How do I protect myself against a COM call that can hang? I'm already running the server out-of-process.

devblogs.microsoft.com/oldnewthing/20210122-00

January 22, 2021

Raymond Chen

Say you invoke a cross-process COM method, and then you decide to cancel it, say, because it's taking too long.

What you definitely *don't* want to do is call `TerminateThread`. That way lies madness.

What you can do is enable call cancellation by calling `CoEnableCallCancellation`, then make the potentially-hanging cross-process COM method call, and then after the call returns, call `CoDisableCallCancellation` to return things to the way they were.

Meanwhile, you can set a timer, and if the timer fires, the handler can call `CoCancelCall` with the thread ID of the thread that is making the potentially-hanging cross-process COM method call. If the potentially-hanging cross-process COM method call returns, then cancel the timer, since you don't need it any more.

But wait, the documentation for the `ICancelMethodCalls::Cancel` method says, "The behavior of the cancel object on receiving a cancel request is entirely at the discretion of the implementer." What if the server never calls `CoTestCancel`, or it pointedly ignores cancel requests? Is this whole exercise pointless?

While it's true that the `Cancel` method could do whatever the implementation wants, it's also the case that `CoGetCancelObject` will return a standard cancellation object for a cross-process call. And the system provides that implementation.

The system implementation issues a cancel request to the remote process and waits up to the specified timeout for the remote process to complete the call. Issuing the cancel request is what makes `CoTestCancel` report that a cancellation was requested, and if you're lucky, the remote process will abandon whatever it was doing and return back to you quickly.

If you're not lucky, the remote process will ignore the cancellation request, and after the timeout period elapses, the `CoCancelCall` function will force the original COM method call to return some cancellation error code, I forget exactly which. (You can pass a time of zero to issue the cancel request and not bother waiting for any sort of acknowledgement.)

If the remote server never calls `CoTestCancel`, that's fine. The server just keeps on going, unaware that its caller has given up waiting for the result.

Supplementary reading: Ready… cancel… wait for it!

Bonus chatter: There is a race condition if the timer fires before your original thread can manage to initiate the call at all. In that case, you'll try to cancel a nonexistent operation. You probably want to wait a little while and retry the cancellation, in the hope that the main thread has finally gotten around to issuing the call that you are trying to cancel.

What I do is schedule a recurring timer, and each time the timer fires, attempt to cancel. Eventually, one of them will succeed, and then the main thread can cancel the periodic timer. Even after the successful cancellation, I keep cancelling, because the main thread might be performing a series of operations, and I need to cancel all of them.

Raymond Chen

Follow