Glucose Service (GLS)

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

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), GLS (Glucose 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 Glucose Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Glucose Service library.

1.1 Scope

This reference manual provides information on the GLS 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

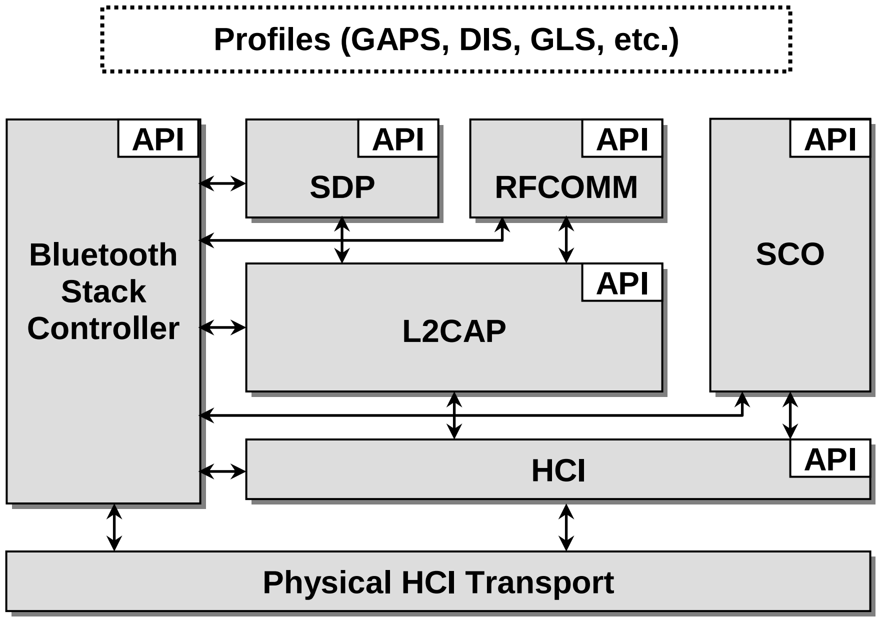

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

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 Glucose 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
GAPS	Generic Access Profile Service
GATT	Generic Attribute Protocol
GLS	Glucose Service
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. GLS Programming Interface

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

2.1 Glucose Service Commands

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

Server Commands	
Function	**Description**
GLS_Initialize_Service	Opens a GLS Server.
GLS_Cleanup_Service	Closes an opened GLS Server.
GLS_Set_Glucose_Feature	Sets the supported Glucose Features on the specified GLS Instace.
GLS_Query_Glucose_Feature	Gets the current Glucose Features from the specified GLS Instance.
GLS_Read_Client_Configuration_Response	Responds to a GLS Read Client Configuration Request.
GLS_Record_Access_Control_Point_Response	Responds to a Record Access Control Point Command received from a remote device.
GLS_Notify_Glucose_Measurement	Sends a Glucose Measurement notification to a specified remote device.
GLS_Notify_Glucose_Measurement_Context	Sends a Glucose Measurement Context notification to a specified remote device.
GLS_Indicate_Number_Of_Stored_Records	Indicates Number of Stored Records to a specified remote device.
GLS_Indicate_Record_Access_Control_Point_ Result	Sends a Record Access Control Point indication to a specified remote device.
GLS_Decode_Glucose_Measurement	Parses a value received from a remote GLS Server interpreting it as a Glucose Measurement characteristic.
GLS_Decode_Glucose_Measurement_Context	Parses a value received from a remote GLS Server interpreting it as a Glucose

	Context characteristic.
GLS_Decode_Record_Access_Control_Point_ Response	Parses a value received from a remote GLS Server interpreting it as a response code of record access control point.
GLS_Format_Record_Access_Control_Point_ Command	Formats a Record Access Control Point Command into a user specified buffer.

GLS_Initialize_Service

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

Notes:

1. Only one GLS 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 **GLS_Initialize_Service**(unsigned int BluetoothStackID,
  GLS_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 GLS service returned from GATT_Register_Service API.

Return:

Positive non-zero if successful. The return value will be the Service ID of GLS 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:

GLS_ERROR_INSUFFICIENT_RESOURCES
GLS_ERROR_SERVICE_ALREADY_REGISTERED
GLS_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.

GLS_Cleanup_Service

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

Prototype:

int BTPSAPI **GLS_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 GLS_Initialize_Service().

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INVALID_PARAMETER
GLS_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.

GLS_Set_Glucose_Feature

This function is responsible for setting the supported Glucose features on the specified GLS Instance.

Prototype:

int BTPSAPI **GLS_Set_Glucose_Feature**(unsigned int BluetoothStackID,
unsigned int InstanceID, Word_t SupportedFeatures);

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 GLS_Initialize_Service().
SupportedFeatures	The supported features are to set for the specified GLS Instance. Possible values include one or more of the following bit-mask values: GLS_FEATURE_LOW_BATTERY_DETECTION_DURING_ MEASUREMENT GLS_FEATURE_SENSOR_MALFUNCTION_DETECTION GLS_FEATURE_SENSOR_SAMPLE_SIZE GLS_FEATURE_SENSOR_STRIP_INSERTION_ERROR_ DETECTION GLS_FEATURE_SENSOR_TYPE_ERROR_DETECTION GLS_FEATURE_SENSOR_RESULT_HIGH_LOW_ DETECTION GLS_FEATURE_SENSOR_TEMPERATURE_HIGH_LOW_ DETECTION GLS_FEATURE_SENSOR_READ_INTERRUPT_ DETECTION GLS_FEATURE_GENERAL_DEVICE_FAULT GLS_FEATURE_TIME_FAULT GLS_FEATURE_MULTIPLE_BOND_SUPPORT

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INVALID_INSTANCE_ID
GLS_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.

GLS_Query_Glucose_Feature

This function is responsible for querying the current Glucose Features from the specified GLS Instance.

Prototype:

int BTPSAPI **GLS_Query_Glucose_Feature**(unsigned int BluetoothStackID,
unsigned int InstanceID, Word_t *SupportedFeatures);

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 GLS_Initialize_Service().
SupportedFeatures	A pointer to return the current Glucose Features for the specified GLS Instance.

Return:

Zero if successful.

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

GLS_ERROR_INVALID_INSTANCE_ID
GLS_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.

GLS_Read_Client_Configuration_Response

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

Prototype:

int BTPSAPI **GLS_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 GLS_Initialize_Service().
TransactionID	The Transaction ID of the original read request. This value was received in the etGLS_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:

GLS_ERROR_INVALID_INSTANCE_ID
GLS_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.

GLS_Record_Access_Control_Point_Response

The following function is responsible for responding to a Record Access Control Point Command received from a remote device.

Note:

1. This function is primarily provided to allow a way to reject Record Access Control Point commands when the Server has not been configured properly for RACP operation, the Client does not have proper authentication to write to the RACP characteristic or a RACP procedure with the Client is already in progress. All other reasons should return ZERO for the ErrorCode and then send RACP Result indication to indicate any other errors. For Example: If the Operand in the Request is not supported by the Server this API should be called with ErrorCode set to ZERO and then the GLS_Indicate_Record_Access_Control_Point_Result() should be called with the ResponseCode set to GLS_RECORD_ACCESS_ RESPONSE_CODE_OPERATOR_NOT_SUPPORTED.

Prototype:

int BTPSAPI **GLS_Record_Access_Control_Point_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. This value was received in the etGLS_Record_Access_Control_ Point_Command event.
ErrorCode	ErrorCode is used to determine if the Request is being accepted by the server or if an error response is issued instead. If the ErrorCode parameter is set to 0x00 the Procedure Request will be accepted. If the ErrorCode is non-zero then an error response will be sent to the remote device. Possible values of non-zero error response are: GLS_ERROR_CODE_PROCEDURE_ALREADY_IN_ PROGRESS, GLS_ERROR_CODE_CHARACTERISTIC_ CONFIGURATION_IMPROPERLY_ CONFIGURED

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Client_Read_Response

etGATT_Client_Error_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.

GLS_Notify_Glucose_Measurement

The following function is responsible for sending a Glucose Measurement Context notification to a specified remote device.

Prototype:

int BTPSAPI **GLS_Notify_Glucose_Measurement_Context**(
unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, GLS_Glucose_Measurement_Context_Data_t *ContextData);

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 GLS_Initialize_Service().
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value notification.
ContextData	The Glucose Context Data structure contains all of the required and optional data for the notification. This structure is declared as follows:

```
typedef struct
{
    Byte_t                   OptionFlags;
    Word_t                   SequenceNumber;
    Byte_t                   ExtendedFlags;
    GLS_Carbohydrate_Data_t  Carbohydrate;
    Byte_t                   Meal;
    Byte_t                   Tester;
    Byte_t                   Health;
    GLS_Exercise_Data_t      ExerciseData;
    GLS_Medication_Data_t    Medication;
    Word_t                   HbA1c;
} GLS_Glucose_Measurement_Context_Data_t;
```

Where Carbohydrate Data Structure, Exercise Data Structure and Medication Data Structure are defined as follows:

```
typedef struct
{
    Byte_t ID;
    Word_t Value;
} GLS_Carbohydrate_Data_t;

typedef struct
{
    Word_t Duration;
    Byte_t Intensity;
} GLS_Exercise_Data_t;

typedef struct
{
    Byte_t ID;
    Word_t Value;
} GLS_Medication_Data_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INVALID_INSTANCE_ID
GLS_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.

GLS_Notify_Glucose_Measurement_Context

The following function is responsible for sending a Glucose Measurement Context notification to a specified remote device.

Prototype:

int BTPSAPI **GLS_Notify_Glucose_Measurement_Context**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, GLS_Glucose_Measurement_Context_Data_t *ContextData);

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 GLS _Initialize_Service().
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value notification.
ContextData	The Glucose Context Data structure contains all of the required and optional data for the notification. This structure is declared as follows:

```
typedef struct
{
    Byte_t                      OptionFlags;
    Word_t                      SequenceNumber;
    Byte_t                      ExtendedFlags;
    GLS_Carbohydrate_Data_t     Carbohydrate;
    Byte_t                      Meal;
    Byte_t                      Tester;
    Byte_t                      Health;
    GLS_Exercise_Data_t         ExerciseData;
    GLS_Medication_Data_t        Medication;
    Word_t                       HbA1c;
} GLS_Glucose_Measurement_Context_Data_t;
```

Where Carbohydrate Data Structure, Exercise Data Structure and Medication Data Structure are defined as follows:

```
typedef struct
{
    Byte_t ID;
    Word_t Value;
} GLS_Carbohydrate_Data_t;

typedef struct
{
    Word_t Duration;
    Byte_t Intensity;
} GLS_Exercise_Data_t;

typedef struct
{
    Byte_t ID;
    Word_t Value;
} GLS_Medication_Data_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

```
GLS_ERROR_INVALID_INSTANCE_ID
GLS_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.

GLS_Indicate_Number_Of_Stored_Records

The following function is responsible for Number of Stored Records indication to a specified remote device. Only 1 Number of Stored Records indication may be outstanding per GLS Instance.

Prototype:

int BTPSAPI **GLS_Indicate_Number_Of_Stored_Records**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, Word_t NumberOfStoredRecords);

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 GLS _Initialize_Service().
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value indication.
NumberOfStoredRecords	Number of stored records to be indicated.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INDICATION_OUTSTANDING
GLS_ERROR_INVALID_INSTANCE_ID
GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Server_Indication

Notes:

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

GLS_Indicate_Record_Access_Control_Point_Result

The following function is responsible for sending a Record Access Control Point indication to a specified remote device. Only 1 RACP Request indication may be outstanding per GLS Instance.

Prototype:

int BTPSAPI **GLS_Indicate_Record_Access_Control_Point_Result**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, GLS_RACP_Command_Type_t CommandType, Byte_t ResponseCode);

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 GLS _Initialize_Service().
ConnectionID	Connection ID of the currently connected remote client device to send the handle/value indication
CommandType	The Requested data to indicate.

```
typedef enum
{
    racReportStoredRecordsRequest,
    racDeleteStoredRecordsRequest,
    racAbortOperationRequest,
    racNumberOfStoredRecordsRequest
} GLS_RACP_Command_Type_t;
```

ResponseCode	Response Code to respond to the remote device. Value of ResponseCode should be between GLS_RECORD_ACCESS_RESPONSE_CODE_SUCCESS to GLS_RECORD_ACCESS_RESPONSE_CODE_FILTER_TYPE_NOT_SUPPORTED

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INDICATION_OUTSTANDING
GLS_ERROR_INVALID_INSTANCE_ID
GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

etGATT_Connection_Server_Indication

Notes:

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

GLS_Decode_Glucose_Measurement

The following function is responsible for parsing a value received from a remote GLS Server interpreting it as a Glucose Measurement characteristic.

Prototype:

```
int BTPSAPI GLS_Decode_Glucose_Measurement(unsigned int ValueLength,
    Byte_t *Value, GLS_Glucose_Measurement_Data_t *MeasurementData);
```

Parameters:

ValueLength	Specifies the length of the Glucose Measurement value returned by the remote GLS Server.
Value	Value is a pointer to the Glucose Measurement data returned by the remote GLS Server.
MeasurementData	A pointer to store the parsed Glucose Measurement value. It should be non NULL pointing to valid memory.

```
typedef struct
{
    Byte_t                      OptionFlags;
    Word_t                      SequenceNumber;
    GLS_Date_Time_Data_t        BaseTime;
    Word_t                      TimeOffset;
    GLS_Concentration_Data_t    GlucoseConcentration;
    Word_t                      SensorStatus;
} GLS_Glucose_Measurement_Data_t;
```

Where the Concentration Data Structure, Date Time Data Structure are defined as follows:

```
typedef struct
{
    Boolean_t   ConcentrationValid;
    Word_t      Value;
    Byte_t      Type;
    Byte_t      SampleLocation;
} GLS_Concentration_Data_t;

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

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_MALFORMATTED_DATA
GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID

BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown\XXX

GLS_Decode_Glucose_Measurement_Context

The following function is responsible for parsing a value received from a remote GLS Server interpreting it as a Glucose Context characteristic.

Prototype:

int BTPSAPI **GLS_Decode_Glucose_Measurement_Context**(unsigned int ValueLength, Byte_t *Value, GLS_Glucose_Measurement_Context_Data_t *ContextData);

Parameters:

ValueLength	Specifies the length of the Glucose Measurement Context value returned by the remote GLS Server.
Value	Value is a pointer to the Glucose Measurement Context data returned by the remote GLS Server.
ContextData	A pointer to store the parsed Glucose Context value. It should be non NULL pointing to valid memory.

```
Typedef struct
{
    Byte_t                    OptionFlags;
    Word_t                    SequenceNumber;
    Byte_t                    ExtendedFlags;
    GLS_Carbohydrate_Data_t   Carbohydrate;
    Byte_t                    Meal;
    Byte_t                    Tester;
    Byte_t                    Health;
    GLS_Exercise_Data_t       ExerciseData;
    GLS_Medication_Data_t     Medication;
    Word_t                    HbA1c;
} GLS_Glucose_Measurement_Context_Data_t;
```

Where Carbohydrate Data Structure, Exercise Data Structure and Medication Data Structure are defined as follows:

```
typedef struct
{
    Byte_t ID;
    Word_t Value;

} GLS_Carbohydrate_Data_t;

typedef struct
{
    Word_t Duration;
    Byte_t Intensity;
} GLS_Exercise_Data_t;
```

```
typedef struct
{
      Byte_t ID;
      Word_t Value;
} GLS_Medication_Data_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

```
GLS_ERROR_MALFORMATTED_DATA
GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER
```

Possible Events:

Unknown\XXX

GLS_Decode_Record_Access_Control_Point_Response

The following function is responsible for parsing a value received from a remote GLS Server interpreting it as a response code of record access control point.

Prototype:

int BTPSAPI **GLS_Decode_Record_Access_Control_Point_Response**(unsigned int ValueLength, Byte_t *Value, GLS_Record_Access_Control_Point_Response_Data_t *RACPData);

Parameters:

ValueLength	Specifies the length of the Record Access Control Point Response value returned by the remote GLS Server.
Value	Value is a pointer to the Record Access Control Point Response data returned by the remote GLS Server.
RACPData	A pointer to store the parsed Record Access Control Point Response data value. It should be non NULL pointing to valid memory.

```
Typedef struct
{
      GLS_RACP_Response_Type_t        ResponseType;
      union
      {
            Word_t       NumberOfStoredRecordsResult;
            GLS_RACP_Response_Code_Vault_t
                  ResponseCodeVault;
      }ResponseData;
} GLS_Record_Access_Control_Point_Response_Data_t;
```

Where RACP Response Code Value Structure is defined as follows:

```
typedef struct
{
    Byte_t RequestOpCode;
    Byte_t ResponseCodeValue;
} GLS_RACP_Response_Code_Value_t;
```

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_MALFORMATTED_DATA
GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown/XXX

GLS_Format_Record_Access_Control_Point_Command

The following function is responsible for formatting a Record Access Control Point Command into a user specified buffer.

Prototype:

int BTPSAPI **GLS_Format_Record_Access_Control_Point_Command**
(GLS_Record_Access_Control_Point_Format_Data_t *FormatData, unsigned int *BufferLength, Byte_t *Buffer);

Parameters:

FormatData The input command to format:

```
typedef struct
{
    GLS_RACP_Command_Type_t      CommandType;
    GLS_RACP_Operator_Type_t     OperatorType;
    GLS_RACP_Filter_Type_t       FilterType;
    (One of the Following values:
        rafSequenceNumber, rafUserFacingTime)
    union
    {
        Word_t                   SequenceNumber;
        GLS_Date_Time_Data_t     UserFacingTime;
        GLS_Sequence_Number_Range_Data_t
            SequenceNumberRange;
        GLS_Date_Time_Range_Data_t
            UserFacingTimeRange;
    }FilterParameters;
}GLS_Record_Access_Control_Point_Format_Data_t;
```

Where the RACP Command Type enum, RACP operator Type enum, Date Time Data Structure, Sequence Number Range Data Structure and Date Time Range Data Structure are defined as follows:

```
typedef enum
{
    racReportStoredRecordsRequest,
    racDeleteStoredRecordsRequest,
    racAbortOperationRequest,
    racNumberOfStoredRecordsRequest
} GLS_RACP_Command_Type_t;

typedef enum
{
    raoNull,
    raoAllRecords,
    raoLessThanOrEqualTo,
    raoGreaterThanOrEqualTo,
    raoWithinRangeOf,
    raoFirstRecord,
    raoLastRecord
}GLS_RACP_Operator_Type_t;

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

typedef struct
{
    Word_t Minimum;
    Word_t Maximum;
} GLS_Sequence_Number_Range_Data_t;

typedef struct
{
    GLS_Date_Time_Data_t Minimum;
    GLS_Date_Time_Data_t Maximum;
} GLS_Date_Time_Range_Data_t;
```

BufferLength	Specifies the Length of the Buffer. **Note**: After formatting the BufferLength, it will contain the actual length of formatted the Record Access Control Point Data.

Buffer	A pointer, pointing to memory of size BufferLength to store the Record Control Access request Data after formatting.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

GLS_ERROR_INSUFFICIENT_BUFFER_SPACE
GLS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_NOT_INITIALIZED
BTGATT_ERROR_INVALID_BLUETOOTH_STACK_ID
BTGATT_ERROR_INVALID_PARAMETER

Possible Events:

Unknown/XXX

2.2 Glucose Service Event Callback Prototypes

2.2.1 Server Event Callback

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

GLS_Event_Callback_t

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

Prototype:

typedef void (BTPSAPI ***GLS_Event_Callback_t**)(unsigned int BluetoothStackID,
    GLS_Event_Data_t *GLS_Event_Data, unsigned long CallbackParameter);

Parameters:

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

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

```
typedef struct
{
    GLS_Event_Type_t      Event_Data_Type;
    Word_t                Event_Data_Size;
    union
    {
        GLS_Read_Client_Configuration_Data_t
            *GLS_Read_Client_Configuration_Data;
        GLS_Client_Configuration_Update_Data_t
            *GLS_Client_Configuration_Update_Data;
```

```
              GLS_Record_Access_Control_Point_Command_Data_t
                 *GLS_Record_Access_Control_Point_Command_Data;
              GLS_Configuartion_Data_t       *GLS_Configuartion_Data;
           } Event_Data;
        } GLS_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 Glucose Service Events

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

2.3.1 Glucose Service Server Events

The possible Glucose 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**
etGLS_Read_Client_Configuration_Request	Dispatched to a GLS Server when a GLS Client is attempting to read the Client Configuration Descriptor.
etGLS_Client_Configuration_Update	Dispatched to a GLS Server when a GLS Client attempts to write to a Client Configuration descriptor.
etGLS_Record_Access_Control_Point_Command	Dispatched to a GLS Server in response to the reception of request from a Client to write to the Record Access Control Point.
etGLS_Confirmation_Data	Dispatched to a GLS Server when a GLS Client has sent a confirmation to a

	previously sent confirmation.

etGLS_Read_Client_Configuartion_Request

Dispatched to a GLS Server when a GLS Client is attempting to read the Client Configuration Descriptor.

Return Structure:

```
typedef struct
{
    unsigned int                InstanceID;
    unsigned int                ConnectionID;
    unsigned int                TransactionID;
    GATT_Connection_Type_t      ConnectionType;
    BD_ADDR_t                   RemoteDevice;
    GLS_Characteristic_Type_t   ClientConfigurationType;
} GLS_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 GLS 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 etGLS_Server_Read_Client_Configuration_Request or etGLS_Server_Client_Configuration_Update event. This is also used by the GLS_Send_Notification to denote the characteristic value to notify.

etGLS_Client_Configuration_Update

Dispatched to a GLS Server when a GLS Client attempts to write to a Client Configuration descriptor.

Return Structure:

```
typedef struct
{
    unsigned int                InstanceID;
    unsigned int                ConnectionID;
    GATT_Connection_Type_t      ConnectionType;
    BD_ADDR_t                   RemoteDevice;
    GLS_Characteristic_Type_t   ClientConfigurationType;
    Word_t                      ClientConfiguration;
} GLS_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 GLS 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 etGLS_Server_Read_Client_Configuration_Request or etGLS_Server_Client_Configuration_Update event. This is also used by the GLS_Send_Notification to denote the characteristic value to notify.
ClientConfiguration	The New Client Configuration for the specified characteristic.

etGLS_Record_Access_Control_Point_Command

Dispatched to a GLS Server in response to the reception of request from a Client to write to the Record Access Control Point.

Return Structure:

```
typedef struct
{
    unsigned int                                   InstanceID;
    unsigned int                                   ConnectionID;
    unsigned int                                   TransactionID;
    GATT_Connection_Type_t                         ConnectionType;
    BD_ADDR_t                                      RemoteDevice;
    GLS_Record_Access_Control_Point_Format_Data_t  FormatData;
} GLS_Record_Access_Control_Point_Command_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 GLS 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.
FormatData	Specifies the format of the Record Access Control Point Command Request Data. This structure is passed as a parameter to GLS_Format_Record_Access_Control_Point_Command API.

```
typedef struct
{
   GLS_RACP_Command_Type_t  CommandType;
   GLS_RACP_Operator_Type_t OperatorType;
   GLS_RACP_Filter_Type_t   FilterType;
   union
   {
      Word_t                              SequenceNumber;
      GLS_Date_Time_Data_t                UserFacingTime;
      GLS_Sequence_Number_Range_Data_t
         SequenceNumberRange;
      GLS_Date_Time_Range_Data_t
         UserFacingTimeRange;
   } FilterParameters;
} GLS_Record_Access_Control_Point_Format_Data_t;
```

etGLS_Confirmation_Data

Dispatched to a GLS Server when a GLS Client has sent a confirmation to a previously sent confirmation.

Return Structure:

```
typedef struct
{
    unsigned int              InstanceID;
    unsigned int              ConnectionID;
    Byte_t                    Status;
    GATT_Connection_Type_t    ConnectionType;
    BD_ADDR_t                 RemoteDevice;
} GLS_Confirmation_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 GLS server device.
Status	The Status member is a bit-mask. This value must be one (or more) of the following bit-mask constant flags GATT_CONFIRMATION_STATUS_SUCCESS GATT_CONFIRMATION_STATUS_TIMEOUT.
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 Glucose Service Library are listed in the table below

File	Contents/Description
GLSAPI.h	Bluetooth Glucose Service (GATT based) API Type Definitions, Constants, and Prototypes.
GLSTYPES.h	Bluetooth Glucose Service Types.
SS1BTGLS.h	Bluetooth Glucose Service Include file