Why does CoCreateInstance work even though my thread never called CoInitialize? The curse of the implicit MTA

devblogs.microsoft.com/oldnewthing/20130419-00

April 19, 2013

Raymond Chen

While developing tests, a developer observed erratic behavior with respect to `CoCreateInstance`:

> In my test, I call `CoCreateInstance` and it fails with `CO_E_NOTINITIALIZED`. Fair enough, because my test forgot to call `CoInitialize`.
>
> But then I went and checked the production code: In response to a client request, the production code creates a brand new thread to service the request. The brand new thread does not call `CoInitialize`, yet its call to `CoCreateInstance` *succeeds*. How is that possible? I would expect the production code to also get a `CO_E_NOTINITIALIZED` error.

I was able to debug this psychically, but only because I knew about the *implicit MTA*.

The *implicit MTA* is not something I can find very much documentation on, except in the documentation for the `APTTYPEQUALIFIER` enumeration, where it mentions:

> [The **APTTYPEQUALIFIER_IMPLICIT_MTA**] qualifier is only valid when the *pAptType* parameter of the **CoGetApartmentType** function specifies APTTYPE_MTA on return. A thread has an implicit MTA apartment type if it does not initialize the COM apartment itself, and if another thread has already initialized the MTA in the process. This qualifier informs the API caller that the MTA of the thread is implicitly inherited from other threads and is not initialized directly.

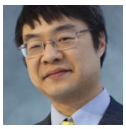
Did you get that? If any thread in the process calls `CoInitialize[Ex]` with the `COINIT_MULTITHREADED` flag, then that not only initializes the current thread as a member of the multi-threaded apartment, but it also says, "Any thread which has never called `CoInitialize[Ex]` is also part of the multi-threaded apartment."

Further investigation revealed that yes, some other thread in the process called `CoInitializeEx(0, COINIT_MULTITHREADED)`, which means that the thread which forgot to call `CoInitialize` was implicitly (and probably unwittingly) placed in the MTA.

The danger of this implicit MTA, of course, is that since you didn't know you were getting it, you also don't know if you're going to lose it. If that other thread which called `CoInitializeEx(0, COINIT_MULTITHREADED)` finally gets around to calling `CoUninitialize`, then it will tear down the MTA, and your thread will have the MTA rug ripped out from under it.

Moral of the story: If you want the MTA, make sure you ask for it explicitly. And if you forget, you may end up in the implicit MTA, whether you wanted it or not. (Therefore, conversely, if you *don't* want the MTA, make sure to deny it explicitly!)

Exercise: Use your psychic debugging skills to diagnose the following problem. "When my code calls `GetOpenFileName`, it behaves erratically. I saw a Knowledge Base article that says that this can happen if I initialize my thread in the multi-threaded apartment, but my thread does not do that."



Raymond Chen

Follow