Why does an attempt to create a SysLink control in my plug-in sometimes fail?

devblogs.microsoft.com/oldnewthing/20141107-00

November 7, 2014

Raymond Chen

A customer had written a plug-in for some application, and they found that their plug-in was unable to create a SysLink control via the `CreateWindowExW` function. The same code in a standalone application works fine, but when the code is placed in their plug-in, the code fails. Debugging showed that the call to `InitCommonControlsEx` succeeded, but the `Create-WindowExW` call failed with “Cannot find window class.” The customer is another victim of not keeping their eye on the activation context. They attached a manifest to their DLL so that the call to `InitCommonControlsEx` maps to the version of the common controls library that supports the SysLink control. But they did nothing to ensure that that context was active at the time they called `CreateWindowExW`. The customer’s plug-in clearly falls into the case *Adding Visual Style Support to an Extension, Plug-in, MMC Snap-in or a DLL That Is Brought into a Process*. but they failed to follow the instructions provided therein (which boil down to “use isolation awareness”). From the symptoms, it appears that the host application for their plug-in does not activate a version-6 common controls manifest at the time it calls into the plug-in, which means that your attempt to create version-6 common controls will fail.

On the other hand, the standalone application probably uses the technique given in *Using ComCtl32.dll Version 6 in an Application That Uses Only Standard Extensions*, which activates the version-6 common controls when the process starts and *leaves it active* for the duration of the process.

Raymond Chen

Follow