Calling ShutdownBlockReasonCreate from my service doesn’t stop the user from shutting down

devblogs.microsoft.com/oldnewthing/20151002-00

October 2, 2015

Raymond Chen

A customer had a service application that needs to perform a sequence of operations that cannot be interrupted by a system shutdown.

> The service starts performing those operations when it starts up, but we found that group policy or other shenanigans can trigger other actions that ultimately lead to a system reboot. We would like to delay the reboot until our service completes its critical sequence of operations.
>
> We created a simple test application that calls `ShutdownBlockReasonCreate`, and it works. If we try to reboot the computer, the shutdown dialog appears indicating that it is waiting for the test application to complete.
>
> We ported this code into the service, and even though the call succeeds, there is no message shown to the user at reboot, and the reboot proceeds anyway.
>
> Is `ShutdownBlockReasonCreate` expected to work when called from a service? If so, then what we are doing wrong? If not, what alternative designs are available?

Shutdown blocks apply only to the session in which they are created. This is sort-of implied by the fact that the `ShutdownBlockReasonCreate` function is exported from `user32.dll`, and it takes a window handle parameter, and window handles are valid only in the desktop from which they were created. Since the interactive session is session 1 and services are in session 0, the block request has no effect. (This is just another application of the principle *Does this work in a service? The default answer is NO*.)

Note also that the shutdown block is not a hard block. The user could click “Shut down anyway”. Besides, the system assumes “Shut down anyway” if the user takes no action within some number of seconds.

Note also that if the system is shut down remotely, there is nobody present to answer the question, and it’s not clear where to display the message anyway.

The correct way to delay shutdown indefinitely from a service is to set the `SERVICE_ACCEPT_PRESHUTDOWN` flag in its service status. This registers the service for preshutdown notifications, at which point it can finish up your critical actions before releasing shutdown to continue.

Note that long shutdown times is one of the major points of end-user frustration, so you should do what you can to wrap things up quickly. Otherwise, you're going to be another data point in the "Windows sucks" database.

Raymond Chen

Follow