Archived:How to detect a bluetooth device being in non-discoverable (private) mode

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

This article shows how to detect Bluetooth devices that are in private mode, using SDP (Service Discovery Protocol).

Overview

It is simple to detect all devices being in discoverable (public) mode. There is "btdiscovery" example in S60 v2 SDKs illustrating how to do this. A short description is also here: A simple way of discovering and selecting a single Bluetooth device

Unfortunately there is no good method of discovering devices being in non-discoverable mode. However, it is possible to detect a specific device if its bluetooth address is known. It can be done by opening a connection to the device.

There are two challenges with this approach:

1. different devices implement different features and a device can be detected only if it listens on the PSM (bluetooth port) we are trying to connect to,
2. any connection attempt on a non-paired device results in a confirmation dialog displayed on the detected device which must be accepted by the user to let detect the device.

Both issues can be solved by taking advantage of the fact all bluetooth devices should implement SDP (Service Discovery Protocol). SDP itself is implemented on top of low level L2CAP protocol, it listens on PSM == 0x01 and connections to it do not require to be accepted by the device user.

This approach has been successfuly implemented in BlueRadar.

Code

Library needed:

```
LIBRARY esock.lib bluetooth.lib
```

The source code is as follows:

```
// declarations
RSocketServ iSocketServer;

// detect device with bluetooth address in devAddr
void CBtDetectorEngine::DetectL(TBTDevAddr devAddr)
{
    User::LeaveIfError(iSocketServer.Connect());
    TBTSockAddr btSockAddr;
    btSockAddr.SetBTAddr(devAddr);
    btSockAddr.SetPort(0x01); // SDP PSM
    iConn = CBtDetectorConnection::NewL(iSocketServer);
    iConn->SetBTSockAddr(btSockAddr);
    iConn->Connect();
}


// BtDetectorConnection.h

#include <e32base.h>
#include <es_sock.h>
#include <bt_sock.h>

class CBtDetectorConnection : MBluetoothSocketNotifier
{
public:
    static CBtDetectorConnection* NewL(RSocketServ& aSocketServer);
    virtual ~CBtDetectorConnection();
    void SetBTSockAddr(TBTSockAddr& aBTSockAddr);
    void Connect();
    void Cancel();
private:
    CBtDetectorConnection(RSocketServ& aSocketServer);
    void ConstructL();
private:
    void HandleConnectCompleteL(TInt aErr);
    void HandleAcceptCompleteL(TInt aErr) {};
    void HandleShutdownCompleteL(TInt aErr);
    void HandleSendCompleteL(TInt aErr) {};
    void HandleReceiveCompleteL(TInt aErr) {};
    void HandleIoctlCompleteL(TInt aErr) {};
    void HandleActivateBasebandEventNotifierCompleteL(TInt aErr, TBTBasebandEventNotification& aEventNotification) {};
private:
    RSocketServ& iSocketServer;
    TBTSockAddr iBTSockAddr;
    CBluetoothSocket* iSocket;
};

// BtDetectorConnection.cpp

#include <bttypes.h>
#include "BtDetectorConnection.h"
```


(C) Copyright Nokia 2013. All rights reserved.

```
CBtDetectorConnection* CBtDetectorConnection::NewL(RSocketServ& aSocketServer)
{
    CBtDetectorConnection* self=new(ELeave) CBtDetectorConnection(aSocketServer);
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop(self);
    return self;
}

void CBtDetectorConnection::ConstructL()
{
    iSocket = CBluetoothSocket::NewL(*this, iSocketServer, KSockSeqPacket, KL2CAP);
}

CBtDetectorConnection::~CBtDetectorConnection()
{
    Cancel();
    CBluetoothSocket* s = iSocket;
    iSocket = NULL;
    delete s;
}

void CBtDetectorConnection::SetBTSockAddr(TBTSockAddr& aBTSockAddr)
{
    iBTSockAddr = aBTSockAddr;
}

void CBtDetectorConnection::Connect()
{
    Cancel();
    iSocket->Connect(iBTSockAddr);
}

void CBtDetectorConnection::HandleConnectCompleteL(TInt aErr)
{
    if (aErr == KErrNone)
    {
        iSocket->Shutdown(RSocket::ENormal);
        // here is the place for notifying that the device has been detected
    }
}

void CBtDetectorConnection::Cancel()
{
    iSocket->CancelConnect();
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.