

Re: CE6.0 Exposing Driver Interfaces to an Application

Source:

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.platbuilder/2007-01/msg00917.html>

- *From:* "Dean Ramsier" <ramsiernospam@xxxxxxxxxx>
 - *Date:* Tue, 23 Jan 2007 08:54:10 -0500
-

You're confusing a couple of different things. You're correct, the least complicated mechanism is to create a simple stream driver and expose your functionality via IOCTLs. However, the interface provided by oalioctl is for *kernel* ioctls, not device driver ioctls. User mode components can't call kernel ioctls unless they are exposed via oalioctl. User mode components can call device driver ioctls though.

Unless your functionality needs to live in the OAL (unlikely) then you just need to create a simple device driver. You will open the driver with CreateFile, and access the ioctl using DeviceIoControl.

—
Dean Ramsier – eMVP
BSQUARE Corporation

"jld" <jld@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
<news:A2C0F00D-BF10-49E7-8D07-84CD573CE947@xxxxxxxxxxxxxxxxxxxx>

Thank you for your answer.

It is my understanding of this issue that the least complicated mechanism for transferring data to a custom "driver" is to use the stream driver interface and use IOCTLs to implement generic APIs.

In CE 6.0 these IOCTLs must then be added to public\common\oak\oalioctl in order for them to be accessed from the application.

Reference: OAL Ioctl Codes

http://blogs.msdn.com/ce_base/archive/2006/11/14/application-compatibility-in-windows-ce-6-0.aspx

The custom IOCTLs are accessed from the application using DeviceIoControl
(?)

"Valter Minute" wrote:

Re: CE6.0 Exposing Driver Interfaces to an Application

=?Utf-8?B?amxk?=<jld@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in
news:86F03121-64E4-4BD6-81B5-E420A3272675@xxxxxxxxxxxxxx:

[...]

I found some documentation for creating a stream driver. Are all
drivers forced to be stream drivers and use IOCTLs ?

The drivers that don't support a native peripheral interface (ex: USB host, display adapter etc.) should be implemented as streaming I/O drivers and the functions that don't fit the Read/Write model should be implemented as IOCTLs.

You can write a simple wrapper library that hides the IOCTLs from the application programmer. This library could be an advantage also for the driver developer since you can put more controls inside it (ex: allocate buffers and don't free them while a call is pending etc.) making your solution more reliable (of course this doesn't mean that you should put validations and checks only inside the .lib!).

--

Valter Minute

(the reply address of this message is invalid)

(l'indirizzo di reply di questo messaggio non è valido)