Service/Device Discovery Sub-system (DISC)

Application Programming Interface Reference Manual

Profile Version: 1.0

Release: 2.1.3
May 30, 2011

Louisville, KY www.stonestreetone.com

Table of Contents

1. Introduction

Bluetopia®, the Bluetooth Protocol Stack by Stonestreet One provides a software architecture that encapsulates the upper functionality of the Bluetooth Protocol Stack. More specifically, this stack is a software solution that resides above the Physical HCI (Host Controller Interface) Transport Layer and extends through the L2CAP (Logical Link Control and Adaptation Protocol) and the SCO (Synchronous Connection-Oriented) Link layers. In addition to basic functionality at these layers, the Bluetooth Protocol Stack by Stonestreet One provides implementations of the Service Discovery Protocol (SDP), RFCOMM (the Radio Frequency serial COMMunications port emulator), and several of the Bluetooth Profiles. Program access to these layers, services, and profiles is handled via Application Programming Interface (API) calls.

This document focuses on the API reference that contains a description of all programming interfaces for the Bluetooth service/device discovery sub-system library provided by Bluetopia. Chapter 2 contains a description of the programming interfaces for this module. And, Chapter 3 contains the header file name list for the Bluetooth service/device discovery sub-system.

1.1 Scope

This reference manual provides information on the service/device discovery library 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

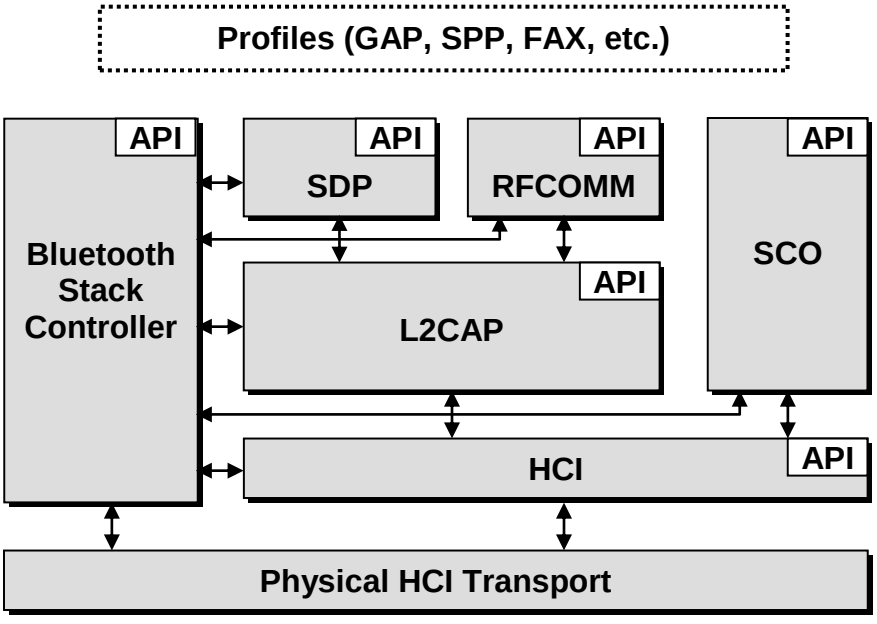

Figure 1-1 Stonestreet One Bluetooth Protocol Stack

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 2, Core System Package*, version 4.0 + BR/EDR, June 30, 2010.
2. *Specification of the Bluetooth System, Volume 3, Core System Package*, version 4.0 + BR/EDR, June 30, 2010.
3. *Bluetooth Assigned Numbers*, version 1.1, February 22, 2001.
4. *Bluetopia®Protocol Stack, Application Programming Interface Reference Manual,* version 2.1.3, August 31, 2010.

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
BD_ADDR	Bluetooth Device Address
BT	Bluetooth
DISC	Discovery
LSB	Least Significant Bit
LE	Low Energy
MSB	Most Significant Bit
SDP	Service Discovery Protocol
SPP	Serial Port Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2. Service/Device Discovery Sub-system Programming Interface

The service/device discovery programming interface defines the procedures to be used to implement the discovery capabilities. The discovery commands are listed in section 2.1, the event callback prototype is described in section 2.2, and the discovery events are itemized in section 2.3. The actual prototypes and constants outlined in this section can be found in the **DISCAPI.H** header file in the Bluetopia distribution.

2.1 Service/Device Discovery Sub-system Commands

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

Function	Description
DISC_Initialize	This function is responsible for initializing a service/device discovery sub-system.
DISC_Cleanup	This function is responsible for cleaning up a previously initialized service/device discovery sub-system instance.
DISC_Device_Discovery_Start	This function is responsible for beginning a device discovery process.
DISC_Device_Discovery_Stop	This function is responsible for ending a currently on-going device discovery process.
DISC_Service_Discovery_Start	This function is responsible for beginning a service discovery process.
DISC_Service_Discovery_Stop	This function is responsible for ending a currently on-going service discovery process.

DISC_Initialize

This function is responsible for initializing the service/device discovery sub-system for the specified Bluetooth protocol stack.

Prototype:

int BTPSAPI **DISC_Initialize**(unsigned int BluetoothStackID);

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth protocol stack instance via a call to BSC_Initialize().

Return:

Zero if successful. On error, a negative value is returned.

DISC_Cleanup

This function is responsible for cleaning up a previously initialized service/device discovery sub-system instance. After calling this function, no other discovery functions may be called for the specified Bluetooth protocol stack unless DISC_Initialize() is called again to re-initialize the sub-system.

Prototype:

void BTPSAPI **DISC_Cleanup**(void);

Parameters:

None.

Return:

None.

DISC_Device_Discovery_Start

This function is responsible for initiating a device discovery process.

Prototype:

int BTPSAPI **DISC_Device_Discovery_Start**(unsigned int BluetoothStackID, Device_Filter_t *DeviceFilter, DISC_Event_Callback_t DiscoveryCallback, unsigned long DiscoveryCallbackParameter);

Parameters:

BluetoothStackID — Unique identifier assigned to this Bluetooth protocol stack via a call to BSC_Initialize().

DeviceFilter — Optional filter structure that may be used to filter the devices returned from the Device Discovery procedure. This structure has the following format:

```
typedef struct
{
  Class_of_Device_t    ClassOfDeviceMask;
  LAP_t                LAP;
} Device_Filter_t;
```

Where,

ClassOfDeviceMask specifies the class of device filter mask to match (any bits). Specifying all zeros for mask means match ALL class of devices (i.e. no filter is applied).

LAP specifies the Inquiry Access Code (IAC) Lower Address Part (LAP) that is specified when performing the inquiry. Specifying a LAP of all zeros means to not apply a LAP filter.

DiscoveryCallback — Pointer to the callback function that will receive the device information as it becomes available.

DiscoveryCallbackParameter Value that will be returned to in the callback parameter of the callback function.

Return:

Zero if successful. On error, a negative value is returned.

DISC_Device_Discovery_Stop

This function is responsible for terminating an on-going device discovery process.

Prototype:

int BTPSAPI **DISC_Device_Discovery_Stop**(unsigned int BluetoothStackID);

Parameters:

BluetoothStackID Unique identifier assigned to this Bluetooth protocol stack instance via a call to BSC_Initialize().

Return:

Zero if successful. On error, a negative value is returned.

DISC_Service_Discovery_Start

This function is responsible for initiating a service discovery process or to queue additional service discovery requests.

Prototype:

int BTPSAPI **DISC_Service_Discovery_Start**(unsigned int BluetoothStackID, BD_ADDR_t BD_ADDR, int NumberOfProfiles, Profile_Identifier_t *ProfileIDList, DISC_Event_Callback_t ServiceDiscoveryCallback, unsigned long ServiceDiscoveryCallbackParameter);

Parameters:

BluetoothStackID Unique identifier assigned to this Bluetooth protocol stack instance via a call to BSC_Initialize().

BD_ADDR BD_ADDR of the device that is to be searched.

NumberOfProfiles Number of profile identifiers in the array pointed to by profile ID List.

ProfileIDList Array of profile IDs to search for. Each entry has the following structure:

```
typedef struct
{
  Word_t          ServiceNameLength;
  Byte_t         *ServiceName;
  Word_t          ServiceDescLength;
  Byte_t         *ServiceDesc;
  Word_t          ServiceProviderLength;
  Byte_t         *ServiceProvider;
```

```
   Profile_Identifier_t  ProfileIdentifier;
   union
   {
     SPP_Info_t       SPPInfo;
     HID_Info_t       HIDInfo;
     HDS_Info_t       HDSInfo;
     HFRE_Info_t      HFREInfo;
   } Profile;
} Profile_Info_t;
```

ServiceDiscoveryCallback	Defines the callback function to use when the service discovery is complete.
ServiceDiscoveryCallbackParameter	Defines the callback parameter to use when the service discovery is complete.

Return:

Zero if successful. On error, a negative value is returned.

DISC_Service_Discovery_Stop

This function is responsible for terminating an on-going service discovery operation.

Prototype:

int BTPSAPI **DISC_Service_Discovery_Stop**(unsigned int BluetoothStackID)

Parameters:

BluetoothStackID	Unique identifier assigned to this Bluetooth protocol stack instance via a call to BSC_Initialize().

Return:

Zero if successful. On error, a negative value is returned.

2.2 Service/Device Discovery Sub-system Event Callback Prototypes

The event callback functions mentioned in the Service Discovery Module Open commands all accept the callback function described by the following prototype.

DISC_Event_Callback_t

Prototype of callback function that is is registered with the DISC_Device_Discovery_Start() or DISC_Service_Discovery_Start() functions.

Prototype:

void (BTPSAPI ***DISC_Event_Callback_t**)(unsigned int BluetoothStackID, DISC_Event_Data_t *DISC_Event_Data, unsigned long CallbackParameter);

Parameters:

BluetoothStackID[1] Unique identifier assigned to this Bluetooth protocol stack instance via a call to BSC_Initialize().

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

```
typedef struct
{
  DISC_Event_Type_t  Event_Data_Type;
  Word_t             Event_Data_Size;
  union
  {
     DISC_Device_Information_Indication_Data_t
              *DISC_Device_Information_Indication_Data;
     DISC_Service_Information_Indication_Data_t
              *DISC_Service_Information_Indication_Data;
     DISC_Service_Search_Error_Indication_Data_t
              *DISC_Service_Search_Error_Indication_Data;
  } Event_Data;
} DISC_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 (e.g., tag value) that was provided in the callback registration.

2.3 Service/Device Discovery Sub-system Events

The possible service/device discovery events from the Bluetooth stack are listed in the table below and are described in the text that follows:

Event	Description
etDISC_Device_Information_Indication	Event containing device information from a device discovery process.
etDISC_Service_Information_Indication	Event containing service information from a service discovery process.
etDISC_Service_Search_Error_Indication	Event indicating an error occurred during a service discovery process.

etDISC_Device_Information_Indication

This event is dispatched to indicate that the discovery sub-system has located a new device and contains the discovered information about the new device.

Return Structure:

```
typedef struct
{
  Device_Info_t DeviceInfo;
} DISC_Device_Information_Indication_Data_t;
```

Event Parameters:

DeviceInfo — Structure that contains the discovered device information. This structure has the following format:

```
typedef struct
{
  BD_ADDR_t          BD_ADDR;
  Class_of_Device_t  ClassOfDevice;
  Word_t             ClockOffset;
  Byte_t             Page_Scan_Repetition_Mode;
  Boolean_t          NameValid;
  char              *DeviceName;
} Device_Info_t;
```

etDISC_Service_Information_Indication

This event is dispatched to indicate that the discovery sub-system has located services on a remote device. Note that the service search had to be explicitly queued via a call to the DISC_Service_Discovery_Start() function.

Return Structure:

```
typedef struct
{
   Service_Info_t        ServiceInfo;
   SDP_Response_Data_t  *Raw_SDP_Response_Data;
} DISC_Service_Information_Indication_Data_t;
```

Event Parameters:

ServiceInfo — Contains the discovered service information. This structure is defined as follows:

```
typedef struct
{
  BD_ADDR_t       BD_ADDR;
  int             NumberOfProfiles;
  Profile_Info_t *ProfileInfo;
} Service_Info_t;
```

Raw_SDP_Response_Data — Pointer to the raw SDP information that was searched to provide the included service information.

etDISC_Service_Search_Error_Indication

This event is dispatched when the discovery sub-system encountered an error with an SDP service search request operation. Note that the service search had to be explicitly queued via a call to the DISC_Service_Discovery_Start() function.

Return Structure:

```
typedef struct
{
  BD_ADDR_t                 BD_ADDR;
  DISC_SDP_Error_Type_t     Error_Type;
  SDP_Error_Response_Data_t *SDP_Error_Response_Data;
} DISC_Service_Search_Error_Indication_Data_t;
```

Event Parameters:

BD_ADDR	Contains the BD_ADDR of the device that was being processed when the error occurred.
Error_Type	The type of error that has occurred during the service discovery. This value will be one of the following types: etRequestFailure etRequestTimeout etConnectionError etErrorResponse etMemoryAllocationFailure etUnknownError
SDP_Error_Response_Data	Contains information returned from SDP when an etErrorResponse is received from SDP.

3. File Distributions

The header files that are distributed with the Bluetooth Service/Device discovery sub-system are listed in the table below.

File	Contents/Description
DISCAPI.h	Bluetooth Service/Device discovery API definitions
SS1BTDIS.h	Bluetooth Service/Device discovery include file