How can I wait more than 30 seconds for a delay-rendered clipboard format to become rendered?

devblogs.microsoft.com/oldnewthing/20220609-00

June 9, 2022

Raymond Chen

Last time, we saw that the system will wait up to 30 seconds for a clipboard owner to produce delay-rendered data. But what if you want to extend this timeout?

The timeout itself is embedded in the clipboard manager and is not configurable. Thirty seconds is the longest the clipboard manager will wait for the clipboard data to be rendered.

But that doesn't mean that's the longest *you* will wait.

Recall that the `WM_RENDERFORMAT` message handler will call `SetClipboardData` when it's done, hoping that the clipboard is still open by the original code that requested the data.

So you just need to keep the clipboard open long enough for the clipboard owner to finish rendering the data format and calling `SetClipboardData`.

But how do you know that the clipboard owner has finished rendering the data format?

One idea is to poll by calling `GetClipboardSequenceNumber` until the sequence number changes, indicating that somebody (presumably the clipboard owner) called `SetClipboardData`. This is unsatisfying because (1) it's polling, and (2) if an error occurs which prevents the clipboard owner from producing the data, you're just going to be polling forever, unless you create your own timeout.

I have another idea: Call `GetClipboardOwner` to obtain the handle of the clipboard owner. This is the window that is busy trying to generate the clipboard data. Send this window a harmless message like `WM_NULL`. The purpose is not to get a meaningful reply to the message; the purpose is to know "Hey, this window is responding to messages again!"

The theory here is that if the window is not responding to messages, then that means that it is still busy producing the data to be put onto the clipboard. Once it starts responding, then the delay-rendering has completed.

This theory holds up if the work is being done by the thread that owns the clipboard owner window. However, if the clipboard owner window is delegating the work to another thread, then this mechanism will report that the clipboard owner processed the `WM_NULL` message, because threads can process inbound sent messages while waiting for an outbound sent message to return.

But hey, it's better than nothing.

```
if (OpenClipboard(hwnd)) {
    if (IsClipboardFormatAvailable(format)) {
        auto data = GetClipboardData(format);
        if (!data) {
            auto owner = GetClipboardOwner();
            SendMessage(owner, WM_NULL, 0, 0);
            data = GetClipboardData(format);
        }
        if (data) {
            // yay, we have clipboard data
        }
    }
    CloseClipboard();
}
```

After opening the clipboard, we check whether our desired format is available. If so, we ask for it. If that request failed, then we send the clipboard owner a `WM_NULL` message to wait for it to finish generating the format, and then try a second time. If that second try also fails, then we just give up.

Of course, instead of sending the `WM_NULL` message with `SendMessage`, you can use `SendMessageTimeout` to apply your own custom timeout. If the `SendMessageTimeout` of `WM_NULL` fails, then you can probably skip calling `GetClipboardData` a second time. The thread is probably still busy processing the original request. There's an off chance that the thread completed the original request but became hung on something else in the meantime. I don't consider this off chance worth the effort to work around. The whole thing is just a workaround anyway.

It turns out that if you go back to the customer's original problem, none of these workarounds are needed anyway. We'll look at a better solution next time. </P.

Raymond Chen

Follow