Service Discovery Protocol

Ting-Chao Hou
Outline

1. Introduction
2. Overview
3. Protocol Description
4. Service Attribute Definitions
General Description

- The service discovery protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services.
- In traditional networks, these services might be configured and managed by a network administrator.  
  - e.g., e-mail service, printer service.
- Administered network services of this sort are not suitable for temporary mobile networks (ad hoc networks) formed by using Bluetooth.
Motivation

- Service Discovery in the Bluetooth environment, where the set of services that are available changes dynamically based on the RF proximity of devices in motion, is qualitatively different from service discovery in traditional network based environments.

- The set of SDP servers that are available to an SDP client can change dynamically based on the RF proximity of the servers to the client.
SDP Requirements

- Provide the ability for clients to **Search** for needed services based on **specific attributes** of those services.
- Permit services to be discovered based on the **class of service**.
- Enable **Browsing** of services without a priori knowledge of the specific characteristic of those services.
- Allow a client on one device to discover a service on another device **without** consulting a third device.
SDP Requirements (cont’d)

• Suitable for use on devices of limited complexity.
• Provide the means for the discovery of new services that become available when devices enter RF proximity with a client device as well as when a new service is made available on a device that is in RF proximity with the client device.
• Provides a mechanism to incrementally discover information about the services provided by a device.
SDP Requirements (cont’d)

• Support the caching of service discovery information by intermediary agents to improve the speed or efficiency of the discovery process.
• SDP should be transport independent.
• Function while using L2CAP as its transport protocol.
• Permit the discovery and use of services that provide access to other service discovery protocol.
• Support the creation and definition of new services without requiring registration with a central authority.
SDP Non-Requirements (SDP 1.0)

- Access to services. It only provides access to information about services. Services can be accessed using other protocols.
- Brokering of services.
- Negotiation of service parameters.
- Billing of service use.
- Means for a client to control or change the operation of a service.
- A mechanism for notifying clients when service records are added to or removed from an SDP server.
Bit and Byte Ordering Conventions

- When multiple bit fields are contained in a single byte and represented in a drawing in this specification, the more significant (high-order) bits are shown toward the left and less significant (low-order) bits toward the right.
- Multiple-byte fields are drawn with the more significant bytes toward the left and the less significant bytes toward the right.
- Multiple-byte fields are transferred in network byte order.
Outline

1. Introduction
2. Overview
3. Protocol Description
4. Service Attribute Definitions
Bluetooth protocol stack

- Applications
  - AT Com
  - OBEX
  - TCP/IP
  - PPP
- SDP
- L2CAP
- HCI
- Audio
- Link Manager (LM)
- Baseband
- Bluetooth Radio
Steps to Find and Connect to a Service

1. Establish L2CAP connection to remote device using channel identified by PSM = 0x0001.
2. Search for specific class of service, or browse for services.
3. Retrieve attributes needed to connect to the chosen service.
4. Establish a separate (non-SDP) connection to use the service.
SDP Client/Server Model

• An SDP server is any Bluetooth device which offers a service or services to other Bluetooth devices (a service is a feature usable by another device).
• Information about services is maintained in SDP databases.
• Each SDP server maintains its own database; there is no centralized database.
• SDP clients use services provided by servers.
SDP Client/Server Model (cont’d)

- Servers and clients exchange information about services using service records.
- A device wanting to find out about services in the area is an SDP client.
- A device offering services is an SDP server. Devices can simultaneously be both clients (using services) and servers (offering services).
SDP Client-Server Interaction

Client Application

SDP Client

SDP request

SDP Response

Server Application

SDP Server
SDP Client-Server Interaction

Client

Application

SDP

L2CAP

Baseband

Server

Application

SDP

L2CAP

Baseband
SDP Client-Server Interaction

• SDP involves communication between an SDP server and an SDP client.
• The server maintains a list of service records that describe the characteristics of services associated with the server.
• Each service record contains information about a single service.
• If the client, or an application associated with the client, decides to use a service, it must open a separate connection to the service provider in order to utilize the service.
Service Records

Service record

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
Service Attribute 2
Service Attribute 3
... (repeated)
Service Attribute N

Service Attribute 1
SDP Client-Server Interaction (cont’d)

- There is a maximum of **one** SDP server per Bluetooth device.
- If multiple applications on a device provides services, an SDP server may act on behalf of those service providers.
- Similarly, multiple client applications may utilise an SDP client to query servers on behalf of the client applications.
Setting up an SDP session

LocDev
(SDP Client)

RemDev
(SDP Server)

Inquiry
Paging

LMP_host_connection_req
LMP_accepted
LMP_name_req
LMP_name_res
Authentication
LMP_Setup_complete
LMP_Setup_complete
Setting up an SDP session (cont’d)

LocDev
(SDP Client)

RemDev
(SDP Server)

L2CAP Connection Setup

L2CAP_connection_req
L2CAP_connection_res

SDP Session

SDP_inquiries
SDP_responses

Disconnect

Terminate Connection
Service Record

• All of the information about a service maintained by an SDP server is contained within a single service record.

<table>
<thead>
<tr>
<th>Service Attribute 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Attribute 2</td>
</tr>
<tr>
<td>Service Attribute 3</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Service Attribute N</td>
</tr>
</tbody>
</table>
Service Record (cont’d)

• Service record handle:
  – A 32-bit number that uniquely identifies each service record
  – each handle is unique only within each SDP server
  – 0x00000000 represents the SDP server itself

• A service record contains attributes that are specific to a service class as well as universal attributes that are common to all services.

<table>
<thead>
<tr>
<th>Service attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute ID</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
## Intercom Profile Service Record

<table>
<thead>
<tr>
<th>ServiceRecordHandle (0x0000)</th>
<th>0x00003253</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceClassIDList (0x0001)</td>
<td>Generic Telephony (0x1204)</td>
</tr>
<tr>
<td>ProtocolDescriptorList (0x0004)</td>
<td>Intercom (0x1110)</td>
</tr>
<tr>
<td>BluetoothProfileDescriptorList (0x0009)</td>
<td>L2CAP (0x0100)</td>
</tr>
<tr>
<td>ServiceName (offset 0x0000)</td>
<td>TCS-BIN (0x0005)</td>
</tr>
<tr>
<td></td>
<td>Intercom (0x1110)</td>
</tr>
<tr>
<td></td>
<td>0x0100 (Profile Version)</td>
</tr>
<tr>
<td></td>
<td>&quot;Intercom&quot;</td>
</tr>
</tbody>
</table>
Data Element

- In the Service Discovery Protocol, an attribute id or an attribute value is often represented as a data element.
- The format of a data element follows the TLV (type-length-value) convention.

![Data Element Diagram]

Type | Size Index
--- | ---
1 | 1

Attribute ID or Attribute Value

Length: 5 + 3 + 16 = 24
## Data Element -- Type Descriptor

<table>
<thead>
<tr>
<th>Type Descriptor Value</th>
<th>Valid Size Descriptor Values</th>
<th>Type Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Nil, the null type</td>
</tr>
<tr>
<td>1</td>
<td>0, 1, 2, 3, 4</td>
<td>Unsigned Integer</td>
</tr>
<tr>
<td>2</td>
<td>0, 1, 2, 3, 4</td>
<td>Signed twos-complement integer</td>
</tr>
<tr>
<td>3</td>
<td>1, 2, 4</td>
<td>UUID, a universally unique identifier</td>
</tr>
<tr>
<td>4</td>
<td>5, 6, 7</td>
<td>Text string</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>Boolean</td>
</tr>
<tr>
<td>6</td>
<td>5, 6, 7</td>
<td>Data element sequence, a data element whose data field is a sequence of data elements</td>
</tr>
<tr>
<td>7</td>
<td>5, 6, 7</td>
<td>Data element alternative, a data element whose data field is a sequence of data elements from which one data element is to be selected.</td>
</tr>
<tr>
<td>8</td>
<td>5, 6, 7</td>
<td>URL, a uniform resource locator</td>
</tr>
<tr>
<td>9-31</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>
## Data Element -- Size Descriptor

<table>
<thead>
<tr>
<th>Size Index</th>
<th>Additional bits</th>
<th>Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1 byte. Exception: if the data element type is nil, the data size is 0 bytes.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2 bytes</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4 bytes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>8 bytes</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>16 bytes</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>The data size is contained in the additional 8 bits, which are interpreted as an unsigned integer.</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>The data size is contained in the additional 16 bits, which are interpreted as an unsigned integer.</td>
</tr>
<tr>
<td>7</td>
<td>32</td>
<td>The data size is contained in the additional 32 bits, which are interpreted as an unsigned integer.</td>
</tr>
</tbody>
</table>
Data Element Examples

Nil is represented as:

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Size Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

A 16-bit signed integer is represented as:

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Size Index</th>
<th>16-bit data value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

The 3 character ASCII string "Hat" is represented as:

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Size Index</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'H'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'a'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'t'</td>
</tr>
</tbody>
</table>
```

Center for Telecommunication Research
Service Attribute

• Each service attribute describes a single characteristics of a service.

Service Attribute

<table>
<thead>
<tr>
<th>Attribute ID</th>
<th>Attribute Value</th>
</tr>
</thead>
</table>

• Examples:
  – ServiceClassIDList : list of classes of which the service is an instance.
  – ServiceID : identifies a specific instance of a service.
  – IconURL : Icon image that may be used to represent a service.
Attribute ID

- Attribute ID:
  
  A **16-bit unsigned integer** that distinguishes each service attribute from other service attributes within a service record.

- A service class definition specifies each of the attribute IDs for a service class and assigns a meaning to the attribute value associated with each attribute ID.

- All services belonging to a given service class assign the same meaning to each particular attribute ID.
Attribute Value

- The attribute value is a variable length field whose meaning is determined by the attribute ID associated with it and by the service class of the service record in which the attribute is contained.
## Service Attribute Example

### Data element

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Type = 1 (Unsigned Integer)</td>
<td>Size = 1 (2 bytes)</td>
<td>AttributeID = 0x0000 (ServiceRecordHandle)</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>AttributeID = 0x0000 (Service RecordHandle)</td>
<td>Type = 1 (Unsigned Integer)</td>
<td>Size = 2 (4 bytes)</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>Service RecordHandle = 0x25413698</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service RecordHandle = 0x25413698</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Service Attribute Example (cont’d)

#### Data element containing a data element sequence

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type = 1 (Unsigned Integer)</td>
<td>Size = 1 (2 bytes)</td>
<td>AttributeID = 0x00000108 (HeadsetServiceClass)</td>
<td>Type = 1 (Unsigned Integer)</td>
<td>Size = 1 (2 bytes)</td>
<td>AttributeID = 0x0001 (ServiceClassIDList)</td>
<td>Type = 6 (Data element sequence)</td>
<td>Size = 5 (additional 1 byte)</td>
</tr>
<tr>
<td></td>
<td>0x0A (Bytes)</td>
<td>Type = 3 (UUID)</td>
<td>Size = 2 (4 bytes)</td>
<td>AttributeID = 0x00000108 (HeadsetServiceClass)</td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeID = 0x00000108 (HeadsetServiceClass)</td>
<td>Type = 6 (Data element sequence)</td>
<td>Size = 5 (additional 1 byte)</td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td>Type = 6 (Data element sequence)</td>
<td>Size = 5 (additional 1 byte)</td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td>Type = 6 (Data element sequence)</td>
<td>Size = 5 (additional 1 byte)</td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td>Type = 6 (Data element sequence)</td>
<td>Size = 5 (additional 1 byte)</td>
<td>AttributeID = 0x00001203 (GenericAudio ServiceClass)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Service Class

- Each service is an instance of a service class.
- The service class definition provides the definitions of all attributes contained in service records that represent instances of that class.
- Each service class is also assigned a unique identifier.
- The service class identifier is contained in the attribute value for the ServiceClassIDList attribute, and is represented as a UUID (Universally Unique ID).
Service Class (cont’d)

- Typically, each service class is a subclass of another class whose identifier is contained in the list.
- The service class identifiers in the ServiceClassIDList attribute are listed in order from the most specific class to the most general class.
A Printer Service Class Example

- A color postscript printer with duplex capability might conform to 4 Service Class definitions:

  - DuplexColorPostscriptPrinterServiceClassID
  - ColorPostscriptPrinterServiceClassID
  - PostscriptPrinterServiceClassID
  - PrinterServiceClassID
Searching for Service

• The Service Search transaction allows a client to retrieve the service record handles for particular service records based on the values of attributes contained within those service records.

• Once an SDP client has a service record handle, it may easily request the values of specific attributes.

• This capability is provided to search only for attributes whose values are Universally Unique Identifiers (UUIDs), but not for arbitrary attributes.
A UUID is a universally unique identifier that is guaranteed to be unique across all space and all time.

- UUIDs can be independently created in a distributed fashion.
- No central registry of assigned UUIDs is required.
- A UUID is a 128-bit value.
- A range of UUID values has been pre-allocated for assignment to often-used, registered purposes.
UUID (cont’d)

- Bluetooth Base UUID:
  00000000-0000-1000-7007-00805F9B34FB

- 16-bit UUID:
  128_bit_value=16_bit_value * 2^{96} + BT_Base_UUID

- 32-bit UUID:
  128_bit_value=32_bit_value * 2^{96} + BT_Base_UUID
Service Search Pattern

- A service search pattern is a list of UUIDs used to locate matching service records.
- A service search pattern is said to match a service record if each and every UUID in the service search pattern is contained within any of the service record’s attribute values.
- A service search pattern does not match a service record if at least one UUID in the service search pattern is not contained within the service record’s attribute value.
Browsing for Services

• Normally, a client searches for services based on some desired characteristics (UUID) of the services.

• However, it is desirable to discover which type of services are described by an **SDP server’s service records** without any a priori information about the services. -- **Browsing**

• The mechanism for browsing for services is based on an attribute shared by all service classes. This attribute is called the **BrowseGroupList** attribute.
Browsing for Services (cont’d)

• The value of $BrowseGroupList$ contains a list of UUIDs.
• Each UUID represents a browse group with which a service may be associated for the purpose of browsing.
• When a client desires to browse an SDP server’s services, it creates a service search pattern containing the UUID (00001002-0000-1000-7007-00805F9B34FB) that represents the root browse group.
Browsing for Services (cont’d)

- All services that may be browsed at the top level are made members of the **root browse group** by having the root browse group’s UUID as a value within the `BrowseGroupList` Attribute.
- Services offered by an SDP server may be organized in a browse group hierarchy, by defining additional browse groups below root group.
- Additional browse group is described by a service record with a service class of `BrowseGroupDescriptor`. 
## Browsing hierarchy

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Service Class</th>
<th>Attribute Name</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP</td>
<td>BrowseGroupDescriptor</td>
<td>BrowseGroupList</td>
<td>PublicBrowseRoot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GroupID</td>
<td>SDP_Group</td>
</tr>
<tr>
<td>L2CAP</td>
<td>BrowseGroupDescriptor</td>
<td>BrowseGroupList</td>
<td>PublicBrowseRoot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GroupID</td>
<td>L2CAP_Group</td>
</tr>
<tr>
<td>TCS-Bin</td>
<td>BrowseGroupDescriptor</td>
<td>BrowseGroupList</td>
<td>PublicBrowseRoot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GroupID</td>
<td>TCS_Group</td>
</tr>
<tr>
<td>SDAP</td>
<td>ServiceDiscoveryServer</td>
<td>ServiceClassID</td>
<td>BrowseGroupList</td>
</tr>
<tr>
<td>Cordless</td>
<td>CordlessTelephony</td>
<td>BrowseGroupList</td>
<td>TCS_Group</td>
</tr>
<tr>
<td>Intercom</td>
<td>Intercom</td>
<td>BrowseGroupList</td>
<td>TCS_Group</td>
</tr>
</tbody>
</table>
Browsing hierarchy

GroupID: SDP_Group
- SDP
  - SDAP

GroupID: L2CAP_Group
- L2CAP
  - Cordless
  - Intercom

GroupID: TCS_Group
- Public Browse Root
  - TCS-BIN
Outline

1. Introduction
2. Overview
3. Protocol Description
4. Service Attribute Definitions
Protocol Description

- SDP uses a request/response model where each transaction consists of one request PDU (protocol data unit) and one response PDU.
- When SDP utilizes the Bluetooth L2CAP transport protocol, multiple SDP PDUs may be sent in a single L2CAP packet.
- SDP transfers multiple-byte fields in standard network byte order.
PDU Format

- Every SDP PDU consists of a PDU header followed by PDU-specific parameters.

<table>
<thead>
<tr>
<th>Header</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU ID</td>
<td>Parameter 1</td>
</tr>
<tr>
<td>1 Byte</td>
<td>Parameter 2</td>
</tr>
<tr>
<td></td>
<td>Parameter N</td>
</tr>
<tr>
<td>2 Bytes</td>
<td>ParameterLength</td>
</tr>
<tr>
<td>2 Bytes</td>
<td></td>
</tr>
</tbody>
</table>

ParameterLength Bytes
# PDU ID

**PDU ID:** The PDU ID field identifies the type of PDU, i.e., its meaning and the specific parameters.

<table>
<thead>
<tr>
<th>Value</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>The PDU ID field identifies the type of PDU, i.e., its meaning and the specific parameters.</td>
</tr>
<tr>
<td>0x00</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x01</td>
<td>SDP_ErrorResponse</td>
</tr>
<tr>
<td>0x02</td>
<td>SDP_ServiceSearchRequest</td>
</tr>
<tr>
<td>0x03</td>
<td>SDP_ServiceSearchResponse</td>
</tr>
<tr>
<td>0x04</td>
<td>SDP_ServiceAttributeRequest</td>
</tr>
<tr>
<td>0x05</td>
<td>SDP_ServiceAttributeResponse</td>
</tr>
<tr>
<td>0x06</td>
<td>SDP_ServiceSearchAttributeRequest</td>
</tr>
<tr>
<td>0x07</td>
<td>SDP_ServiceSearchAttributeResponse</td>
</tr>
<tr>
<td>0x07-0xFF</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Transaction Type

- SDP_ServiceSearchRequest
  SDP_ServiceSearchResponse
- SDP_ServiceAttributeRequest
  SDP_ServiceAttributeResponse
- SDP_ServiceSearchAttributeRequest
  SDP_ServiceSearchAttributeResponse
- Any Request: SDP_ErrorResponse
Partial Responses and Continuation State

Continuation State Format:

<table>
<thead>
<tr>
<th>InfoLength</th>
<th>Continuation Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>InfoLength bytes</td>
</tr>
</tbody>
</table>

- Some SDP requests may require responses that are larger than can fit in a single response PDU.
- SDP server will generate a partial response along with a continuation state parameter.
Partial Responses and Continuation State (cont’d)

- The continuation state parameter can be supplied by the client in a subsequent request to retrieve the next portion of the complete response.
- The format of the continuation information is not standardized among SDP servers.
- The continuation state parameter is a variable length field whose first byte contains the number of additional bytes of continuation information in the field.
Partial Responses and Continuation State (cont’d)

- Each continuation state parameter is meaningful only to the SDP server that generated it.
- After a client receives a partial response and the accompanying continuation state parameter, it can re-issue the original request (with a new transaction ID) and include the continuation state in the new request indicating to the server that the remainder of the original response is desired.
- The maximum allowable value of the InfoLength field is 16 (0x10).
Error Handling

- If the server determines that a request is improperly formatted or for any reason the server cannot respond with the appropriate PDU type, it will respond with an SDP_ErrorResponse PDU.

<table>
<thead>
<tr>
<th>PDU Type</th>
<th>PDU ID</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP_ErrorResponse</td>
<td>0x01</td>
<td>ErrorCode, ErrorInfo</td>
</tr>
</tbody>
</table>
## ErrorCodes

<table>
<thead>
<tr>
<th>Value</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>The ErrorCode identifies the reason that an SDP_ErrorResponse PDU was generated.</td>
</tr>
<tr>
<td>0x0000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0001</td>
<td>Invalid/unsupported SDP version</td>
</tr>
<tr>
<td>0x0002</td>
<td>Invalid Service Record Handle</td>
</tr>
<tr>
<td>0x0003</td>
<td>Invalid request syntax</td>
</tr>
<tr>
<td>0x0004</td>
<td>Invalid PDU Size</td>
</tr>
<tr>
<td>0x0005</td>
<td>Invalid Continuation State</td>
</tr>
<tr>
<td>0x0006</td>
<td>Insufficient Resources to satisfy Request</td>
</tr>
<tr>
<td>0x0007-0xFFFF</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
ErrorInfo

<table>
<thead>
<tr>
<th>Value</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error-specific</td>
<td>ErrorInfo is an ErrorCode-specific parameter. Its interpretation depends on the ErrorCode parameter. The currently defined ErrorCode values do not specify the format of an ErrorInfo field.</td>
</tr>
</tbody>
</table>
ServiceSearchTransaction

Client → SDP_ServiceSearchRequest → Server

Server → SDP_ServiceSearchResponse → Client
ServiceSearchRequest

<table>
<thead>
<tr>
<th>PDU Type</th>
<th>PDU ID</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP_ServiceSearchRequest</td>
<td>0x02</td>
<td>ServiceSearchPattern, MaximumServiceRecordCount, ContinuationState</td>
</tr>
</tbody>
</table>

- The SDP client generates an SDP_ServiceSearchRequest to locate service records that match the service search pattern given as the first parameter of the PDU.
### ServiceSearchRequest Example

<table>
<thead>
<tr>
<th>PDU ID = 0x02</th>
<th>Transaction ID = 0x0001</th>
<th>Parameter Length = 0x000A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 6</td>
<td>Size = 5 Bytes</td>
<td>Type = 3</td>
</tr>
<tr>
<td></td>
<td>Size index = 5</td>
<td>Size index = 2</td>
</tr>
<tr>
<td>HeadsetClassID = 0x00001108</td>
<td>MaximumServiceRecord Count = 0x0003</td>
<td>Continuation State = 0x00</td>
</tr>
</tbody>
</table>

**ServiceSearchRequest**
ServiceSearchResponse

<table>
<thead>
<tr>
<th>PDU Type</th>
<th>PDU ID</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP_ServiceSearchResponse</td>
<td>0x03</td>
<td>TotalServiceRecordCount, CurrentServiceRecordCount, ServiceRecordHandleList, ContinuationState</td>
</tr>
</tbody>
</table>

- The SDP server generates an SDP_ServiceSearchResponse upon receipt of a valid SDP_ServiceSearchRequest. The response contains a list of service record handles for service records that match the service search pattern given in the request.
### ServiceSearchResponse Example

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU ID = 0x03</td>
<td>Transaction ID = 0x0001</td>
<td>Parameter Length = 0x000D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TotalServiceRecordCount = 0x0002</td>
<td>CurrentServiceRecordCount = 0x0002</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServiceRecordHandle = 0x23558468</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServiceRecordHandle = 0x89688758</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuation State = 0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ServiceAttribute Transaction

Client → Server

SDP_ServiceAttributeRequest → SDP_ServiceAttributeResponse
The SDP client generates an SDP_ServiceAttributeRequest to retrieve specified attribute values from a specific service record. The service record handle of the desired service record and a list of desired attribute IDs to be retrieved from that service record are supplied as parameters.
**ServiceAttributeRequest Example**

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU ID = 0x04</td>
<td>Transaction ID = 0x0002</td>
<td>Parameter Length = 0x000C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ServiceRecordHandle = 0x23558468</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXimumAttributeByteCount = 0x0080</td>
<td>Type = 6</td>
<td>Size = 3 Bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type = 1</td>
<td>Size index = 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AttributeID = 0x0004</td>
<td>(ProtocolDescriptorList)</td>
<td>Continuation State = 0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

*ServiceAttributeRequest*
The SDP server generates an SDP_ServiceAttributeResponse upon receipt of a valid SDP_ServiceAttributeRequest. The response contains a list of attributes (both attribute ID and attribute value) from the requested service record.
# ServiceAttributeResponse Example

<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDU ID = 0x05</td>
<td>Transaction ID = 0x0002</td>
<td>Parameter Length = 0x001A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AttributeListByteCount = 0x17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 6 Size index = 5</td>
</tr>
<tr>
<td>Type = 1 Size index = 1</td>
</tr>
<tr>
<td>AttributeID = 0x0004 (ProtocolDescriptorList)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type = 6 Size index = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 6 Size index = 5</td>
</tr>
<tr>
<td>Type = 6 Size index = 5</td>
</tr>
<tr>
<td>Size = 5 Bytes</td>
</tr>
<tr>
<td>Type = 3 Size index = 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L2CAP Protocol = 0x00000100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 6 Size index = 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Size = 7 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 3 Size index = 2</td>
</tr>
<tr>
<td>RFCOMM Protocol = 0X00000003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RFCOMM protocol = 0X00000003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = 1 Size index = 0</td>
</tr>
<tr>
<td>Server Channel Number = 2</td>
</tr>
<tr>
<td>Continuation State = 0x00</td>
</tr>
</tbody>
</table>
ServiceSearchAttribute Transaction

- Advantage:
  - reduce the total number of SDP transactions
- Disadvantage:
  - more complex
  - require more bytes than separate SDP_ServiceSearch and SDP_ServiceAttribute transactions.
**ServiceSearchAttributeRequest**

<table>
<thead>
<tr>
<th>PDU Type</th>
<th>PDU ID</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP_ServiceSearchAttributeRequest</td>
<td>0x06</td>
<td>ServiceSearchPattern, MaximumAttributeByteCount, AttributeIDList, ContinuationState</td>
</tr>
</tbody>
</table>

- The SDP_ServiceSearchAttributeRequest transaction combines the capabilities of the SDP_ServiceSearchRequest and the SDP_ServiceAttributeRequest into a single request.
- SDP_ServiceSearchAttributeRequest contains both a service search pattern and a list of attributes to be retrieved from service records that match the service search pattern.
ServiceSearchAttributeResponse

<table>
<thead>
<tr>
<th>PDU Type</th>
<th>PDU ID</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDP_ServiceSearchAttributeResponse</td>
<td>0x07</td>
<td>AttributeLists[ByteCount, AttributeLists, ContinuationState]</td>
</tr>
</tbody>
</table>

The SDP server generates an SDP_ServiceSearchAttributeResponse upon receipt of a valid SDP_ServiceSearchAttributeRequest. The response contains a list of attributes (both attribute ID and attribute value) from the service records that match the requested service search pattern.
Outline

1. Introduction
2. Overview
3. Protocol Description
4. Service Attribute Definitions
Service Attribute Definitions

- The Bluetooth spec. currently lists three sets of attribute definitions:
  - Universal Attribute Definitions
  - ServiceDiscoveryServer Service Class Attribute Definitions
  - BrowseGroupDescriptor Service Class Attribute Definitions
Universal Attribute Definitions

- Universal attributes are those service attributes whose definitions are common to all service records.
- Only two attributes are required to exist in every service record instance: ServiceRecordHandle and ServiceClassIDList.
- All other service attributes are optional.
**ServiceRecordHandle**

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute ID</th>
<th>Attribute Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceRecordHandle</td>
<td>0x0000</td>
<td>32-bit unsigned integer</td>
</tr>
</tbody>
</table>

- A service record handle is a 32-bit number that uniquely identifies each service record within an SDP server.
ServiceClassIDList

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute ID</th>
<th>Attribute Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceClassIDList</td>
<td>0x0001</td>
<td>Data Element Sequence</td>
</tr>
</tbody>
</table>

- The ServiceClassIDList attribute consists of a data element sequence in which each data element is a UUID representing the service classes that a given service record conforms to.
ProtocolDescriptorList

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute ID</th>
<th>Attribute Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProtocolDescriptorList</td>
<td>0x0004</td>
<td>Data Element Sequence or Data Element Alternative</td>
</tr>
</tbody>
</table>

- The ProtocolDescriptorList attribute describes one or more protocol stacks that may be used to gain access to the service described by the service record.
ProtocolDescriptorList Examples

• *IrDA-like printer*
  
  (( L2CAP, PSM=RFCOMM ), ( RFCOMM, CN=1 ),
  ( PostscriptStream ))

• *IP Network Printing*
  
  (( L2CAP, PSM=RFCOMM ), ( RFCOMM, CN=2 ), ( PPP ), ( IP ),
  ( TCP ), ( IPP ))

• Synchronization Protocol Descriptor Example
  
  (( L2CAP, PSM=0x1001 ), ( RFCOMM, CN=1 ), ( Obex ), ( vCal ))
  (( L2CAP, PSM=0x1002 ), ( RFCOMM, CN=1 ), ( Obex ),
  ( otherSynchronisationApplication )))
SDP Usage

- Bluetooth Piconet Minder: present a view of available devices and services in proximity (in a piconet) to the user and to other applications.
- Every profile includes an SDP service record to be used when implementing that profile.
- The Service Discovery Application Profile (SDAP) offers guidance for application interactions with SDP.
SDP Summary

- Service search vs. service browsing
- Service record and service attributes
- Concept of data element
- SDP protocol