How can I detect that the system is no longer showing a UAC prompt?

devblogs.microsoft.com/oldnewthing/20200429-00

April 29, 2020

Raymond Chen

A customer wanted to know how to detect when the system is no longer showing a UAC prompt.

They explained that their program uses Direct3D9. When the UAC dialog is shown, a desktop switch occurs, and their application loses the graphics device. The attempt to reacquire the graphics device fails for as long as the UAC dialog is on screen. They want to know when the UAC dialog is dismissed so they know when they can attempt to reacquire the graphics device. Otherwise, they are forced to poll.

Before checking with the product team, the customer liaison suggested that the customer look for a process called `consent.exe` and wait for it to exit.

Okay, first of all, that's a bad idea. The `consent.exe` program is an implementation detail. Nothing in the platform requires that the UAC feature be implemented by a program called `consent.exe`, nor does anything bind that the disappearance of such a program to the ability to create a Direct3D9 device.

The customer is too focused on the UAC dialog and missing the big picture. Their application can lose access to graphics hardware for reasons other than the UAC dialog. The user might lock the workstation. The user might hit `Ctrl`+`Alt`+`Del`. The screen saver could start.

If you are specifically interested in when the user has switched to your desktop, you can get the desktop associated with the current thread and see if it is the one that is receiving input.

```
BOOL IsCurrentThreadOnInputDesktop()
{
  BOOL result;
  HDESK desktop = GetThreadDesktop(GetCurrentThreadId());
  return desktop &&
         GetUserObjectInformation(desktop, UOI_IO, &result,
                                  sizeof(result), nullptr) &&
         result;
}
```

To know when the input desktop has changed, you can register an accessibility event hook and listen for the `EVENT_SYSTEM_DESKTOPSWITCH` event. When you receive that event, check again.

But really, you don't care about whether you're the input desktop. You want to know whether you can try to acquire graphics resources.

So do that.

Wait for a `WM_PAINT` message and attempt your acquisition then. When your window isn't visible, the system stops sending `WM_PAINT` messages, so the fact that they have returned is an indicator that now might be a good time to try to acquire your graphics resources.

This does have a thundering herd problem: When the user switches desktops, all the programs on the new desktop will try to grab graphics resources all at once. The user may find themselves with a momentarily unresponsive system as all the programs wake up and try to do stuff all at the same time. Better would be to wait until the user activates your application's window, if possible.

A member of the graphics team suggested that the customer could switch to Direct3D 9Ex, which has improvements to the way it deals with desktop switching.

Bonus chatter: The UAC dialog can be configured to show on the same desktop, rather than a custom desktop. But the customer doesn't care about that case. If we had answered the customer's question as original stated, they would have been handling that case incorrectly.

Raymond Chen

Follow