Reference Time Update Service (RTUS)

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 3
1.1 Scope 3
1.2 Applicable Documents 4
1.3 Acronyms and Abbreviations 4
2. REFERENCE TIME UPDATE SERVICE PROGRAMMING INTERFACES 5
2.1 Reference Time Update Service Commands 5
RTUS_Initialize_Service 5
RTUS_Initialize_Service_Handle_Range 6
RTUS_Cleanup_Service 7
RTUS_Query_Number_Attributes 7
RTUS_Set_Time_Update_State 8
RTUS_Query_Time_Update_State 8
RTUS_Decode_Time_Update_State 9
RTUS_Format_Control_Point_Command 9
2.2 Reference Time Update Service Event Callback Prototypes 10
2.2.1 SERVER EVENT CALLBACK 10
RTUS_Event_Callback_t 10
2.3 Reference Time Update Service Events 11
2.3.1 REFERENCE TIME UPDATE SERVICE SERVER EVENTS 11
etRTUS_Server_Read_Time_With_Dst_Request 11
3. FILE DISTRIBUTIONS 13

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), RTUS (Reference Time Update 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 Reference Time Update Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Reference Time Update Service 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

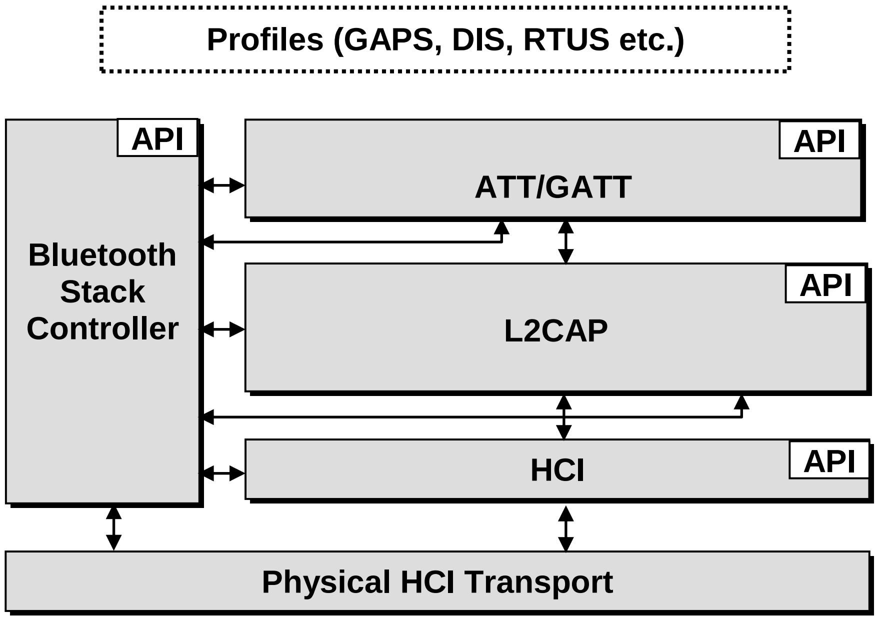

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

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, 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 Doc Reference Time Update Service Specification*, version v10r00, September 15, 2011

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
RTUS	Reference Time Update Service
BD_ADDR	Bluetooth Device Address
BT	Bluetooth
DIS	Device Information Service
GATT	Generic Attribute Protocol
GAPS	Generic Access Profile Service
HCI	Host Controller Interface
HS	High Speed
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy

2. Reference Time Update Service Programming Interfaces

The Reference Time Update Service programming interface defines the protocols and procedures to be used to implement Reference Time Update Service capabilities. The Reference Time Update Service commands are listed in section 2.1, the event callback prototypes are described in section 2.2, and the Reference Time Update Service events are itemized in section 2.3. The actual prototypes and constants outlined in this section can be found in the **RTUSAPI.H** header file in the Bluetopia distribution.

2.1 Reference Time Update Service Commands

The available Reference Time Update Service command functions are listed in the table below and are described in the text that follows.

Function	Description
RTUS_Initialize_Service	Opens a RTUS Server.
RTUS_Initialize_Service_Handle_Range	Opens a RTUS Server with the ability to control the location of the service in the GATT database.
RTUS_Cleanup_Service	Closes an opened RTUS Server.
RTUS_Query_Number_Attributes	Queries the number of attributes.
RTUS_Set_Time_Update_State	Sets the Time Update State on the specified RTUS Instance
RTUS_Query_Time_Update_State	Gets the Time Update State from the specified RTUS Instance
RTUS_Decode_Time_Update_State	Parses a value received from a remote RTUS Server interpreting it as a Time Update State characteristic
RTUS_Format_Control_Point_Command	Formats a Reference Time Update Control Command into a user specified buffer

RTUS_Initialize_Service

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

Prototype:

int BTPSAPI **RTUS_Initialize_Service** (unsigned int BluetoothStackID, RTUS_Event_Callback_t EventCallback, unsigned long CallbackParameter, unsigned int *ServiceID);

Parameters:

BluetoothStackID	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 RTUS service returned from GATT_Register_Service API

Return:

Positive, non-zero if successful. The return value will be the Service Instance ID of RTUS 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.

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

RTUS_ERROR_INSUFFICIENT_RESOURCES
RTUS_ERROR_SERVICE_ALREADY_REGISTERED
RTUS_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

Possible Events:

RTUS_Initialize_Service_Handle_Range

This function is responsible for opening a RTUS Server with the ability to control the location of the service in the GATT database.

Prototype:

int BTPSAPI **RTUS_Initialize_Service_Handle_Range**(unsigned int BluetoothStackID,
RTUS_Event_Callback_t EventCallback, unsigned long CallbackParameter,
unsigned int *ServiceID, GATT_Attribute_Handle_Group_t *ServiceHandleRange);

Parameters:

BluetoothStackID	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 RTUS service returned from GATT_Register_Service API
ServiceHandleRange	Pointer to a Service Handle Range structure, that on input can be used to control the location of the service in the GATT

database, and on output returns the handle range that the service is using in the GATT database.

Return:

Positive, non-zero if successful. The return value will be the Service Instance ID of RTUS 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.

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

RTUS _ERROR_INSUFFICIENT_RESOURCES
RTUS _ERROR_INVALID_PARAMETER
RTUS _ERROR_SERVICE_ALREADY_REGISTERED
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:

RTUS_Cleanup_Service

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

Prototype:

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

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The Service Instance ID to close. This is the value that was returned from the RTUS_Initialize_Service() function.

Return: Zero if successful. An error code if negative; one of the following values:

RTUS_ERROR_INVALID_PARAMETER
RTUS_ERROR_INVALID_INSTANCE_ID

Possible Events:

RTUS_Query_Number_Attributes

This function is responsible for querying the number of attributes that are contained in the RTUS Service that is registered with a call to RTUS_Initialize_Service().

Prototype:

unsigned int BTPSAPI **RTUS_Query_Number_Attributes** (void)

Parameters:

Return: Zero if successful.

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

RTUS _ERROR_INVALID_PARAMETER
RTUS _ERROR_INVALID_INSTANCE_ID

Possible Events:

RTUS_Set_Time_Update_State

This function is responsible for setting the Time Update State of Reference time update on the specified RTUS instance.

Prototype:

int BTPSAPI **RTUS_Set_Time_Update_State**(unsigned int BluetoothStackID, unsigned int InstanceID, RTUS_Time_Update_State_Data_t *TimeUpdateState);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
InstanceID	The InstanceID returned from a successful call to RTUS_Initialize_Server().
Next_Dst_Change_Time	Time Update State to set for the specified RTUS Instance.

Return:

Zero if successful.

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

RTUS_ERROR_INVALID_INSTANCE_ID
RTUS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

RTUS_Query_Time_Update_State

This function is responsible for querying the Time Update State of reference time update from the specified RTUS Instance.

Prototype:

int BTPSAPI **RTUS_Query_Time_Update_State**(unsigned int BluetoothStackID, unsigned int InstanceID, RTUS_Time_Update_State_Data_t *TimeUpdateState);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth Protocol Stack via a call to RTUS _Initialize.

InstanceID	The InstanceID returned from a successful call to RTUS_Initialize_Server().
TimeUpdateState	A Pointer to return the Time Update State for the specified RTUS Instance.

Return:

Zero if successful.

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

RTUS_ERROR_INVALID_INSTANCE_ID
RTUS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

RTUS_Decode_Time_Update_State

The following function is responsible for parsing a value received from a remote RTUS Server interpreting it as a Time Update State characteristic.

Prototype:

int BTPSAPI **RTUS_Decode_Time_Update_State**(unsigned int ValueLength, Byte_t *Value, RTUS_Time_Update_State_Data_t *TimeUpdateState);

Parameters:

ValueLength	Specifies the length of the Time update state value returned by the remote RTUS Server.
Value	Value is a pointer to the time update state data returned by the remote RTUS Server.
TimeUpdateState	A pointer to store the parsed Time Update State value.

Return:

Zero if successful.

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

RTUS_ERROR_MALFORMATTED_DATA

Possible Events:

RTUS_Format_Control_Point_Command

The following function is responsible for formatting a Reference Time Update Control Command into a user specified buffer.

Prototype:

int BTPSAPI **RTUS_Format_Control_Point_Command**
(RTUS_Time_Update_Control_Command_t Command, unsigned int BufferLength, Byte_t *Buffer);

Parameters:

Command	Specifies the command to format.
BufferLength	Specifies the length of the buffer.
Buffer	A pointer to format the Command into.

Return:

Zero if successful.

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

RTUS_ERROR_INVALID_PARAMETER

Possible Events:

2.2 Reference Time Update Service Event Callback Prototypes

2.2.1 Server Event Callback

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

RTUS_Event_Callback_t

Prototype of callback function passed in the RTUS_Initialize_Service command.

Prototype:

typedef void (BTPSAPI ***RTUS_Event_Callback_t**)(unsigned int BluetoothStackID, RTUS_Event_Data_t *RTUS_Event_Data, unsigned long CallbackParameter);

Parameters:

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

```
typedef struct _tagRTUS_Event_Data_t
{
    RTUS_Event_Type_t       Event_Data_Type;
    Word_t                  Event_Data_Size;
    union
    {
            RTUS_Time_Update_Control_Command_Data_t
                                    *RTUS_Time_Update_Control_Command_Data;
      } Event_Data;
   } RTUS_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:

2.3 Reference Time Update Service Events

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

2.3.1 Reference Time Update Service Server Events

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

Event	Description
etRTUS_Server_Time_Update_Control_Point _Command	RTUS Service Event is dispatched to a RTUS Server when a RTUS Client has sent a Time Update Control Point Command.

etRTUS_Server_Read_Time_With_Dst_Request

Dispatched when a RTUS Client requests Time with DST read request to a registered RTUS Server.

Return Structure:

```
typedef struct _tagRTUS_Time_Update_Control_Command_Data_t
{
  unsigned int                          InstanceID;
  unsigned int                          ConnectionID;
  GATT_Connection_Type_t                ConnectionType;
  BD_ADDR_t                             RemoteDevice;
  RTUS_Time_Update_Control_Command_t    Command;
} RTUS_Time_Update_Control_Command_Data_t;
```

Event Parameters:

InstanceID	Identifies the Local Server Instance to which the Remote Client has connected.

ConnectionID	Identifier that uniquely identifies the actual connection of remote device that is making the 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.
Command	Specifies the Time Update Control Point command that the client sent. The following enumerated type represents all of the valid commands that may be received in an etRTUS_Server_Time_Update_Control_Point_Command Server event OR that may be written to a remote RTUS Server.

```
typedef enum
{
   cpGet_Reference_Update    =
                             RTUS_TIME_UPDATE_CONTROL_POINT_GET_REFERE
                             NCE_UPDATE,
   cpCancel_Reference_Update =
                             RTUS_TIME_UPDATE_CONTROL_POINT_CANCEL_REF
                             ERENCE_UPDATE
} RTUS_Time_Update_Control_Command_t;
```

3. File Distributions

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

File	Contents/Description
RTUSAPI.h	Bluetooth Reference Time Update Service (GATT based) API Type Definitions, Constants, and Prototypes.
RTUSType.h	Bluetooth Reference Time Update Service Types.
SS1BTRTU.h	Bluetooth Reference Time Update Service Include file