Psychic debugging: Why does my app sometimes fail to change the display settings?

devblogs.microsoft.com/oldnewthing/20200207-00

February 7, 2020

Raymond Chen

A customer was having trouble with their Windows 7 Embedded system. (You can tell how old this story is because Windows 7 Embedded was still a thing.)

Their app runs as part of the Startup group, and they found that if a field technician replaced the system's screen, the resolution would sometimes reset to the default, because y'know, Plug and Play is like that sometimes. To fix this, they had their app call `ChangeDisplaySettings` when it starts up, to ensure that the screen resolution was what they wanted. This works great… most of the time. Sometimes, the call to `ChangeDisplaySettings` fails. The problem was intermittent, so debugging it was difficult.

While searching for possible reasons why they couldn't change the screen resolution, they discovered the `JOBOBJECT_BASIC_UI_RESTRICTIONS` structure, and in particular the `JOB_OBJECT_UILIMIT_DISPLAYSETTINGS` flag which prevents processes in the job from calling `ChangeDisplaySettings`. They also remembered that programs in the Startup group ran inside a job object, and they put two and two together and concluded that Explorer was running their app in a job object that had blocked screen resolution changes.

The customer had a few questions: Was their theory correct? If so, is there a way to escape the job object, or is there a policy or setting that can disable the job object outright?

I'm going to answer the questions in the wrong order.

Yes, there is a way to escape the job object, but it's not obvious. Explorer does not enable `JOB_OBJECT_LIMIT_BREAKAWAY_OKAY` on the job. Was this due to oversight, or was it an intentional limit, but it really doesn't matter because either way, you have to deal with it.

Instead, you'll have to use sneaky tricks to break away. I called out one of them in the original article: Use a logon-triggered scheduled task. Another one is mentioned in the documentation for job objects: Child processes created with the WMI method `Win32.Process.Create` are not associated with the job.

Okay, next question: Is there a setting to disable the job object? Yes there is. As I noted in the original article, you can set the `Delay_ Sec` to zero to disable running Startup apps in a box.

But the customer said that they're running Windows 7, and the default value for `Delay_ Sec` is already zero on Windows 7, so the only way they could end up in a job object is if somebody changed the default setting. (Given that this is an embedded system, that's possible, though I don't see why anybody would want to enable it. Usually, embedded systems *disable* stuff.)

Is their theory correct? Not really. While it's true that Explorer can be asked to run Startup apps in a job object, it does not apply any UI restrictions on those apps. Apps in the Startup group are not blocked by the job object from changing the display settings.

I have a different theory about why their app cannot change the display settings: In order to change the display settings, the app must be running on the current input desktop. My theory is that under sporadic conditions, the app either starts up too soon (before the Welcome screen has switched to the application desktop), or starts up too late (at which point the screen saver may have already started).

Unfortunately, we never heard back from the customer, so I will never know whether my psychic powers were effective.

Raymond Chen

Follow