If an operation results in messages being sent, then naturally the target window must be processing messages for the operation to complete

devblogs.microsoft.com/oldnewthing/20110221-00

February 21, 2011

Raymond Chen

If an operation includes as part of its processing sending messages, then naturally the target window for those messages must be processing messages (or more precisely, the thread which owns the target window must be processing messages) in order for the operation to complete. Why? Because processing messages is the only way a window can receive messages! It's sort of tautological yet not obvious to everyone. Generally you run into this problem when you try to manipulate a window from a thread different from the one which created the window. Since windows have thread affinity, operations from off-thread typically need to get moved *onto* the thread which owns the window because that's where the window really "lives". The window manager will often try to see how much it can do without marshalling to the thread which owns the window, but when message traffic is involved, you are pretty much stuck. Messages are delivered to a window on the thread to which the window belongs, and there's no way around that. There are subtle ways in which a function called off-thread can result in message traffic. Generally speaking, you should just assume that any operation on a window may generate messages: Even if they don't do so today, they may do so in the future. For example, changing a window's style did not generate message traffic in early versions of Windows, but in Windows 95, it began generating `WM_STYLECHANGING` and `WM_STYLECHANGED` messages. This isn't called out explicitly in the documentation for `SetWindowLong` but it's implied by the documentation for `WM_STYLECHANGING` and `WM_STYLECHANGED` . Why isn't there an explicit callout in the documentation for `SetWindowLong` ? At the time the `SetWindowLong` documentation was originally written, the `WM_STYLECHANGING` and `WM_STYLECHANGED` messages did not exist. Therefore the documentation was complete at the time of writing. Circumstances changed elsewhere in the system that had secondary effects on `SetWindowLong` , but nobody bothered to update the documentation, probably because it didn't even occur to anybody that these effects existed. And then these secondary effects lead to tertiary effects: `SetScrollInfo` may change the window style to add or remove the `WS_HSCROLL` or `WS_VSCROLL` style, which in turn results in a call to `SetWindowLong` which in turn results in sending the `WM_STYLECHANGING` and `WM_STYLECHANGED` messages. Next come

quaternary effects on functions like `FlatSB_SetScrollInfo`, since they call `SetScrollInfo` as part of their functioning. And so on, and so on. Just tracking down the full ripple effect of those two new messages is probably impossible.

But the root cause of all these ripple effects is operating on a window (particularly *modifying* a window) from a thread different from the thread that owns the window. Avoid that, and you'll avoid the whole issue of which operations generate messages and which manage to sneak by without needing to send any messages (at least not yet).

Raymond Chen

Follow