Don't just grab the foreground window and host UI on it

devblogs.microsoft.com/oldnewthing/20070727-00

July 27, 2007

Raymond Chen

I was asked to look at an application compatibility bug in a program from a major vendor. But that's actually irrelevant; what I'm writing about today has nothing to do with application compatibility. That's just what drew my attention to the program in the first place. At some point during the install, the setup program encountered an error and wanted to display an error message. When it called `DialogBox` to display the error message, it didn't pass the setup program's main window as the `hwndParent`. Instead it passed `GetForegroundWindow()`. They chose the wrong owner for modal UI. (I've also seen people try `GetTopWindow(0)`.) It so happened that the foreground window was Task Manager, since I had switched to Task Manager to look at various statistics of their installer as it ran. I hope you can see where this is going. They passed Task Manager as their modal owner, and since modal dialog boxes disable the owner, they ended up disabling Task Manager. (Meanwhile, their main setup program remained enabled, so I could have clicked on the Cancel button if I wanted to, which would have led to the "stack without support" problem.) Now I can't terminate their broken setup program from Task Manager since they **inadvertently disabled Task Manager**. But why did the programmers choose to use the foreground window anyway? One possibility is the programmer's version of the politician's fallacy.

- We must pass a window.
- The foreground window is a window.
- Therefore, we must pass it.

Another possibility is that they did this on purpose in order to ensure that their error message steals focus. Because their program is the most important program in the history of mankind. Unfortunately, I see this a lot. People who think their program is so important that they will abuse the rest of the system in order to get what they want instead of just waiting their turn. Of course, these people also fail to realize that setting a window as the owner for UI creates its own problems. As already noted, you disabled a random program. What's more, you've now attached the two input queues and tied your fates together. If the program that owns the foreground window stops responding to messages, then your program will also stop responding to messages.

But primarily it's just rudeness. You took somebody else's window and started acting as if you owned the place. It's like looking up somebody's address in the phone book and using it as your own. That's not your house, and that's not your window.

Raymond Chen

Follow