Scan Parameters Service (SCPS)

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. SCPS PROGRAMMING INTERFACE 5

2.1 Scan Parameters Service Commands 5

SCPS_Initialize_Service 5
SCPS_Cleanup_Service 6
SCPS_Read_Client_Configuration_Response 7
SCPS_Notify_Scan_Refresh 8
SCPS_Format_Scan_Interval_Window 8

2.2 Scan Parameter Service Event Callback Prototypes 9

2.2.1 SERVER EVENT CALLBACK 9
SCPS_Event_Callback_t 9

2.3 Scan Parameters Service Events 10

2.3.1 SCAN PARAMETERS SERVICE SERVER EVENTS 10
etSCPS_Read_Client_Configuartion_Request 11
etSCPS_Client_Configuration_Update 12
etSCPS_Server_Write_Scan_Interval_Window_Request 12

3. FILE DISTRIBUTIONS 14

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 Scan Parameters Service Profile Stack provided by Bluetopia®+LE Single. And, Chapter 3 contains the header file name list for the Scan Parameters Service library.

1.1 Scope

This reference manual provides information on the SPCS 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

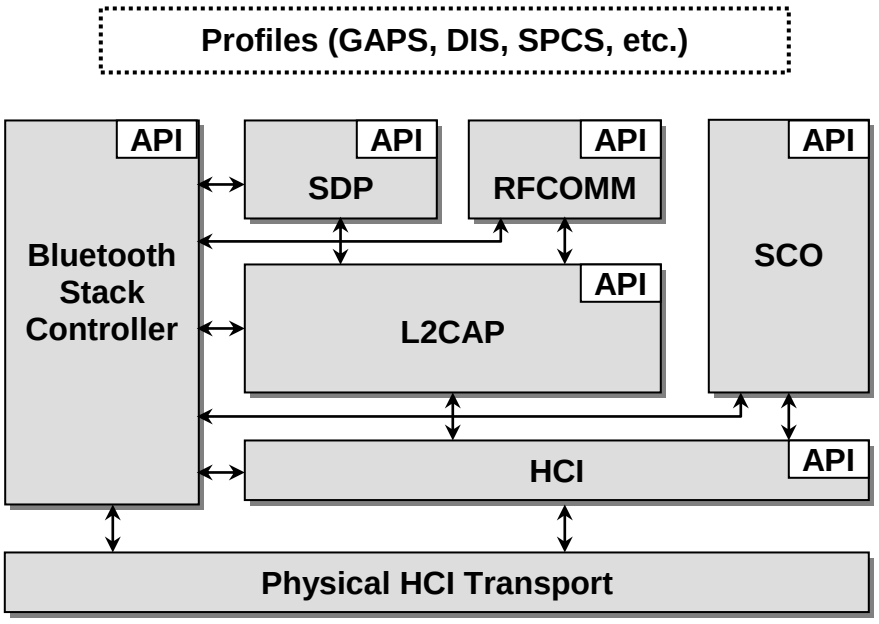

Figure 1-1 The Stonestreet One Bluetooth Protocol Stack

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 Scan Parameters 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
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
SCPS	Scan Parameters Service

2. SCPS Programming Interface

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

2.1 Scan Parameters Service Commands

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

Server Commands	
Function	**Description**
SCPS_Initialize_Service	Opens a SCPS Server.
SCPS _Cleanup_Service	Closes an opened SCPS Server.
SCPS_Read_Client_Configuration_Response	Responds to a SCPS Read Client Configuration Request.
SCPS_Notify_Scan_Refresh	Sends a Scan Refresh notification to the specified remote device.
SCPS_Format_Scan_Interval_Window	Formats a Scan Interval Window Command into the specified buffer.

SCPS_Initialize_Service

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

Notes:

1. Only one SCPS 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 **SCPS_Initialize_Service**(unsigned int BluetoothStackID,
SCPS_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 SCPS service returned from GATT_Register_Service API.

Return:

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

SCPS_ERROR_INSUFFICIENT_RESOURCES
SCPS_ERROR_SERVICE_ALREADY_REGISTERED
SCPS_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.

SCPS_Cleanup_Service

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

Prototype:

int BTPSAPI **SCPS_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 SCPS_Initialize_Service().

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

SCPS_ERROR_INVALID_PARAMETER
SCPS_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.

SCPS_Read_Client_Configuration_Response

This function is responsible for responding to a SCPS Read Client Configuration Request.

Prototype:

int BTPSAPI **SCPS_Read_Client_Configuration_Response**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int TransactionID, Word_t Client_Configuration);

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 SCPS_Initialize_Service().
TransactionID	The Transaction ID of the request.
Client_Configuration	Contains the Client Configuration to send to the remote device.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:

SCPS_ERROR_INVALID_INSTANCE_ID
SCPS_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.

SCPS_Notify_Scan_Refresh

This function is responsible for sending a Scan Refresh notification to a specified remote device.

Prototype:

int BTPSAPI **SCPS_Notify_Scan_Refresh**(unsigned int BluetoothStackID, unsigned int InstanceID, unsigned int ConnectionID, Byte_t ScanRefreshValue);

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 SCPS_Initialize_Service().
ConnectionID	The ConnectionID of the remote device to send the notification to.
ScanRefreshValue	The value for Scan Refresh. If Scan Refresh value is 0, this indicates Server requires refresh.

Return:

Zero if successful.

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

SCPS_ERROR_INVALID_INSTANCE_ID
SCPS_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.

SCPS_Format_Scan_Interval_Window

The following function is responsible for formatting a Scan Interval Window Command into a user specified buffer.

Note:

The BufferLength and Buffer parameter must point to a buffer of at least SCPS_SCAN_INTERVAL_WINDOW_SIZE in size.

Prototype:

int BTPSAPI
SCPS_Format_Scan_Interval_Window(SCPS_Scan_Interval_Window_Data_t *Scan_Interval_Window, unsigned int BufferLength, Byte_t *Buffer);

Parameters:

Scan_Interval_Window	The SCPS_Scan_Interval_Window_Data_t to format. The Scan Interval Window Data structure is declared as follows: typedef struct { Word_t LE_Scan_Interval; Word_t LE_Scan_Window; } **SCPS_Scan_Interval_Window_Data_t;**
BufferLength	The length of the buffer.
Buffer	The buffer that the Scan Interval Window Command is being formatted into.

Return:

Zero if successful.

Negative if an error occurred. Possible values are:
SCPS_ERROR_INVALID_PARAMETER
BTGATT_ERROR_INVALID_PARAMETER

2.2 Scan Parameter Service Event Callback Prototypes

2.2.1 Server Event Callback

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

SCPS_Event_Callback_t

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

Notes:

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

Prototype:

typedef void (BTPSAPI ***SCPS_Event_Callback_t**)(unsigned int BluetoothStackID, SCPS_Event_Data_t *SCPS_Event_Data, unsigned long CallbackParameter);

Parameters:

BluetoothStackID[1]	Unique identifier assigned to this Bluetooth Protocol Stack via a call to BSC_Initialize.
SCPS_Event_Data_t	Data describing the event for which the callback function is called. The Event Data structure is defined as follows:

```
typedef struct
{
    SCPS_Event_Type_t    Event_Data_Type;
    Word_t               Event_Data_Size;
    union
    {
        SCPS_Read_Client_Configuration_Data_t
            *SCPS_Read_Client_Configuration_Data;
        SCPS_Client_Configuration_Update_Data_t
            *SCPS_Client_Configuration_Update_Data;
        SCPS_Write_Scan_Interval_Window_Data_t
            *SCPS_Write_Scan_Interval_Window_Data;
    } Event_Data;
} SCPS_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 Scan Parameters Service Events

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

2.3.1 Scan Parameters Service Server Events

The possible Scan Parameters 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**
etSCPS_Server_Read_Client_Configuration_Request	Dispatched to a SCPS Server when a SCPS Client is attempting to read a descriptor.
etSCPS_Server_Update_Client_Configuration_Request	Dispatched to SCPS Server when a SCPS Client has written a Client Configuration descriptor.
etSCPS_Server_Write_Scan_Interval_Window_Request	Dispatched to a SCPS Server in response to the reception of request from a Client to write to the Scan Interval Window characteristics

etSCPS_Read_Client_Configuartion_Request

Dispatched to a SCPS Server when a SCPS 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;
    SCPS_Characteristic_Type_t  ClientConfigurationType;
} SCPS_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 SCPS 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.
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 etSCPS_Server_Read_Client_Configuration_Request event.

etSCPS_Client_Configuration_Update

Dispatched to a SCPS Server when a SCPS 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;
    SCPS_Characteristic_Type_t  ClientConfigurationType;
    Word_t                      ClientConfiguration;
} SCPS_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 SCPS server device.
ConnectionType	Identifies the type of remote Bluetooth device that is connected.
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 etSCPS_Server_Read_Client_Configuration_Request or etSCPS_Server_Client_Configuration_Update event.
ClientConfiguration	The New Client Configuration for the specified characteristic.

etSCPS_Server_Write_Scan_Interval_Window_Request

Dispatched to a SCPS Server in response to the reception of request from a Client to write to the Scan Interval Window characteristics.

Return Structure:

```
typedef struct
{
    unsigned int                InstanceID;
    unsigned int                ConnectionID;
    GATT_Connection_Type_t      ConnectionType;
```

```
    BD_ADDR_t                       RemoteDevice;
    SCPS_Scan_Interval_Window_Data_t ScanIntervalWindowData;
} SCPS_Write_Scan_Interval_Window_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 SCPS server device.
ConnectionType	Identifies the type of remote Bluetooth device that is connected.
RemoteDevice	Specifies the address of the Client Bluetooth device that has connected to the specified Server.
ScanIntervalWindowData	Specifies the Scan Interval Window Data. The Scan Interval Window Data structure is defined as follows: typedef struct { Word_t LE_Scan_Interval; Word_t LE_Scan_Window; } **SCPS_Scan_Interval_Window_Data_t;**

3. File Distributions

The header files that are distributed with the Bluetooth Scan Parameters Service Library are listed in the table below

File	Contents/Description
SCPSAPI.h	Bluetooth Scan Parameters Service (GATT based) API Type Definitions, Constants, and Prototypes.
SCPSTYPES.h	Bluetooth Scan Parameters Service Types.
SS1BTSCPS.h	Bluetooth Scan Parameters Service Include file