RegisterDeviceNotification function

Registers the device or type of device for which a window will receive notifications.

Syntax

C++

```
HDEVNOTIFY WINAPI RegisterDeviceNotification(
  _In_  HANDLE hRecipient,
  _In_  LPVOID NotificationFilter,
  _In_  DWORD Flags
);
```

Parameters

hRecipient [in]
A handle to the window or service that will receive device events for the devices specified in the *NotificationFilter* parameter. The same window handle can be used in multiple calls to **RegisterDeviceNotification**.

Services can specify either a window handle or service status handle.

NotificationFilter [in]
A pointer to a block of data that specifies the type of device for which notifications should be sent. This block always begins with the **DEV_BROADCAST_HDR** structure. The data following this header is dependent on the value of the **dbch_devicetype** member, which can be **DBT_DEVTYP_DEVICEINTERFACE** or **DBT_DEVTYP_HANDLE**. For more information, see Remarks.

Flags [in]
This parameter can be one of the following values.

Value	Meaning
DEVICE_NOTIFY_WINDOW_HANDLE 0x00000000	The *hRecipient* parameter is a window handle.
DEVICE_NOTIFY_SERVICE_HANDLE 0x00000001	The *hRecipient* parameter is a service status handle.

In addition, you can specify the following value.

Value	Meaning
DEVICE_NOTIFY_ALL_INTERFACE_CLASSES 0x00000004	Notifies the recipient of device interface events for all device interface classes. (The **dbcc_classguid** member is ignored.) This value can be used only if the **dbch_devicetype** member is **DBT_DEVTYP_DEVICEINTERFACE**.

Return value

If the function succeeds, the return value is a device notification handle.

If the function fails, the return value is **NULL**. To get extended error information, call **GetLastError**.

Remarks

Applications send event notifications using the **BroadcastSystemMessage** function. Any application with a top-level window can receive basic notifications by processing the **WM_DEVICECHANGE** message. Applications can use the **RegisterDeviceNotification** function to register to receive device notifications.

Services can use the **RegisterDeviceNotification** function to register to receive device notifications. If a service specifies a window handle in the *hRecipient* parameter, the notifications are sent to the window procedure. If *hRecipient* is a service status handle, **SERVICE_CONTROL_DEVICEEVENT** notifications are sent to the service control handler. For more information about the service control handler, see **HandlerEx**.

Be sure to handle Plug and Play device events as quickly as possible. Otherwise, the system may become unresponsive. If your event handler is to perform an operation that may block execution (such as I/O), it is best to start another thread to perform the operation asynchronously.

Device notification handles returned by **RegisterDeviceNotification** must be closed by calling the **UnregisterDeviceNotification** function when they are no longer needed.

The DBT_DEVICEARRIVAL and DBT_DEVICEREMOVECOMPLETE events are automatically broadcast to all top-level windows for port devices. Therefore, it is not necessary to call **RegisterDeviceNotification** for ports, and the function fails if the **dbch_devicetype** member is **DBT_DEVTYP_PORT**. Volume notifications are also broadcast to top-level windows, so the

function fails if **dbch_devicetype** is **DBT_DEVTYP_VOLUME**. OEM-defined devices are not used directly by the system, so the function fails if **dbch_devicetype** is **DBT_DEVTYP_OEM**.

Examples

For an example, see Registering for Device Notification.

Requirements

Minimum supported client	Windows XP
Minimum supported server	Windows Server 2003
Header	Winuser.h (include Windows.h)
Library	User32.lib
DLL	User32.dll
Unicode and ANSI names	**RegisterDeviceNotificationW** (Unicode) and **RegisterDeviceNotificationA** (ANSI)

See also

Device Management Functions
Device Notifications
BroadcastSystemMessage
HandlerEx
RegisterDeviceNotification
UnregisterDeviceNotification
DEV_BROADCAST_HDR
WM_DEVICECHANGE

Community Additions

Consider using CM_Register_Notification if developing for Windows 8 and later.

If you want to avoid using HWND and you are developing your application for Windows 8 and later, then you may consider using the newer CM_Register_Notification instead of RegisterDeviceNotification. CM_Register_Notification has fewer dependencies and may be faster.


Travis Martin [MSFT]
5/6/2014

RegisterDeviceNotification fails with last error 1066

If RegisterDeviceNotication fails returning a 0 handle with GetLastError() == 1066, then it may be that the struct member alignment (packing) is set wrong when you included dbt.h. Packing needs to be 4 or 8. For example, in the following code:

```
DEV_BROADCAST_DEVICEINTERFACE Filter;
ZeroMemory(&Filter,sizeof Filter);
Filter.dbcc_size = sizeof Filter;   // size gets set to 29 with 1-byte packing or 32 with 4- or 8-byte packing
Filter.dbcc_devicetype = DBT_DEVTYP_DEVICEINTERFACE;
Filter.dbcc_classguid = myDeviceGuid;
hMonitor = RegisterDeviceNotification(hWnd,&Filter,DEVICE_NOTIFY_WINDOW_HANDLE);
```

If the structure packing is set to 1-byte, the DEV_BROADCAST_DEVICEINTERFACE struct is 29 bytes in the example. However sending the dbcc_size field as 29 results in RegisterDeviceNotication returning a NULL handle and error code 1066. If you set the dbcc_size to 32 or just correctly set packing to 4-byte or 8-byte when dbt.h is included, then the RegisterDeviceNotification works properly.

To set the proper packing:

```
#pragma pack(push,8)
#include <dbt.h>
#pragma pack(pop)
OR
#include <pshpack8.h>
#include <dbt.h>
#include <poppack.h>
```

It was annoying having to figure out this packing problem. I give up figuring out why this post has so many extra freaking newlines


some_coder
1/8/2014

Using the class GUID

Specifying the GUID works fine as long as you do not use the **DEVICE_NOTIFY_ALL_INTERFACE_CLASSES** flag.


pavlosalpha
9/4/2012

Device Class GUIDs

Anyone thinking that they could register for events for a particular device by finding the device in Device Manager and looking up its Device Class GUID for use with the DEV_BROADCAST_DEVICEINTERFACE.dbcc_classguid field in this function would be mistaken.


molesmoke
9/10/2010