On the importance of making sure WaitForInputIdle doesn't think you're idle, episode 2

devblogs.microsoft.com/oldnewthing/20160812-00

August 12, 2016

Raymond Chen

Continuing our DDE micro-series, we'll look at another customer who was having trouble getting the shell to recognize their DDE server.

> We have a program that supports DDE for legacy reasons. More specifically, we have two versions of that program, and we support the user installing both of them side by side. To mediate this uneasy coexistence, we intend to have a "selector" program that registers for all the file extensions, and the user configures the selector so that the requests are directed at the specific version of the program the user chooses.
>
> We threw together a quick prototype of the selector, which simply looks up the user's preference, and then forwards its command line to the appropriate version. For example, we register the *open* verb as `ddeexec = selector.exe`, and the selector program decides that (say) the user wants files to open in v1, so the selector runs `"C:\Program Files\Contoso\v1\contoso.exe"`.
>
> What we found is that we get *There was a problem sending the command to the program*. What are we doing wrong?

Recall that after the shell launches the registered command, the shell calls `WaitForInputIdle`, and then when that call returns, the shell goes looking for the DDE server.

The first catch is that `WaitForInputIdle` requires a GUI program. This makes sense because console programs don't pump messages, so any such wait would be infinite. What's happening is that the shell launches the selector, and then the shell calls `WaitForInputIdle`, which returns (with an error), and then the shell goes looking for the DDE server. But the DDE server isn't ready yet.

The selector needs to be a GUI program, and it needs to perform a `WaitForInputIdle` on the final program, so that it doesn't go input idle until the actual server goes input idle. (In a sense, the selector is proxying input idle-ness.)

The customer tried a quick prototype with a WPF program, but it still didn't work. I don't know for sure, but I suspect that something in the WPF framework is pumping messages (perhaps due to a cross-thread COM operation) or creating a background thread that goes input-idle, which causes the entire process to go input-idle before the business logic can launch the true DDE server.

The customer tried another prototype with a pure Win32 program that launched the true DDE server, and then used `WaitForInputIdle` to wait for the true DDE server to go idle, and then exited.

And this worked, sort of.

What the customer found is that they needed to add a `Sleep(1000)` between launching the true DDE server and calling `WaitForInputIdle`. If they called `WaitForInputIdle` immediately after the `CreateProcess`, then the shell error occurred.

This stumbles across another fine detail of `WaitForInputIdle`: The process must be a GUI process. And even though the true DDE server is a GUI process, the selector is so fast that it calls `WaitForInputIdle` before the true DDE server can call into the window manager and create its message queue, which is what causes the program to be marked as a GUI program. When this happens, the `WaitForInputIdle` function returns `WAIT_FAILED`.

Therefore, the selector program should check whether `WaitForInputIdle</code returns WAIT_FAILED ; if so, it should sleep a little bit and try again. (And eventually give up.)`

Phew.

Please stop using DDE.

Raymond Chen

Follow