

Enumerating Bluetooth Devices Using the Windows Bluetooth API

After looking into the Windows Bluetooth API for research purposes, I decided to write a simple application to enumerate Bluetooth devices in the range of the Bluetooth adapter. A simple console-based enumeration application can be implemented in less than 60 lines of code. The application can enumerate all the devices and get the names of the devices.

First of all I would like to note that the device name of the Bluetooth device is stored as a wide character string. This means we will have to implement the application using the wide character functions, instead of the ANSI functions. More information regarding Unicode and ANSI differences please visit the following web site: <http://www.i18nguy.com/unicode/c-unicode.html>.

The two main Bluetooth API functions we are going to be using are *BluetoothFindFirstDevice* and *BluetoothFindNextDevice*. The first function *BluetoothFindFirstDevice* begins the enumeration of Bluetooth devices. The parameters that this function takes are shown in the table below.

Parameter	Description
BLUETOOTH_DEVICE_SEARCH_PARAMS* pbtsp	A pointer to a BLUETOOTH_DEVICE_SEARCH_PARAMS structure
BLUETOOTH_DEVICE_INFO* pbtDi	A pointer to a BLUETOOTH_DEVICE_INFO structure to receive the Bluetooth device information

To use this function we need to declare two data structures and a handle. Below in section 1.0 is an example of initializing the data.

Section 1.0

```
#include <windows.h>
#include <bthdef.h>
#include <bluetoothapis.h>

int wmain(int argc, wchar_t** argv)
{
    BLUETOOTH_DEVICE_SEARCH_PARAMS BluetoothSearchParams;
    BLUETOOTH_DEVICE_INFO BluetoothDeviceInfo;
    HBLUETOOTH_DEVICE_FIND hBluetoothDevice;

    ZeroMemory(&BluetoothSearchParams, sizeof(BluetoothSearchParams));
    ZeroMemory(&BluetoothDeviceInfo, sizeof(BluetoothDeviceInfo));

    BluetoothSearchParams.dwSize = sizeof(BLUETOOTH_DEVICE_SEARCH_PARAMS);
    BluetoothSearchParams.fReturnAuthenticated= true;
    BluetoothSearchParams.fReturnRemembered = true;
    BluetoothSearchParams.fReturnUnknown = true;
    BluetoothSearchParams.fReturnConnected = true;
    BluetoothSearchParams.fIssueInquiry = true;
    BluetoothSearchParams.cTimeoutMultiplier = 15;
    BluetoothSearchParams.hRadio = NULL;

    BluetoothDeviceInfo.dwSize = sizeof(BluetoothDeviceInfo);

    return 0;
}
```

Once the data structures are filled out accordingly we can start to enumerate the Bluetooth devices. We can now call the *BluetoothFindFirstDevice* function. This will locate the first Bluetooth device in range and return a *HBLUETOOTH_DEVICE_FIND* if one is found. Below in section 2.0 is an example of finding the first Bluetooth device.

Section 2.0

```

    hBluetoothDevice = BluetoothFindFirstDevice(&BluetoothSearchParams,
&BluetoothDeviceInfo);
    if (hBluetoothDevice != NULL)
    {
        // Handle the found device...
        // Continue enumerating Bluetooth devices...
    }
    else
    {
        wprint(L"Unable to find a Bluetooth device.\n");
    }

    return 0;
}

```

If *hBluetoothDevice* is **NULL** after calling *BluetoothFindFirstDevice* it means that it was unable to find a Bluetooth device. If the handle is not **NULL** we can continue enumerating the rest of the Bluetooth devices using the function *BluetoothFindNextDevice* and continue looping until it no longer finds anymore devices.

Parameter	Description
HBLUETOOTH_DEVICE_FIND hFind	A handle for the query obtained in a previous call to <i>BluetoothFindFirstDevice</i>
BLUETOOTH_DEVICE_INFO* pbtdi	A pointer to a <i>BLUETOOTH_DEVICE_INFO</i> structure

Below in section 3.0 is an example of looping to find the rest of the Bluetooth devices.

Section 3.0

```

if (hBluetoothDevice != NULL)
{
    while (true)
    {
        wprintf(L"Found a Bluetooth device!\n");

        if (BluetoothFindNextDevice(hBluetoothDevice, &BluetoothDeviceInfo)
== false)
        {
            break;
        }
    }
}
else
{
    wprintf(L"Unable to find a Bluetooth device.\n");
}

```

The above code loops continuously until *BluetoothFindNextDevice* returns false, which means that there is no longer a Bluetooth device to be found. It will show the message "Found a Bluetooth device!" for each Bluetooth device that is found. Below in code section 4.0 is the complete application code for this guide.

Section 4.0

```

#include <windows.h>
#include <bthdef.h>
#include <bluetoothapis.h>

int wmain(int argc, wchar_t** argv)
{
    BLUETOOTH_DEVICE_SEARCH_PARAMS BluetoothSearchParams;
    BLUETOOTH_DEVICE_INFO BluetoothDeviceInfo;
    HBLUETOOTH_DEVICE_FIND hBluetoothDevice;

    ZeroMemory(&BluetoothSearchParams, sizeof(BluetoothSearchParams));

```

```
ZeroMemory(&BluetoothDeviceInfo, sizeof(BluetoothDeviceInfo));

BluetoothSearchParams.dwSize = sizeof(BLUETOOTH_DEVICE_SEARCH_PARAMS);
BluetoothSearchParams.fReturnAuthenticated= true;
BluetoothSearchParams.fReturnRemembered = true;
BluetoothSearchParams.fReturnUnknown = true;
BluetoothSearchParams.fReturnConnected = true;
BluetoothSearchParams.fIssueInquiry = true;
BluetoothSearchParams.cTimeoutMultiplier = 15;
BluetoothSearchParams.hRadio = NULL;

BluetoothDeviceInfo.dwSize = sizeof(BluetoothDeviceInfo);

hBluetoothDevice = BluetoothFindFirstDevice(&BluetoothSearchParams,
&BluetoothDeviceInfo);

if (hBluetoothDevice != NULL)
{
    while (true)
    {
        wprintf(L"Found a Bluetooth device!\n");

        if (BluetoothFindNextDevice(hBluetoothDevice,
&BluetoothDeviceInfo) == false)
        {
            break;
        }
    }
}
else
{
    wprint(L"Unable to find a Bluetooth device.\n");
}

return 0;
}
```

The above code is a complete example of enumerating Bluetooth devices. There are further functions you can use and test. They can be found at the following link <http://msdn2.microsoft.com/en-us/library/aa362927.aspx>. You can go further and combine Microsoft's Winsock API with the Bluetooth API to send and receive data.