Advertising Bluetooth services using Symbian C++

Overview

`RSdp` provides a session to the Service Discovery Database. It is used to create subsessions to database functionality. A client must create and connect a session before using a RSdpDatabase subsession to access the database.

`RSdpDatabase` is a subsession of the SDP through which service records and their attributes can be added, deleted, and updated.

The following example shows how to advertise a certain (KBT_serviceID 0x10ff) Bluetooth service. Discovering this particular service is explained in Archived:Discovering Bluetooth services using Symbian C++.

Listening to the Bluetooth connection is explained in Archived:Establishing a Bluetooth connection using Symbian C++. The channel number parameter in `CMyServiceAdvertiser::StartAdvertiserL(TInt aChannel)` method comes there.

MMP file

The following capabilities and libraries are required:

```
CAPABILITY LocalServices
LIBRARY sdpagent.lib
LIBRARY sdpdatabase.lib
```

Header file

```
#include <btsdp.h>
#include <bt_sock.h>

// The service id that identifies the service. This id will be
// used when advertising the service and discovering the service.
#define KBT_serviceID 0x10ff

// Service name and description for our service
_LIT(KBTServiceName, "BTpmp");
_LIT(KBTServiceDesc, "BTpmp");

// Service discovery protocol session
RSdp iSdp;

// Service discovery database (sdp)
RSdpDatabase iSdpDB;

// Service record
TSdpServRecordHandle iRecord;

// Service record state
TInt iRecordState;
```

Source file

Start the service advertiser on a given channel. Entry to service discovery database will be entered describing the advertised service.

```
void CMyServiceAdvertiser::StartAdvertiserL(TInt aChannel)
    {
    // Open sdp session
    User::LeaveIfError(iSdp.Connect());
    // Open sdp database session
    User::LeaveIfError(iSdpDB.Open(iSdp));

    // Create a record of the correct service class
    TUUID serviceUUID(KBT_serviceID);
    iSdpDB.CreateServiceRecordL(serviceUUID, iRecord);

    // Add a protocol to the record
    CSdpAttrValueDES* protocolDescriptorList = CSdpAttrValueDES::NewDESL(NULL);
    CleanupStack::PushL(protocolDescriptorList);

    TBuf8<1> channel;
    channel.Append((TChar)aChannel);

    // Create protocol list for our service
    protocolDescriptorList
    ->StartListL()   //  list of protocols required for this method
        ->BuildDESL()
        ->StartListL()
            ->BuildUUIDL(KL2CAP)
        ->EndListL()

        ->BuildDESL()
        ->StartListL()
            ->BuildUUIDL(KRFCOMM)
            ->BuildUintL(channel)
        ->EndListL()
    ->EndListL();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    // Set protocol list to the record
    iSdpDB.UpdateAttributeL(iRecord, KSdpAttrIdProtocolDescriptorList,
        *protocolDescriptorList);
    CleanupStack::PopAndDestroy(protocolDescriptorList);

    // Add a name to the record
    iSdpDB.UpdateAttributeL(iRecord,
                                KSdpAttrIdBasePrimaryLanguage +
                                    KSdpAttrIdOffsetServiceName,
                                KBTServiceName);

    // Add a description to the record
    iSdpDB.UpdateAttributeL(iRecord,
                                KSdpAttrIdBasePrimaryLanguage +
                                    KSdpAttrIdOffsetServiceDescription,
                                KBTServiceDesc);

    // Set service available
    UpdateAvailabilityL(ETrue);
    }
```

Set the availability of the advertised service. The service record on the service discovery database will be updated accordingly.

```
void CMyServiceAdvertiser::UpdateAvailabilityL(TBool aAvailable)
    {
    TInt state = aAvailable ? 0xFF : 0x00;

    // Set availability
    iSdpDB.UpdateAttributeL(iRecord, KSdpAttrIdServiceAvailability, state);

    // Mark record changed
    iSdpDB.UpdateAttributeL(iRecord, KSdpAttrIdServiceRecordState,
        ++iRecordState);
    }
```

Stop advertising the service

```
void CMyServiceAdvertiser::StopAdvertiserL()
    {
    if ( iRecord!=NULL )
        {
        // Delete record from service discovery database
        iSdpDB.DeleteRecordL(iRecord);

        // Close sdp and sdp db sessions
        iSdpDB.Close();
        iSdp.Close();
        iRecord=NULL;
        }
    }
```

Postconditions

The Bluetooth service is advertised and can be discovered as explained in Archived:Discovering Bluetooth services using Symbian C++.

See also

Archived:Discovering Bluetooth devices using Symbian C++

Archived:Discovering Bluetooth services using Symbian C++

Archived:Establishing a Bluetooth connection using Symbian C++

Archived:RHostResolver and redundant display of access point selection dialog (Known Issue)

Example application

This code snippet has been used in the S60 Platform: Bluetooth Point-to-multipoint Example application.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.