Current Time Service (CTS)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 4.0.1
January 10, 2013

Louisville, KY www.stonestreetone.com

Table of Contents

1. INTRODUCTION 3

1.1 Scope 3

1.2 Applicable Documents 4

1.3 Acronyms and Abbreviations 4

2. CTS PROGRAMMING INTERFACE 5

2.1 Current Time Service Commands 5

CTS_Initialize_Service 6
CTS_Cleanup_Service 7
CTS_Current_Time_Read_Request_Response 7
CTS_Current_Time_Read_Request_Error_Response 9
CTS_Set_Local_Time_Information 10
CTS_Query_Local_Time_Information 12
CTS_Reference_Time_Information_Read_Request_Response 12
CTS_Reference_Time_Information_Request_Error_Response 14
CTS_Read_Client_Configuration_Response 14
CTS_Notify_Current_Time 15
CTS_Decode_Current_Time 17
CTS_Decode_Local_Time_Information 18
CTS_Decode_Reference_Time_Information 20

2.2 Current Time Service Event Callback Prototypes 21

2.2.1 SERVER EVENT CALLBACK 21
CTS_Event_Callback_t 21

2.3 Current Time Service Events 22

2.3.1 CURRENT TIME SERVICE SERVER EVENTS 22
etCTS_Read_Client_Configuartion_Request 23
etCTS_Client_Configuration_Update 24
etCTS_ Read_Current_Time_Request 24
etCTS_ Read_Reference_Time_Information_Request 25

3. FILE DISTRIBUTIONS 27

1. Introduction

Bluetopia®+LE is Stonestreet One's Bluetooth protocol stack that supports the adopted Bluetooth low energy specification. Stonestreet One's upper level protocol stack that supports Single Mode devices is Bluetopia®+LE Single. 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), ATT (Attribute Protocol) Link Layers, the GAP (Generic Attribute Profile) Layer and the Genetic Attribute Protocol (GATT) Layer. In addition to basic functionality of these layers, the Bluetooth Protocol Stack by Stonestreet One provides implementations of the Device Information Service (DIS), CTS (Current Time Service), 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 Current Time Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Current Time Service library.

1.1 Scope

This reference manual provides information on the CTS API. This API is available on the full range of platforms supported by Stonestreet One:

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

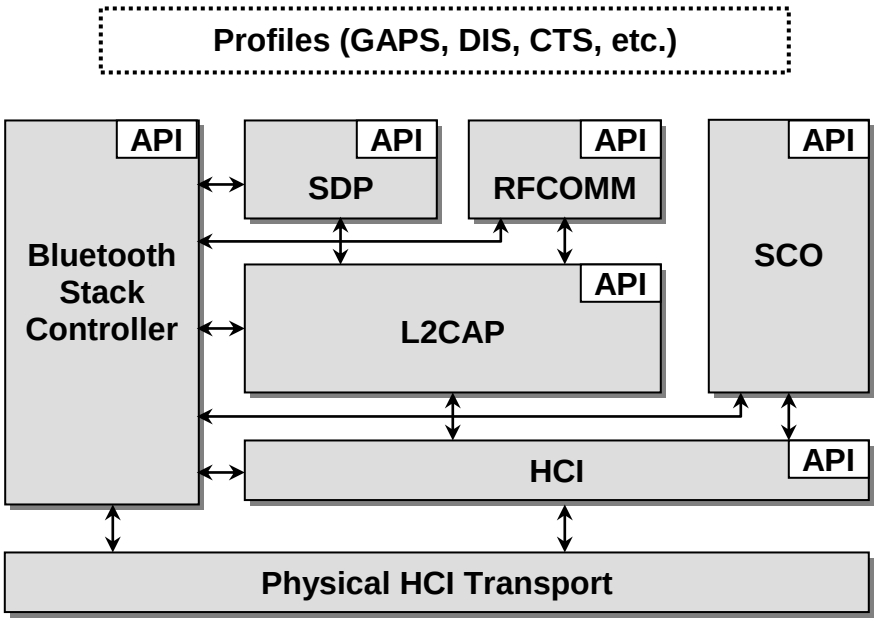

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

1.2 Applicable Documents

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

1. *Specification of the Bluetooth System, Volume 1, Architecture and Terminology Overview*, version 4.0, June 30, 2010.
2. *Specification of the Bluetooth System, Volume 6, Core System Package [Low Energy Controller Volume]*, version 4.0, June 30, 2010.
3. *Bluetopia® Protocol Stack, Application Programming Interface Reference Manual*, version 4.0.1, January 10, 2013.
4. *Bluetooth Current Time Service Specification*, version v10r00, April 3, 2012.

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

1.3 Acronyms and Abbreviations

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

Term	Meaning
API	Application Programming Interface
ATT	Attribute Protocol
BD_ADDR	Bluetooth Device Address
BT	Bluetooth
CTS	Current Time Service
GAPS	Generic Access Profile Service
GATT	Generic Attribute Protocol
HCI	Host Controller Interface
HS	High Speed
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy
LSB	Least Significant Bit
MSB	Most Significant Bit

2. CTS Programming Interface

The Current Time Service, CTS, programming interface defines the protocols and procedures to be used to implement CTS capabilities for both Server and Client services. The CTS commands are listed in section 2.1, the event callback prototypes are described in section 2.2, the CTS events are itemized in section 2.3. The actual prototypes and constants outlines in this section can be found in the **CTSAPI.h** header file in the Bluetopia distribution.

2.1 Current Time Service Commands

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

Server Commands	
Function	**Description**
CTS_Initialize_Service	Opens a CTS Server.
CTS_Cleanup_Service	Closes an opened CTS Server.
CTS_Current_Time_Read_Request_Response	Responds to a CTS Read Current Time Request.
CTS_Current_Time_Read_Request_Error_ Response	Responds to a CTS Read Current Time Request when an error occurs.
CTS_Set_Local_Time_Information	Sets the Local Time information on the specifited CTS Instance
CTS_Query_Local_Time_Information	Queries the Local Time Information on the specified CTS Instance.
CTS_Reference_Time_Information_Read_Request _Response	Responds to a CTS Read Reference Time Information Request
CTS_Reference_Time_Information_Read_Request _Error_Response	Responds to a CTS Read Reference Time Information Request when an error occurs.
CTS_Read_Client_Configuration_Response	Responds to a CTS Read Client Configuation Request.
CTS_Notify_Current_Time	Sends a Current Time Notification to a specified remote device.
CTS_Decode_Current_Time	Parses a value received for a remote CTS Server interpreting it as a Current Time characteristic.
CTS_Decode_Local_Time_Information	Parses a value received from a remote CTS Server interpresting it as a Local Time information characteristic.

CTS_Decode_Reference_Time_Information	Parses a value received from a remote CTS Server interpreting it as a Refernece Time Information characteristic.

CTS_Initialize_Service

This function opens a CTS Server on a specified Bluetooth Stack.

Notes:

1. Only one CTS Server, per Bluetooth Stack ID, may be open at a time.
2. All Client Requests will be dispatched to the EventCallback function that is specified by the second parameter to this function.

Prototype:

int BTPSAPI **CTS_Initialize_Service**(unsigned int BluetoothStackID, CTS_Event_Callback_t EventCallback, unsigned long CallbackParameter, unsigned int *ServiceID);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
EventCallback	Callback function that is registered to receive events that are associated with the specified service.
CallbackParameter	A user-defined parameter that will be passed back to the user in the callback function.
ServiceID	Unique GATT Service ID of the registered CTS service returned from GATT_Register_Service API.

Return:

Positive non-zero if successful. The return value will be the Service ID of CTS Server that was successfully opened on the specified Bluetooth Stack ID. This is the value that should be used in all subsequent function calls that require Instance ID.

Negative if an error occurred. Possible values are:

CTS_ERROR_INSUFFICIENT_RESOURCES
CTS_ERROR_SERVICE_ALREADY_REGISTERED
CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_SERVICE_TABLE_FORMAT
BTGATT_ERROR_INSUFFICIENT_RESOURCES
BTGATT_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_NOT_INITIALIZED

Notes:

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

CTS_Cleanup_Service

This function is responsible for cleaning up and freeing all resources associated with a Current Time Service Instance. After this function is called, no other Current Time Service function can be called until after a successful call to the CTS_Initialize_Service() function is performed.

Prototype:

int BTPSAPI **CTS_Cleanup_Service**(unsigned int BluetoothStackID,
unsigned int InstanceID);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This InstanceID was returned from the CTS_Initialize_Service().

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_PARAMETER
CTS_ERROR_INVALID_INSTANCE_ID

Notes:

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

CTS_Current_Time_Read_Request_Response

The following function is responsible for responding to a CTS Read Current Time Request.

Prototype:

int BTPSAPI **CTS_Current_Time_Read_Request_Response**(unsigned int BluetoothStackID, unsigned int TransactionID, CTS_Current_Time_Data_t *Current_Time);

Parameters:

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

TransactionID The Transaction ID of the original read request.

CurrentTime A pointer to the current time. The Current Time Data structure contains the following:

```
typedef struct
{
    CTS_Exact_Time_Data_t       Exact_Time;
    Byte_t                      Adjust_Reason_Mask;
} CTS_Current_Time_Data_t;
```

Where the Exact Time Data Structure, Day Date Time Data Structure, Date Time Data Structure and Week Day Type Enum are defined as follows:

```
typedef struct
{
    CTS_Day_Date_Time_Data_t    Day_Date_Time;
    Byte_t                      Fractions256;
} CTS_Exact_Time_Data_t;
```

```
typedef  struct
{
    CTS_Date_Time_Data_t        Date_Time;
    CTS_Week_Day_Type_t         Day_Of_Week;
} CTS_Day_Date_Time_Data_t;
```

```
typedef struct
{
    Word_t                      Year;
    CTS_Month_Of_Year_Type_t    Month;
    Byte_t                      Day;
    Byte_t                      Hours;
    Byte_t                      Minutes;
    Byte_t                      Seconds;
} CTS_Date_Time_Data_t;
```

```
typedef enum
{
    wdUnknown,
    wdMonday,
    wdTuesday,
    wdWednesday,
    wdThursday,
    wdFriday,
    wdSaturday,
    wdSunday
} CTS_Week_Day_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown/XXX

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.

CTS_Current_Time_Read_Request_Error_Response

The following function is responsible for responding to a CTS Read Current Time Request when an error occured.

Prototype:

int BTPSAPI **CTS_Current_Time_Read_Request_Error_Response**(unsigned int BluetoothStackID, unsigned int TransactionID, Byte_t ErrorCode);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	The Transaction ID of the original read request.
ErrorCode	ErrorCode is used to determine if the Request is being accepted by the server or if an error response is issued instead. This function returns a zero if successful or a negative return error code if an error occurs

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown/XXX

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.

CTS_Set_Local_Time_Information

This function is responsible for setting the Local Time Informatin on the specified CTS Instance.

Prototype:

int BTPSAPI **CTS_Set_Local_Time_Information**(unsigned int BluetoothStackID, unsigned int InstanceID, CTS_Local_Time_Information_Data_t *Local_Time);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This InstanceID was returned from the CTS_Initialize_Service().
LocalTime	The value to be entered as the Local Time. The Local Time Information Data structure is as follows:

```
typedef  struct
{
    CTS_Time_Zone_Type_t    Time_Zone;
    CTS_DST_Offset_Type_t   Daylight_Saving_Time;
} CTS_Local_Time_Information_Data_t;
```

With the Time Zone Type enum and DST Offset enum defined as follows:

```
typedef enum
{
    tzUTCMinus1200,
    tzUTCMinus1100,
    tzUTCMinus1000,
    tzUTCMinus930,
    tzUTCMinus900,
    tzUTCMinus800,
    tzUTCMinus700,
    tzUTCMinus600,
    tzUTCMinus500,
    tzUTCMinus430,
    tzUTCMinus400,
    tzUTCMinus330,
    tzUTCMinus300,
    tzUTCMinus200,
    tzUTCMinus100,
```

```
    tzUTCPlus000,
    tzUTCPlus100,
    tzUTCPlus200,
    tzUTCPlus300,
    tzUTCPlus330,
    tzUTCPlus400,
    tzUTCPlus430,
    tzUTCPlus500,
    tzUTCPlus530,
    tzUTCPlus545,
    tzUTCPlus600,
    tzUTCPlus630,
    tzUTCPlus700,
    tzUTCPlus800,
    tzUTCPlus845,
    tzUTCPlus900,
    tzUTCPlus930,
    tzUTCPlus1000,
    tzUTCPlus1030,
    tzUTCPlus1100,
    tzUTCPlus1130,
    tzUTCPlus1200,
    tzUTCPlus1245,
    tzUTCPlus1300,
    tzUTCPlus1400,
    tzUTCUnknown
} CTS_Time_Zone_Type_t;

typedef enum

{
    doStandardTime,
    doHalfAnHourDaylightTime,
    doDaylightTime,
    doDoubleDaylightTime,
    doUnknown
} CTS_DST_Offset_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

```
CTS_ERROR_INVALID_INSTANCE_ID
CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
```

Notes:

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

CTS_Query_Local_Time_Information

This function is responsible for querying the Local Time on the specified CTS instance.

Prototype:

int BTPSAPI **CTS_Query_Local_Time_Information**(unsigned int BluetoothStackID, unsigned int InstanceID, CTS_Local_Time_Information_Data_t *Local_Time);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This InstanceID was returned from the CTS_Initialize_Service().
LocalTime	A pointer to return the current Local Time for the specified CTS Instance.

Return:

Zero if successful.

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

CTS_ERROR_INVALID_INSTANCE_ID
CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Notes:

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

CTS_Reference_Time_Information_Read_Request_Response

The following function is responsible for responding to a CTS Read reference Time Information Request.

Prototype:

int BTPSAPI **CTS_Reference_Time_Information_Read_Request_Response**(unsigned int BluetoothStackID, unsigned int TransactionID, CTS_Reference_Time_Information_Data_t *Reference_Time);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	The Transaction ID of the original read request..
Reference_Time	A pointer to the Reference Time information that is to be sent to the remote device. The Reference Time Information Data Structure is as follows:

```
typedef  struct
{
    CTS_Time_Source_Type_t      Source;
    Byte_t                      Accuracy;
    Byte_t                      Days_Since_Update;
    Byte_t                      Hours_Since_Update;
}CTS_Reference_Time_Information_Data_t;
```

Where the Time Source Type enum is as follows:

```
typedef enum
{
  tsUnknown,
  tsNetworkTimeProtocol,
  tsGps,
  tsRadioTimeSignal,
  tsManual,
  tsAtomicClock,
  tsCellularNetwork
} CTS_Time_Source_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown/XXX

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.

CTS_Reference_Time_Information_Request_Error_Response

The following function is responsible for responding to a CTS Read Reference Time Information Request when an error occured.

Prototype:

int BTPSAPI **CTS_Reference_Time_Information_Read_Request_Error_Response** (unsigned int BluetoothStackID, unsigned int TransactionID, Byte_t ErrorCode) ;

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
TransactionID	The Transaction ID of the original read request.
ErrorCode	ErrorCode is used to determine if the Request is being accepted by the server or if an error response is issued instead. This function returns a zero if successful or a negative return error code if an error occurs

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown/XXX

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.

CTS_Read_Client_Configuration_Response

The following function is responsible for responding to a CTS Read Client Configuration Request.

Prototype:

int BTPSAPI **CTS_Read_Client_Configuration_Response**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int TransactionID, Word_t ClientConfiguration);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This InstanceID was returned from the CTS_Initialize_Service().
TransactionID	The Transaction ID of the original read request. This value was received in the etCTS_Read_Client_Configuration_Request event.
ClientConfiguration	The Client Configuration to send to the remote device.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_INSTANCE_ID
CTS_ERROR_INVALID_PARAMETER
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.

CTS_Notify_Current_Time

The following function is responsible for sending a Current Time Notification to a specified remote device.

Prototype:

int BTPSAPI **CTS_Notify_Current_Time**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, CTS_Current_Time_Data_t *Current_Time) ;

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This InstanceID was returned from the CTS_Initialize_Service().
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value notification.

Current_Time

The Curernt Time Data structure contains all of the required and optional data for the notification. This structure is declared as follows:

```
typedef struct
{
    CTS_Exact_Time_Data_t       Exact_Time;
    Byte_t                      Adjust_Reason_Mask;
} CTS_Current_Time_Data_t;
```

Where the Exact Time Data Structure, Day Date Time Data Structure, Date Time Data Structure and Week Day Type Enum are defined as follows:

```
typedef struct
{
    CTS_Day_Date_Time_Data_t    Day_Date_Time;
    Byte_t                      Fractions256;
} CTS_Exact_Time_Data_t;

typedef  struct
{
    CTS_Date_Time_Data_t        Date_Time;
    CTS_Week_Day_Type_t         Day_Of_Week;
} CTS_Day_Date_Time_Data_t;

typedef struct
{
    Word_t                      Year;
    CTS_Month_Of_Year_Type_t    Month;
    Byte_t                      Day;
    Byte_t                      Hours;
    Byte_t                      Minutes;
    Byte_t                      Seconds;
} CTS_Date_Time_Data_t;

typedef enum
{
    wdUnknown,
    wdMonday,
    wdTuesday,
    wdWednesday,
    wdThursday,
    wdFriday,
    wdSaturday,
    wdSunday
} CTS_Week_Day_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_INVALID_INSTANCE_ID

CTS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

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.

CTS_Decode_Current_Time

The following function is responsible for parsing a value received from a remote CTS Server interpreting it as a Current Time charactistic.

Prototype:

int BTPSAPI **CTS_Decode_Current_Time**(unsigned int ValueLength, Byte_t *Value, CTS_Current_Time_Data_t *Current_Time);

Parameters:

ValueLength	Specifies the length of the Current Time value returned by the remote CTS Server.
Value	Value is a pointer to the Current Time data returned by the remote CTS Server.
Current_Time	A pointer to store the parsed Current Time value. It should be a non NULL pointing to valid memory.

```
typedef struct
{
    CTS_Exact_Time_Data_t       Exact_Time;
    Byte_t                      Adjust_Reason_Mask;
} CTS_Current_Time_Data_t;
```

Where the Exact Time Data Structure, Day Date Time Data Structure, Date Time Data Structure and Week Day Type Enum are defined as follows:

```
typedef struct
{
    CTS_Day_Date_Time_Data_t    Day_Date_Time;
    Byte_t                      Fractions256;
} CTS_Exact_Time_Data_t;

typedef  struct
{
    CTS_Date_Time_Data_t        Date_Time;
```

```
    CTS_Week_Day_Type_t         Day_Of_Week;
} CTS_Day_Date_Time_Data_t;

typedef struct
{
    Word_t                      Year;
    CTS_Month_Of_Year_Type_t    Month;
    Byte_t                      Day;
    Byte_t                      Hours;
    Byte_t                      Minutes;
    Byte_t                      Seconds;
} CTS_Date_Time_Data_t;

typedef enum
{
    wdUnknown,
    wdMonday,
    wdTuesday,
    wdWednesday,
    wdThursday,
    wdFriday,
    wdSaturday,
    wdSunday
} CTS_Week_Day_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_MALFORMATTED_DATA
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown\XXX

CTS_Decode_Local_Time_Information

The following function is responsible for parsing a value received from a remote CTS Server interpreting it as a Local Time Information charactistic.

Prototype:

int BTPSAPI **CTS_Decode_Local_Time_Information**(unsigned int ValueLength, Byte_t *Value, CTS_Local_Time_Information_Data_t *Local_Time);

Parameters:

ValueLength	Specifies the length of the Local Time Information value returned by the remote CTS Server.

Value	Value is a pointer to the Local Time Information data returned by the remote CTS Server.
Local_Time	A pointer to store the parsed Local Time Information value. It should be a non NULL pointing to valid memory.

```
typedef  struct
{
    CTS_Time_Zone_Type_t    Time_Zone;
    CTS_DST_Offset_Type_t   Daylight_Saving_Time;
} CTS_Local_Time_Information_Data_t;
```

With the Time Zone Type enum and DST Offset enum defined as follows:

```
typedef enum
{
    tzUTCMinus1200,
    tzUTCMinus1100,
    tzUTCMinus1000,
    tzUTCMinus930,
    tzUTCMinus900,
    tzUTCMinus800,
    tzUTCMinus700,
    tzUTCMinus600,
    tzUTCMinus500,
    tzUTCMinus430,
    tzUTCMinus400,
    tzUTCMinus330,
    tzUTCMinus300,
    tzUTCMinus200,
    tzUTCMinus100,
    tzUTCPlus000,
    tzUTCPlus100,
    tzUTCPlus200,
    tzUTCPlus300,
    tzUTCPlus330,
    tzUTCPlus400,
    tzUTCPlus430,
    tzUTCPlus500,
    tzUTCPlus530,
    tzUTCPlus545,
    tzUTCPlus600,
    tzUTCPlus630,
    tzUTCPlus700,
    tzUTCPlus800,
    tzUTCPlus845,
    tzUTCPlus900,
    tzUTCPlus930,
    tzUTCPlus1000,
    tzUTCPlus1030,
    tzUTCPlus1100,
```

```
    tzUTCPlus1130,
    tzUTCPlus1200,
    tzUTCPlus1245,
    tzUTCPlus1300,
    tzUTCPlus1400,
    tzUTCUnknown
} CTS_Time_Zone_Type_t;

typedef enum

{
    doStandardTime,
    doHalfAnHourDaylightTime,
    doDaylightTime,
    doDoubleDaylightTime,
    doUnknown
} CTS_DST_Offset_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

CTS_ERROR_MALFORMATTED_DATA
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown\XXX

CTS_Decode_Reference_Time_Information

The following function is responsible for parsing a value received from a remote CTS Server interpreting it as a Reference Time Information charactistic.

Prototype:

int BTPSAPI **CTS_Decode_Reference_Time_Information**(unsigned int ValueLength, Byte_t *Value, CTS_Reference_Time_Information_Data_t *Reference_Time);

Parameters:

ValueLength	Specifies the length of the Reference Time Information value returned by the remote CTS Server.
Value	Value is a pointer to the Reference Time Information data returned by the remote CTS Server.
Reference_Time	A pointer to store the parsed Reference Time Information value. It should be a non NULL pointing to valid memory.

```
typedef  struct
{
    CTS_Time_Source_Type_t      Source;
    Byte_t                      Accuracy;
```

```
    Byte_t                  Days_Since_Update;
    Byte_t                  Hours_Since_Update;
} CTS_Reference_Time_Information_Data_t;
```

Where the Time Source Type enum is as follows:

```
typedef enum
{
  tsUnknown,
  tsNetworkTimeProtocol,
  tsGps,
  tsRadioTimeSignal,
  tsManual,
  tsAtomicClock,
  tsCellularNetwork
} CTS_Time_Source_Type_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:
CTS_ERROR_MALFORMATTED_DATA
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown\XXX

2.2 Current Time Service Event Callback Prototypes

2.2.1 Server Event Callback

The event callback function mentioned in the CTS_Initialize_Service command accepts the callback function described by the following prototype.

CTS_Event_Callback_t

This The event callback function mentioned in the CTS_Initialize_Service command accepts the callback function described by the following prototype.

Note:

This function MUST NOT Block and wait for events that can only be satisfied by Receiving CTS Service Event Packets. A Deadlock WILL occur because NO CTS Event Callbacks will be issued while this function is currently outstanding.

Prototype:

```
typedef void (BTPSAPI *CTS_Event_Callback_t)(unsigned int BluetoothStackID,
    CTS_Event_Data_t *CTS_Event_Data, unsigned long CallbackParameter);
```

Parameters:

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

```
typedef struct
{
    CTS_Event_Type_t      Event_Data_Type;
    Word_t                Event_Data_Size;
    union
    {
        CTS_Read_Client_Configuration_Data_t
          *CTS_Read_Client_Configuration_Data;
        CTS_Client_Configuration_Update_Data_t
          *CTS_Client_Configuration_Update_Data;
        CTS_Read_Current_Time_Request_Data_t
          *CTS_ Read_Current_Time_Request _Data;
        CTS_ Read_Reference_Time_Information_Request _Data_t
        *CTS_ Read_Reference_Time_Information_Request _Data;
    } Event_Data;
} CTS_Event_Data_t;
```

Where, Event_Data_Type is one of the enumerations of the event types listed in the table in section 2.3, and each data structure in the union is described with its event in that section as well.

CallbackParameter	User-defined parameter that was defined in the callback registration.

Return:

XXX/None

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 Current Time Service Events

The Current Time Service contains events that are received by the Server. The following sections detail those events.

2.3.1 Current Time Service Server Events

The possible Current Time Service Server Events from the Bluetooth stack are listed in the table below and are described in the text which follows:

Server Commands	
Function	**Description**
etCTS_Read_Client_Configuration_Request	Dispatched to a CTS Server when a CTS Client is attempting to read a descriptor.
etCTS_Client_Configuration_Update	Dispatched to a CTS Server when a CTS Client has written a Client Configuration descriptor.
etCTS_Read_Current_Time_Request	Dispatched to a CTS Server when a CTS client sends a request to read current time data.
etCTS_ Read_Reference_Time_Information_ Request	Dispatched to a CTS Server when a CTS Client sends request to read Reference Time Information data.

etCTS_Read_Client_Configuartion_Request

Dispatched to a CTS Server when a CTS Client is attempting to read a descriptor.

Return Structure:

```
typedef struct
{
    unsigned int                InstanceID;
    unsigned int                ConnectionID;
    unsigned int                TransactionID;
    GATT_Connection_Type_t      ConnectionType;
    BD_ADDR_t                   RemoteDevice;
    CTS_Characteristic_Type_t   ClientConfigurationType;
} CTS_Read_Client_Configuration_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Connection ID of the currently connected remote CTS server device.
TransactionID	The TransactionID identifies the transaction between a client and server. This identifier should be used to respond to the current request.
ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.

ClientConfigurationType	Specifies the valid Read Request types that a server may receive in an etCTS_Server_Read_Client_Configuration_Request or etCTS_Server_Client_Configuration_Update event. This is also used by the CTS_Send_Notification to denote the characteristic value to notify.

etCTS_Client_Configuration_Update

Dispatched to a CTS Server when a CTS Client has written a Client Configuration descriptor.

Return Structure:

```
typedef struct
{
    unsigned int                InstanceID;
    unsigned int                ConnectionID;
    GATT_Connection_Type_t      ConnectionType;
    BD_ADDR_t                   RemoteDevice;
    CTS_Characteristic_Type_t   ClientConfigurationType;
    Word_t                      ClientConfiguration;
} CTS_Client_Configuration_Update_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Connection ID of the currently connected remote CTS server device.
ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.
ClientConfigurationType	Specifies the valid Read Request types that a server may receive in an etCTS_Server_Read_Client_Configuration_Request or etCTS_Server_Client_Configuration_Update event. This is also used by the CTS_Send_Notification to denote the characteristic value to notify.
ClientConfiguration	The New Client Configuration for the specified characteristic.

etCTS_ Read_Current_Time_Request

Dispatched to a CTS Server when a CTS client sends a request to read current time data.

Return Structure:

```
typedef struct
```

```
{
    unsigned int                InstanceID;
    unsigned int                ConnectionID;
    unsigned int                TransactionID;
    GATT_Connection_Type_t      ConnectionType;
    BD_ADDR_t                   RemoteDevice;
} CTS_Read_Current_Time_Request_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.
ConnectionID	Connection ID of the currently connected remote CTS server device.
TransactionID	The TransactionID identifies the transaction between a client and server. This identifier should be used to respond to the current request.
ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.

etCTS_ Read_Reference_Time_Information_Request

Dispatched to a CTS Server when a CTS Client sends request to read Reference Time Information data.

Return Structure:

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

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected..
ConnectionID	Connection ID of the currently connected remote CTS server device.
TransactionID	The TransactionID identifies the transaction between a client and server. This identifier should be used to respond to the current request.

ConnectionType	Identifies the type of remote Bluetooth device that is connected. Currently this value will be gctLE only.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.

3. File Distributions

The header files that are distributed with the Bluetooth Current Time Service Library are listed in the table below

File	Contents/Description
CTSAPI.h	Bluetooth Current Time Service (GATT based) API Type Definitions, Constants, and Prototypes.
CTSTypes.h	Bluetooth Current Time Service Types.
SS1BTCTS.h	Bluetooth Current Time Service Include file