Mom, a modal loop ate my service messages! (http://www.sandon.it/node/100)

Posted on: Tue, 08/04/2015 - 22:44 **By:** ldsandon

A colleague of mine was using the "modernized" service implementation I wrote about (http://www.sandon.it/?q=node/97) some time ago. His services needs to be notified when a user logs on/off or locks/unlocks the machine. Everything was fine for a while, until he started to complain it worked for a while, but then, "randomly", notifications weren't triggered any longer. Debugging, he found the function handling the thread message queue received a WM_TIMER message (and no timer was used!), and then was no longer invoked, while the service control function correctly received service notifications, and posted it to the message queue. After carefully checking my service implementation code, and adding some error checks that weren't available in the implementation (i.e. GetMessage() although returning a BOOL value, can return more than two values...), nothing wrong could be found.

Inspecting the call stack before and after the messages got misteriously lost, we found a call to MessageBox()! What was triggering it? The developer remembered he was using a trial version of an updated version of a library he needed (because of some bugs in the old version), while waiting for procurement to buy the upgrade. It was the library nag screen to generate the WM_TIMER message, then calling MessageBox() and entering a modal loop. The library objects were initialized in the OnStart() event, and it is invoked in the message handling thread context - thereby the dialog got displayed by the same thread.

As Raymond Chen explains (http://blogs.msdn.com/b/oldnewthing/archive/2005/04/26/412116.aspx) in his excellent "The Old New Thing" blog, a message generated by a call to PostThreadMessage() - called by the actual implementation - don't go anywhere when passed to DispatchMessage(), because there is no window handle.

Ok, since Vista no service must ever call MessageBox(), but this can happen whenever a thread receiving messages through PostThreadMessages() creates a window and the enters a modal loop for any reason - and even COM calls could create a modal message loop. Thereby, I'm planning to remove the Windows message queue and replace it with a queue for service notifications local to the class implementation. In my humble opinion, I believe TService implementation uses a Windows message queue because when it was written first in Delphi 4, it was the cheapest solution to implement an asynchronous queue. Time to "modernize" this too.

- Log in (http://www.sandon.it/user/login?destination=/comment/reply/node/100/comment_node_blog%23comment-form) to post comments