Does this operation work when impersonating? The default answer is NO

devblogs.microsoft.com/oldnewthing/20110928-00

September 28, 2011

Raymond Chen

I'll often see a customer ask for assistance with a scenario like this: "We're having trouble doing X. We're doing X1, X2, and X3, but it looks like we're getting the wrong answer back."

The next step in the conversation goes something like "There must be something else going on, because X1, X2 and X3 is the correct way of doing X. To demonstrate, I've written the following sample program that illustrates doing X by the X1, X2, X3 technique. When I run it, I get the correct answer. What do you get when you run it?"

"When we run your program we get the correct answer, but it doesn't work when we do it from our program." And then, as if by afterthought, "Could the problem be that we're impersonating?"

Ohhhhhh, you're *impersonating*. Thanks for not mentioning that.

By default, nothing works when impersonating. Impersonation requires end-to-end awareness. A function might create a worker thread—the worker thread runs with the identity of the process. A function might use a function like `QueueUsereWorkItem`—by default, the work item runs with the identity of the process. (You have to pass `WT_TRANSFER_IMPERSONATION` if you want the work item to respect impersonation.) A function might send a message to another window—that window will do its work under its own security token, not the token of the sender. A function might invoke a method on a remote COM object—that object will run under its own security token, not the token of the invoker. (COM requires you to call `CoSetProxyBlanket` to enable impersonation transfer during marshaling, and the server needs to call `CoImpersonateClient`. For some reason, this is called cloaking.) The registry keys `HKEY_CURRENT_USER` and `HKEY_CLASSES_ROOT` don't work when you're impersonating. (You have to use `RegOpenCurrentUser` or `RegOpenUserClassesRoot`.) Functions like `SHGetKnownFolderPath` have a token parameter which is used when impersonating; if you pass `NULL`, then it assumes you aren't impersonating.

The requirements go beyond just code that runs during the execution of the function in question. If you have a function which caches information across calls, the cache needs to be made impersonation-aware so that a value calculated when called while impersonating user X isn't mistakenly used while impersonating user Y.

In order for impersonation to work, every function all the way down the chain needs to be impersonation-safe. Sure, you might be careful to call `QueueUserWorkItem` with the `WT_TRANSFER_IMPERSONATION` flag, and your work item is careful to call `SetProxyBlanket` on its COM objects, and your COM server is careful to call `CoImpersonateClient` when servicing the call, but if your COM server then calls a helper object which calls `SHGetKnownFolderPath` and passes `NULL` for the impersonation token, then all your careful work has been for naught.

This is another special case of *When you create an object with constraints, you have to make sure everybody who uses the object understands those constraints*.

The Programming Golden Rule can be applied here as well: *When you write your own code, do you do this?* Since most people who write code do not think about impersonation (indeed, the operating system even encourages not-impersonation-safe coding when it provides conveniences like `HKEY_CURRENT_USER`) the default answer to "Does this work when I'm impersonating" is "No."

Raymond Chen

Follow