160 驱动 SDK 使用指南

驱动人生

2015 年 6 月 深圳

本文档并不代表供应商或其代理的承诺 ,驱动人生可在不作任何申明的情况下对本文档内容进行修改。160 驱动 SDK 的一切有关权利属于驱动人生公司所有,涉及的软件产品及其后续升级产品均由驱动人生制作并负责维护

联系我们:
地址：深圳市南山区科技园中区科苑路科兴科学园 A 栋 1 单元 19 楼
邮编：518057
电话: 0755-82504701
网址：www.160.com
商务合作：linlingli0912@160.com
技术支持：lishunxiang@160.com
驱动人生欢迎您的任何建议和意见。

2015 年 6 月 深圳

目录

160 驱动 SDK 使用指南.....1
1 概述.....3
1.1 文档说明.....3
1.2 适用对象.....3
1.3 SDK 功能说明.....3
1.4 SDK 的调用方式.....3
2 调用流程.....4
2.1 动态调用使用流程.....4
2.2 静态调用使用流程.....4
2.3 接口说明.....4
2.3.1 DTLGetWirelessNetArray.....4
2.3.2 DTLUninstallDriver.....5
2.3.3 DTLUpdateBestDriver.....5
2.3.4 DTLUpdateLastDriver.....5
2.3.5 DTLFreeWirelessNetArray.....5
2.4 使用示例.....6

1 概述

1.1 文档说明

本文档旨在描述 160 驱动 SDK（以下简称 SDK）功能流程及调用方式，方便产品经理了解 SDK 对产品的作用，方便开发人员快速使用 SDK。

1.2 适用对象

本文档适用于需要检测及安装电脑上的各种硬件设备及外设驱动项目的相关产品及开发人员。

1.3 SDK 功能说明

对已连接到电脑的硬件设备（包括显卡声卡网卡等）的检测查询，驱动下载安装等的功能；

对已插入的 USB 外设（包括 USB 无线网卡，安卓手机等）的检测及通知；

对新插入的 USB 外设的检测及通知；

对移除的 USB 外设的检测及通知；

检测设备的驱动状态是否正常；

查询及下载设备驱动文件包；

提供驱动包大小，下载进度，下载速度等数据；

安装设备驱动包；

获取本机品牌，型号，设备描述等数据；

获取安卓手机品牌名称，型号名称，通俗名称，手机图片，手机屏幕分辨率等数据

注意：当前 SDK 为 bata 版，现仅开放无线网卡功能，更多功能请与我们联系

1.4 SDK 的调用方式

本 SDK 提供两种调用方式

a）动态调用：动态获取 SDK 的 dll 接口地址
b）静态调用：静态链接 SDK 库。

2 调用流程

2.1 动态调用使用流程

第一步：将 dtlsdk 文件夹拷贝到项目所在目录中，将 include\DtlApiInterface.h、include\DtlApiInterface.cpp 两个文件包含到工程中

第二步:程序初始化时调用 CDtlApiInterface 类中 InitInterface 接口初始化 SDK，传入 dtlapi.dll 文件全路径

第三步:调用其他接口获取硬件信息，安装卸载驱动等

2.2 静态调用使用流程

第一步：将 dtlsdk 文件夹拷贝到项目所在目录中，将 include\ dtlapi.h、lib\dtlapi.lib 连个个文件包含到工程中

第二步:直接调用头接口获取硬件信息，安装卸载驱动等

2.3 接口说明

2.3.1 DTLGetWirelessNetArray

◆int __stdcall DTLGetWirelessNetArray(PDTL_WIRELESSNET_DEVICEINFO* OUT ppInfoArr);

◆说明：获取无线网卡信息数组

◆参数：ppInfoArr 无线网卡信息数组

```
typedef struct _DTL_WIRELESSNET_DEVICEINFO
{
    HDEVICE  hDevice;                  //设备句柄
    WCHAR    szName[MAX_PATH];         //设备名称
    GUID     guid;                     //设备类 GUID
    WCHAR    szHwid[MAX_PATH];         //设备硬件 ID
    BOOL     bIsWirelessNet;           //TRUE 为无线网卡，FALSE 为 USB 接口设备
    BOOL     bHasDriver;               //设备是否安装有驱动
    BOOL     bEnableUpdateBestDriver;  //是否支持更新到推荐驱动
    BOOL     bEnableUpdateLastDriver;  //是否支持更新到最新驱动
    BOOL     bEnableUninstDriver;      //是否支持驱动卸载
```

}DTL_WIRELESSNET_DEVICEINFO,*PDTL_WIRELESSNET_DEVICEINFO;

◆返回值：int 返回获取到的设备个数
-1，获取失败
0，无无线网卡
>0,无线网卡个数

2.3.2 DTLUninstallDriver

◆int __stdcall DTLUninstallDriver(HDEVICE IN hDevice);
◆说明：卸载驱动
◆参数：hDevice 设备句柄
◆返回值：int 操作结果

```
#define DTL_RESULT_ERROR        0    //安装失败，卸载失败
#define DTL_RESULT_SUCESS       1    //安装成功，卸载成功
#define DTL_RESULT_REBOOT       2    //安装成功，卸载成功，需要重启
```

2.3.3 DTLUpdateBestDriver

◆int __stdcall DTLUpdateBestDriver(HDEVICE IN hDevice);
◆说明：安装推荐驱动
◆参数：hDevice 设备句柄
◆返回值：int 操作结果，同上

2.3.4 DTLUpdateLastDriver

◆int __stdcall DTLUpdateLastDriver(HDEVICE IN hDevice);
◆说明：安装最新驱动
◆参数：hDevice 设备句柄
◆返回值：int 操作结果，同上

2.3.5 DTLFreeWirelessNetArray

◆void __stdcall DTLFreeWirelessNetArray( );
◆说明：释放内存数据
◆参数：无
◆返回值：无

2.4 使用示例

见 sdktest.cpp 示例工程,测试用例 sdktest.exe

```
1)  #include "stdafx.h"

2)  LOG_GLOBAL_INIT("sdktest.log");

3)  int AnalyzeCmdTest(PDTL_WIRELESSNET_DEVICEINFO pInfo,int num);

4)  #define CMDSTR_UNINSTALL        L"uninstall"
5)  #define CMDSTR_UPDATEBEST       L"updatebest"
6)  #define CMDSTR_UPDATELAST       L"updatelast"

7)  CDtlApiInterface      g_dtlapi;

8)  int _tmain(int argc, _TCHAR* argv[])
9)  {
10)     int         ret                     = 0;
11)     WCHAR   szMod[MAX_PATH]   = {0};

12)     LOGINOUT;
13)     PRT("正在获取设备列表...");

14)     ::GetModuleFileName(NULL, szMod, MAX_PATH);
15)     ::PathAppend(szMod,L"..\\dtlsdk\\dtlapi.dll");

16)     if ( !::PathFileExists(szMod) )
17)         ZeroMemory(szMod,MAX_PATH*sizeof(WCHAR));
18)     if ( !g_dtlapi.InitInterface(szMod) )
19)     {
20)         PRT("error load[%S]",szMod);
21)         return ret;
22)     }

23)     PDTL_WIRELESSNET_DEVICEINFO pInfo = NULL;
24)     int num = g_dtlapi.GetWirelessNetArray(&pInfo);
25)     PRT("\n 获取到的设备个数为[%d]\n",num);
26)     for ( int i = 0; i < num; i++ )
27)     {
28)         PRT("\n---------%d/%d---------",i,num);
29)         PRT("szName[%S]",pInfo[i].szName);
30)         PRT("guid[%S]",GUIDToString(pInfo[i].guid).c_str());
```

```
31)             PRT("szHwid[%S]",pInfo[i].szHwid);
32)             PRT("bIsWirelessNet[%d]",pInfo[i].bIsWirelessNet);
33)             PRT("bHasDriver[%d]",pInfo[i].bHasDriver);
34)             PRT("bEnableUpdateBestDriver[%d]",pInfo[i].bEnableUpdateBestDriver);
35)             PRT("bEnableUpdateLastDriver[%d]",pInfo[i].bEnableUpdateLastDriver);
36)             PRT("bEnableUninstDriver[%d]",pInfo[i].bEnableUninstDriver);
37)         }

38)         if ( num > 0 )
39)         {
40)             ret = AnalyzeCmdTest(pInfo,num);
41)         }
42)         else
43)         {
44)             system("pause");
45)         }

46)         g_dtlapi.FreeWirelessNetArray();
47)         g_dtlapi.CloseInterface();
48)         PRT("apicmd ret[%d].",ret);
49)         return ret;
50)  }

51)  int OperationCmd(WCHAR* szCmd,HDEVICE hDevice)
52)  {
53)         int ret = 0;
54)         if ( szCmd && wcslen(szCmd) > 0 && hDevice )
55)         {
56)             if ( 0 == _wcsicmp(szCmd,CMDSTR_UNINSTALL) )
57)             {
58)                 ret = g_dtlapi.UninstallDriver(hDevice);
59)             }
60)             else if ( 0 == _wcsicmp(szCmd,CMDSTR_UPDATEBEST) )
61)             {
62)                 ret = g_dtlapi.UpdateBestDriver(hDevice);
63)             }
64)             else if ( 0 == _wcsicmp(szCmd,CMDSTR_UPDATELAST) )
65)             {
66)                 ret = g_dtlapi.UpdateLastDriver(hDevice);
67)             }
68)         }
69)         return ret;
70)  }
```

```
71)  int AnalyzeCmdTest(PDTL_WIRELESSNET_DEVICEINFO pInfo,int num)
72)  {
73)      int ret = 0;
74)      do
75)      {
76)          PRT("\n 输入[%S 0]卸载第 0 个设备的驱动",CMDSTR_UNINSTALL);
77)          PRT("输入[%S 0]更新第 0 个设备的推荐驱动",CMDSTR_UPDATEBEST);
78)          PRT("输入[%S 0]更新第 0 个设备的最新驱动",CMDSTR_UPDATELAST);
79)          PRT("输入[quit 0]退出\n");
80)          WCHAR cmd[MAX_PATH] = {0};
81)          int index = 0;
82)          scanf_s("%S %d",cmd,MAX_PATH,&index);
83)          if ( 0 == _wcsicmp(cmd,L"quit") )
84)          {
85)              break;
86)          }
87)          else if ( index < num )
88)          {
89)              ret = OperationCmd(cmd,pInfo[index].hDevice);
90)              switch(ret)
91)              {
92)              case DTL_RESULT_ERROR:PRT("操作失败");break;
93)              case DTL_RESULT_SUCESS:PRT("操作成功");break;
94)              case DTL_RESULT_REBOOT:PRT("操作成功,需要重启");break;
95)              }
96)          }
97)      } while (true);
98)      return ret;
99)  }
```