that the request is being made to.
AttributeValueLength	Specifies the length of the data contained in the write request.
AttributeValue	Pointer to the buffer that contains the data to write. The length of this data will be given by the attribute value length parameter.
AuthenticationSignature	Structure which contains the signature that was contained in the write request.

etGATT_Server_Execute_Write_Request

This event is dispatched whenever the local GATT server received an execute write request from a remote LE or BR/EDR device. This is used to commit/cancel queued writes.

Return Structure:

```
typedef struct
{
  unsigned int                 ConnectionID;
  unsigned int                 TransactionID;
  GATT_Connection_Type_t       ConnectionType;
  BD_ADDR_t                    RemoteDevice;
  unsigned int                 ServiceID;
  Boolean_t                    CancelWrite;
} GATT_Execute_Write_Request_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication. This value is used when sending a response for the request.
ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
ServiceID	Specifies the unique service ID of the service that is receiving the request. This value is returned from a successful call to GATT_Register_Service.
CancelWrite	Boolean flag that specifies if all queued writes for this connection should be canceled (TRUE) or possibly committed (FALSE). If the Boolean flag specifies that all queued writes should be canceled (TRUE) for this connection then the write queue for this connection should be flushed. If the flag specifies that the writes could possibly be committed (FALSE) then the application must wait for the following event to be received with a successful status before committing the write: etGATT_Server_Confirmation_Response The status of the above event will be the following if it is successful and all of the queued writes for the specified connection must be committed: GATT_EXECUTE_WRITE_CONFIRMATION_STATUS_NO_ERROR

etGATT_Server_Execute_Write_Confirmation

This event is dispatched whenever the local GATT server received an execute write confirmation from a remote LE or BR/EDR device. This is used to commit/cancel queued writes.

Return Structure:

```
typedef struct
{
  unsigned int                 ConnectionID;
  unsigned int                 TransactionID;
  GATT_Connection_Type_t       ConnectionType;
  BD_ADDR_t                    RemoteDevice;
  unsigned int                 ServiceID;
  Byte_t                       Status;
} GATT_Execute_Write_Confirmation_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication. This value is used when sending a response for the request.
ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
ServiceID	Specifies the unique service ID of the service that is receiving the request. This value is returned from a successful call to GATT_Register_Service.
Status	Specifies the status of the confirmation. This value will be one of the following: GATT_EXECUTE_WRITE_CONFIRMATION_STATUS_NO_ERROR GATT_EXECUTE_WRITE_CONFIRMATION_STATUS_ERROR If the status specifies that no error has occurred then all queued writes for the specified connection must be committed in the order that they were received. If an error occurred than all queued writes for the specified connection must be canceled.

etGATT_Server_Confirmation_Response

This event is dispatched whenever the local GATT server received an indication acknowledgement from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Byte_t                      Status;
  Word_t                      BytesWritten;
} GATT_Confirmation_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the indication that generated the confirmation response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the request. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the request.
Status	Specifies the status of the confirmation response. This value will be one of the following: GATT_CONFIRMATION_STATUS_SUCCESS GATT_CONFIRMATION_STATUS_TIMEOUT
BytesWritten	If the indication was successful, this parameter will contain the number of data bytes that were actually contained in the indication that was sent over the air to the remote device.

etGATT_Server_Device_Buffer_Empty

This event is dispatched when the buffer for the specified device connection becomes empty.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
} GATT_Device_Buffer_Empty_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.

ConnectionType	Identifies the type of remote Bluetooth device whose buffer is no longer full. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device whose buffer is no longer full.

2.3.3 Generic Attribute Profile Client Events

The possible GATT client events from the Bluetooth stack are listed in the table below and are described in the text which follows:

Event	Description
etGATT_Client_Error_Response	Dispatched when an error response is received, in response to a request, from a connected Bluetooth device.
etGATT_Client_Service_Discovery_Response	Dispatched when a service discovery response is received from a connected Bluetooth device.
etGATT_Client_Service_Discovery_By_UUID_Response	Dispatched when a service discovery by UUID response is received from a connected Bluetooth device.
etGATT_Client_Included_Services_Discovery_Response	Dispatched when an included services discovery response is received from a connected Bluetooth device.
etGATT_Client_Characteristic_Discovery_Response	Dispatched when a characteristic discovery response is received from a connected Bluetooth device.
etGATT_Client_Characteristic_Descriptor_Discovery_Response	Dispatched when a characteristic descriptor discovery response is received from a connected Bluetooth device.
etGATT_Client_Read_Response	Dispatched when a read response is received from a connected Bluetooth device.
etGATT_Client_Read_Long_Response	Dispatched when a read long response is received from a connected Bluetooth device.
etGATT_Client_Read_By_UUID_Response	Dispatched when a read by UUID response is received from a connected Bluetooth device.

etGATT_Client_Read_Multiple_Response	Dispatched when a read multiple response is received from a connected Bluetooth device.
etGATT_Client_Write_Response	Dispatched when a write response is received from a connected Bluetooth device.
etGATT_Client_Prepare_Write_Response	Dispatched when an prepare write response is received from a connected Bluetooth device.
etGATT_Client_Execute_Write_Response	Dispatched when an execute write response is received from a connected Bluetooth device.
etGATT_Client_Exchange_MTU_Response	Dispatched when an exchange MTU response is received from a connected Bluetooth LE device.

etGATT_Client_Error_Response

This event is dispatched whenever the local GATT client received an eror response from a remote LE or BR/EDR device or a request times out.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  GATT_Request_Error_Type_t   ErrorType;
  Byte_t                      RequestOpCode;
  Word_t                      RequestHandle;
  Byte_t                      ErrorCode;
} GATT_Request_Error_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.

ErrorType	Identifies the type error that occurred. This value will be one of the following: retErrorResponse retProtocolTimeout retPrepareWriteDataMismatch
RequestOpCode	Identifies the opcode of the request that caused the error.
RequestHandle	Identifies the handle of the attribute whose access, in a request, caused the error. This value is only valid if the ErrorType parameter is the following value: retErrorResponse
ErrorCode	Identifies the error code that was received. This value is only valid if the ErrorType parameter is the following value: retErrorResponse If valid this value will be one of the following: ATT_PROTOCOL_ERROR_CODE_INVALID_HANDLE ATT_PROTOCOL_ERROR_CODE_READ_NOT_PERMITTED ATT_PROTOCOL_ERROR_CODE_WRITE_NOT_PERMITTED ATT_PROTOCOL_ERROR_CODE_INVALID_PDU ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_AUTHENTICATION ATT_PROTOCOL_ERROR_CODE_REQUEST_NOT_SUPPORTED ATT_PROTOCOL_ERROR_CODE_INVALID_OFFSET ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_AUTHORIZATION ATT_PROTOCOL_ERROR_CODE_PREPARE_QUEUE_FULL ATT_PROTOCOL_ERROR_CODE_ATTRIBUTE_NOT_FOUND ATT_PROTOCOL_ERROR_CODE_ATTRIBUTE_NOT_LONG ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_ENCRYPTION_KEY_SIZE ATT_PROTOCOL_ERROR_CODE_INVALID_ATTRIBUTE_VALUE_LENGTH ATT_PROTOCOL_ERROR_CODE_UNLIKELY_ERROR ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_ENCRYPTION ATT_PROTOCOL_ERROR_CODE_UNSUPPORTED_GROUP_TYPE

ATT_PROTOCOL_ERROR_CODE_INSUFFICIENT_
RESOURCES

etGATT_Client_Service_Discovery_Response

This event is dispatched whenever the local GATT client received a service discovery response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                  ConnectionID;
  unsigned int                  TransactionID;
  GATT_Connection_Type_t        ConnectionType;
  BD_ADDR_t                     RemoteDevice;
  Word_t                        NumberOfServices;
  GATT_Service_Information_t *ServiceInformationList;
} GATT_Service_Discovery_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
NumberOfServices	Specifies number of services contained in the response.
ServiceInformationList	Pointer to array that contains information on each service received in the response. The number of entries in the array is specified by the NumberOfServices parameter. The structure is defined as follows:

```
typedef struct
{
  Word_t            Service_Handle;
  Word_t            End_Group_Handle;
  GATT_UUID_t       UUID;
} GATT_Service_Information_t;
```

etGATT_Client_Service_Discovery_By_UUID_Response

This event is dispatched whenever the local GATT client received a service discovery by UUID response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                          ConnectionID;
  unsigned int                          TransactionID;
  GATT_Connection_Type_t                ConnectionType;
  BD_ADDR_t                             RemoteDevice;
  Word_t                                NumberOfServices;
  GATT_Service_Information_By_UUID_t *ServiceInformationList;
} GATT_Service_Discovery_By_UUID_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
NumberOfServices	Specifies number of services contained in the response.
ServiceInformationList	Pointer to array that contains information on each service received in the response. The number of entries in the array is specified by the NumberOfServices parameter. The structure is defined as follows: typedef struct { Word_t Service_Handle; Word_t End_Group_Handle; } **GATT_Service_Information_By_UUID_t**;

etGATT_Client_Included_Services_Discovery_Response

This event is dispatched whenever the local GATT client received an included services discovery response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                  ConnectionID;
  unsigned int                  TransactionID;
  GATT_Connection_Type_t        ConnectionType;
  BD_ADDR_t                     RemoteDevice;
  Word_t                        NumberOfIncludes;
  GATT_Include_Information_t *IncludeInformationList;
} GATT_Included_Services_Discovery_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
NumberOfIncludes	Specifies number of included services contained in the response.
IncludeInformationList	Pointer to array that contains information on each included service received in the response. The number of entries in the array is specified by the NumberOfIncludes parameter. The structure is defined as follows:

```
typedef struct
{
  Word_t        Include_Attribute_Handle;
  Word_t        Included_Service_Handle;
  Word_t        Included_Service_End_Group_Handle;
  Boolean_t     UUID_Valid;
  GATT_UUID_t   UUID;
} GATT_Include_Information_t;
```

etGATT_Client_Characteristic_Discovery_Response

This event is dispatched whenever the local GATT client received a characteristic discovery response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                  ConnectionID;
  unsigned int                  TransactionID;
  GATT_Connection_Type_t        ConnectionType;
  BD_ADDR_t                     RemoteDevice;
  Word_t                        NumberOfCharacteristics;
  GATT_Characteristic_Entry_t *CharacteristicEntryList;
} GATT_Characteristic_Discovery_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
NumberOfCharacteristics	Specifies number of characteristics contained in the response.
CharacteristicEntryList	Pointer to array that contains information on each characteristic received in the response. The number of entries in the array is specified by the NumberOfCharacteristics parameter. Each member in this array is defined by the following structure:

```
typedef struct
{
  Word_t                        DeclarationHandle;
  GATT_Characteristic_Value_t   CharacteristicValue;
} GATT_Characteristic_Entry_t;
```

The CharacteristicValue in each entry is defined as follows:

```
typedef struct
{
  Byte_t        CharacteristicProperties;
  Word_t        CharacteristicValueHandle;
  GATT_UUID_t   CharacteristicUUID;
} GATT_Characteristic_Value_t;
```

etGATT_Client_Characteristic_Descriptor_Discovery_Response

This event is dispatched whenever the local GATT client received a characteristic descriptor discovery response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                                ConnectionID;
  unsigned int                                TransactionID;
  GATT_Connection_Type_t                      ConnectionType;
  BD_ADDR_t                                   RemoteDevice;
  Word_t                                      NumberOfCharacteristicDescriptors;
  GATT_Characteristic_Descriptor_Entry_t *CharacteristicDescriptorEntryList;
} GATT_Characteristic_Descriptor_Discovery_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
NumberOfCharacteristicDescriptors	Specifies number of characteristic descriptors contained in the response.

CharacteristicDescriptorEntryList	Pointer to array that contains information on each characteristic descriptor received in the response. The number of entries in the array is specified by the NumberOfCharacteristicDescriptors parameter. Each member in this array is defined by the following structure:
	`typedef struct`
	`{`
	`  Word_t Handle;`
	`  GATT_UUID_t UUID;`
	`} GATT_Characteristic_Descriptor_Entry_t;`

etGATT_Client_Read_Response

This event is dispatched whenever the local GATT client received a read response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                 ConnectionID;
  unsigned int                 TransactionID;
  GATT_Connection_Type_t       ConnectionType;
  BD_ADDR_t                    RemoteDevice;
  Word_t                       AttributeValueLength;
  Byte_t                      *AttributeValue;
} GATT_Read_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
AttributeValueLength	Specifies the length of the data contained in the read response.
AttributeValue	Pointer to a buffer that contains the data that was read. The length of this data will be given by the attribute value length parameter.

etGATT_Client_Read_Long_Response

This event is dispatched whenever the local GATT client received a read long response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                 ConnectionID;
  unsigned int                 TransactionID;
  GATT_Connection_Type_t       ConnectionType;
  BD_ADDR_t                    RemoteDevice;
  Word_t                       AttributeValueLength;
  Byte_t                      *AttributeValue;
} GATT_Read_Long_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.

TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
AttributeValueLength	Specifies the length of the data contained in the read long response.
AttributeValue	Pointer to a buffer that contains the data that was read. The length of this data will be given by the attribute value length parameter.

etGATT_Client_Read_By_UUID_Response

This event is dispatched whenever the local GATT client received a read by UUID response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      NumberOfAttributes;
  GATT_Read_Event_Entry_t     *AttributeList;
} GATT_Read_By_UUID_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
NumberOfAttributes	Specifies the number of attributes contained in the response.
AttributeList	Pointer to an array that contains information on each attribute whose value was returned in the response. The number of entries

in the array is specified by the NumberOfAttributes parameter. Each member in this array is defined by the following structure:

```
typedef struct
{
  Word_t      AttributeHandle;
  Word_t      AttributeValueLength;
  Byte_t     *AttributeValue;
} GATT_Read_Event_Entry_t;
```

etGATT_Client_Read_Multiple_Response

This event is dispatched whenever the local GATT client received a read multiple response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  Word_t                      AttributeValuesLength;
  Byte_t                     *AttributeValues;
} GATT_Read_Multiple_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
AttributeValuesLength	Specifies the total length of the data contained in the read multiple response. Note it is up to the application to know the length of each attribute value that is contained in the response.
AttributeValue	Pointer to a buffer that contains the data that was read. The length of this data will be given by the attribute values length parameter.

etGATT_Client_Write_Response

This event is dispatched whenever the local GATT client received a write response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  unsigned int                BytesWritten;
} GATT_Write_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
BytesWritten	Specifies the numbers of bytes that were written to the remote device.

etGATT_Client_Prepare_Write_Response

This event is dispatched whenever the local GATT client received a prepare write response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int                ConnectionID;
  unsigned int                TransactionID;
  GATT_Connection_Type_t      ConnectionType;
  BD_ADDR_t                   RemoteDevice;
  unsigned int                BytesWritten;
  Word_t                      AttributeHandle;
  Word_t                      AttributeValueOffset;
  Word_t                      AttributeValueLength;
  Byte_t                     *AttributeValue;
} GATT_Prepare_Write_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.

ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device that sent the response.
BytesWritten	Specifies the numbers of bytes that were written to the remote device.
AttributeHandle	The attribute handle that was specified in the request that generated this response. Note, this parameter is echoed by the remote device from the request that was sent to the remote.
AttributeValueOffset	The offset into the attribute value that was specified in the request that generated this response. Note, this parameter is echoed by the remote device from the request that was sent to the remote device.
AttributeValueLength	The length of the data in the prepare write. Note, this parameter is echoed by the remote device from the request that was sent to the remote device.
AttributeValue	Pointer to a buffer that contains the data that was sent to the remote device. Note, this parameter is echoed by the remote device from the request that was sent to the remote device.

etGATT_Client_Execute_Write_Response

This event is dispatched whenever the local GATT client received an execute write response from a remote LE or BR/EDR device.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  unsigned int              TransactionID;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
} GATT_Execute_Write_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
TransactionID	Specifies the unique transaction ID of the request that generated the response.
ConnectionType	Identifies the type of remote Bluetooth device that sent the response. This value will be one of the following: gctLE gctBR_EDR

RemoteDevice	Specifies the address of the Bluetooth device that sent the response.

etGATT_Client_Exchange_MTU_Response

This event is dispatched whenever an exchange MTU response is received from a connected LE device. The new MTU for the connection to the specified LE device will be the smaller of the ServerMTU in this event, and the RequestedMTU specified in the call to GATT_Exchange_MTU_Request.

Return Structure:

```
typedef struct
{
  unsigned int              ConnectionID;
  unsigned int              TransactionID;
  GATT_Connection_Type_t    ConnectionType;
  BD_ADDR_t                 RemoteDevice;
  Word_t                    ServerMTU;
} GATT_Exchange_MTU_Response_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection.
ConnectionType	Identifies the type of remote Bluetooth device that is now connected. This value will be one of the following: gctLE gctBR_EDR
RemoteDevice	Specifies the address of the Bluetooth device who's MTU has been updated.
MTU	Specifies the largest negotiated maximum transmission unit (MTU) that can be used when communicating over this connection.

2.3.4 Generic Attribute Profile Service Discovery Events

The possible GATT service discovery events from the Bluetooth stack are listed in the table below and are described in the text which follows:

Event	Description
etGATT_Service_Discovery_Indication	Dispatched when a service on the remote device has been discovered successfully.
etGATT_Service_Discovery_Complete	Dispatched when a service discovery operation has completed.

etGATT_Service_Discovery_Indication

This event is dispatched whenever a service is discovered on a remote device in response to a previously started service discovery operation.

Return Structure:

```
typedef struct
{
  unsigned int                    ConnectionID;
  GATT_Service_Information_t      ServiceInformation;
  unsigned int                    NumberOfIncludedService;
  GATT_Service_Information_t     *IncludedServiceList;
  unsigned int                    NumberOfCharacteristics;
  GATT_Characteristic_Information_t *CharacteristicInformationList;
} GATT_Service_Discovery_Indication_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection that the service was discovered on.
ServiceInformation	Structure that contains the UUID, Start Handle and End Handle of the discovered service.
NumberOfIncludedService	Contains the number of Included Services that are contained in the Included Service List.
IncludedServiceList	List of Included Services that are contained in the discovered service.
NumberOfCharacteristics	Contains the number of Characteristics that are contained in the Characteristic Information List.
CharacteristicInformationList	List of Characteristics that are contained in the discovered service.

etGATT_Service_Discovery_Complete

This event is dispatched whenever the previously started service discovery operation on a remote device completes.

Return Structure:

```
typedef struct
{
  unsigned int   ConnectionID;
  Byte_t         Status;
} GATT_Service_Discovery_Complete_Data_t;
```

Event Parameters:

ConnectionID	Identifier that uniquely identifies the actual connection that the service discovery has completed for.

Status	Contains the status of the service discovery operation. Will be one of the following values: GATT_SERVICE_DISCOVERY_STATUS_SUCCESS GATT_SERVICE_DISCOVERY_STATUS_RESPONSE_ERROR GATT_SERVICE_DISCOVERY_STATUS_RESPONSE_TIMEOUT GATT_SERVICE_DISCOVERY_STATUS_UNKNOWN_ERROR

3. File Distributions

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

File	Contents/Description
ATTTypes.h	Bluetooth Attribute Protocol type definitions
GATTType.h	Bluetooth Generic Attribute Profile type definitions
GATTAPI.h	Bluetooth Generic Attribute Profile API definitions
SS1BTGAT.h	Bluetooth Generic Attribute Profile Include file