

Introduction

This article describes how we can get information about remote class of devices (CoD). CoD means what kind of devices is the remote device is, what kind of services it provides etc. In Symbian C++, TBTDeviceClass is used to implement the concept of CoD. TBTDeviceClass class has been declared in btdevice.h file that is publicly available.

How to Retrieve the CoD

Here we describe how to get the CoD by BTLINKManager. Following code example shows how we get the information.

```
void CBTInquiryObject::ConstructL()
{
    _LIT(KBTLINKManagerName, "BTLINKManager");
    TProtocolDesc pInfo;
    TProtocolName protocolName;
    protocolName.Copy( KBTLINKManagerName );

    User::LeaveIfError( iSocketServer.Connect() );
    User::LeaveIfError( iSocketServer.FindProtocol( protocolName, pInfo) );
    User::LeaveIfError( iHostResolver.Open(iSocketServer, pInfo.iAddrFamily, pInfo.iProtocol) );

    iInquirySockAddr.SetAction(KHostResInquiry + KHostResName + KHostResIgnoreCache);
    iInquirySockAddr.SetIAC(KGIAC);
}
```

We get our RunL() called and in RunL() retrieve the CoD information by the following code.

```
void CBTInquiryObject::RunL()
{
    if( iStatus == KErrNone)
    {
        TInquirySockAddr& sa = TInquirySockAddr::Cast( iEntry().iAddr );
        <24> DeviceAddress8(0);
        <40> DeviceName8(0);

        BTAddr().GetReadable(DeviceAddress);
        Copy(DeviceName8, DeviceName);

        <48> DeviceAddress8(0);
        <80> DeviceName8(0);
        Copy(DeviceAddress8, DeviceAddress);
        Copy(DeviceName8, DeviceName);
        LogExtra8(DeviceAddress8, DeviceName8, sa.MajorServiceClass(), sa.MajorClassOfDevice(),
        // Get next device if no previous errors
        iHostResolver.Next( iEntry, iStatus );
        SetActive();
    }
    else
    {
        TBuf8<12> complete(0);
    }
}
```

How_to_handle_Bluetooth_class_of_devices_(CoD)

```
        complete.Copy(_L8("_Complete_")); // just to tell the observer that inquiry is complete
        iObserver.LogInfoL( complete,iStatus.Int() );
    }
}
```

Following code classify the CoD as described in btdevice.h.

```
void CHemeBTBrowserDetailView::LogExtraL(TDesC8 & address, TDesC8 & name, TUint16 aMajServClass, T
{

enum TBluetoothDeviceType {EComputer, EPhone, ELANAccessPoint, EAudioVideo, EUnknown};
TBluetoothDeviceType deviceType=EPhone;

    aBuf.Append(_L("Major Service Class: "));
    if (aMajServClass & 0x0001)
        aBuf.AppendFormat(_L(" Limited Discoverable Mode "));

    if (aMajServClass & 0x0008)
        aBuf.AppendFormat(_L(" Positioning "));

    if (aMajServClass & 0x0010)
        aBuf.AppendFormat(_L(" Networking "));
    if (aMajServClass & 0x0020)
        aBuf.AppendFormat(_L(" Rendering "));
    if (aMajServClass & 0x0040)
        aBuf.AppendFormat(_L(" Capturing "));
    if (aMajServClass & 0x0080)
        aBuf.AppendFormat(_L(" Object Transfer "));
    if (aMajServClass & 0x0100)
        aBuf.AppendFormat(_L(" Audio "));
    if (aMajServClass & 0x0200)
        aBuf.AppendFormat(_L(" Telephony "));
    if (aMajServClass & 0x0400)
        aBuf.AppendFormat(_L(" Information "));

    // Major Device Class is one of these
    deviceType=EUnknown;
    aBuf.Append(_L("Major Device Class: "));
    if (aMajDeviceClass == 0x00)
    {
        deviceType=EUnknown;
        aBuf.AppendFormat(_L(" Miscellaneous"));
    }
    else if (aMajDeviceClass == 0x01)
    {
        deviceType=EComputer;
        aBuf.AppendFormat(_L(" Computer "));
    }
    else if (aMajDeviceClass == 0x02)
    {
        deviceType=EPhone;
        aBuf.AppendFormat(_L(" Phone "));
    }
    else if (aMajDeviceClass == 0x03)
    {
        deviceType=ELANAccessPoint;
        aBuf.AppendFormat(_L(" LAN Access Point "));
    }
    else if (aMajDeviceClass == 0x04)
```

How_to_handle_Bluetooth_class_of_devices_(CoD)

```
{
    deviceType=EAudioVideo;
    aBuf.AppendFormat(_L(" AudioVideo "));
}
else if (aMajDeviceClass == 0x05)
{
    aBuf.AppendFormat(_L(" Peripheral "));
}
else if (aMajDeviceClass == 0x06)
{
    aBuf.AppendFormat(_L(" Imaging"));
}
else
{
    aBuf.AppendFormat(_L(" Unknown"));
}

// Minor Device Class
aBuf.Append(_L("Minor Device Class: "));
switch (deviceType)
{
case EComputer:
    if (aMinDeviceClass == 0x00)
        aBuf.AppendFormat(_L(" Unclassified "));
    else if (aMinDeviceClass == 0x01)
        aBuf.AppendFormat(_L(" Desktop Computer "));
    else if (aMinDeviceClass == 0x02)
        aBuf.AppendFormat(_L(" Server Computer "));
    else if (aMinDeviceClass == 0x03)
        aBuf.AppendFormat(_L(" Laptop "));
    else if (aMinDeviceClass == 0x04)
        aBuf.AppendFormat(_L(" Handheld Computer "));
    else if (aMinDeviceClass == 0x05)
        aBuf.AppendFormat(_L(" Palm Computer "));
    else if (aMinDeviceClass == 0x06)
        aBuf.AppendFormat(_L(" Wearable computer "));
    else
        aBuf.AppendFormat(_L(" Unknown device type "));
    break;

case EPhone:
    if (aMinDeviceClass == 0x00)
        aBuf.AppendFormat(_L(" Unclassified "));
    else if (aMinDeviceClass == 0x01)
        aBuf.AppendFormat(_L(" Cellular Phone"));
    else if (aMinDeviceClass == 0x02)
        aBuf.AppendFormat(_L(" Cordless Phone"));
    else if (aMinDeviceClass == 0x03)
        aBuf.AppendFormat(_L(" Smart Phone "));
    else if (aMinDeviceClass == 0x04)
        aBuf.AppendFormat(_L(" Modem Phone"));
    else if (aMinDeviceClass == 0x05)
        aBuf.AppendFormat(_L(" ISDN Phone"));
    else
        aBuf.AppendFormat(_L(" Unknown device type "));
    break;

case ELANAccessPoint:
    if (aMinDeviceClass == 0x00)
        aBuf.AppendFormat(_L(" 0% utilized "));
    else if (aMinDeviceClass == 0x08)
```

How_to_handle_Bluetooth_class_of_devices_(CoD)

```
        aBuf.AppendFormat(_L(" 1-17% utilized "));
    else if (aMinDeviceClass == 0x10)
        aBuf.AppendFormat(_L(" 17-33% utilized "));
    else if (aMinDeviceClass == 0x18)
        aBuf.AppendFormat(_L(" 33-50% utilized "));
    else if (aMinDeviceClass == 0x20)
        aBuf.AppendFormat(_L(" 50-67% utilized "));
    else if (aMinDeviceClass == 0x28)
        aBuf.AppendFormat(_L(" 67-83% utilized "));
    else if (aMinDeviceClass == 0x30)
        aBuf.AppendFormat(_L(" 83-99% utilized "));
    else if (aMinDeviceClass == 0x38)
        aBuf.AppendFormat(_L(" 100% utilized "));
    else
        aBuf.AppendFormat(_L(" Unknown utilisation "));
    break;

case EAudioVideo:
    if (aMinDeviceClass == 0x00)
        aBuf.AppendFormat(_L(" Unclassified "));
    else if (aMinDeviceClass == 0x01)
        aBuf.AppendFormat(_L(" Headset Profile "));
    else if (aMinDeviceClass == 0x02)
        aBuf.AppendFormat(_L(" Handsfree "));
    else if (aMinDeviceClass == 0x04)
        aBuf.AppendFormat(_L(" MicroPhone "));
    else if (aMinDeviceClass == 0x05)
        aBuf.AppendFormat(_L(" Loudspeaker "));
    else if (aMinDeviceClass == 0x06)
        aBuf.AppendFormat(_L(" Head Phone "));
    else
        aBuf.AppendFormat(_L(" Other Audio Video type "));
    break;
default:
    aBuf.AppendFormat(_L(" Unknown "));

    break;
}
}
```

Example Applications

There is a complete SDP property/folder browsing application that can be downloaded from [\[1\]](#).