What does it mean when a call fails with 0x8000001F = RO_E_BLOCKED_CROSS_ASTA_CALL?

devblogs.microsoft.com/oldnewthing/20210225-00

February 25, 2021

Raymond Chen

Last time, we learned a little bit about the Application STA (ASTA) and how it blocks re-entrancy in order to avoid certain categories of deadlocks. But sometimes it rejects a call outright instead of delaying it. Why does that happen?

When a call leaves an ASTA for another thread, that destination thread is still allowed to call back into the ASTA. A common pattern is for a method to accept one or more parameters which are themselves objects, and the method implementation naturally wants to access those objects. This is allowed even though it is technically a re-entrancy situation.

If the destination thread in turn calls a third thread, that third thread is also allowed to access the object that belongs to the initiating ASTA. COM keeps track of the chain of threads involved in the call, and if the chain of threads leads back to the ASTA, that's still okay.

However, what COM does not allow is a call chain that passes through *two* ASTAs, and then tries to re-enter one of those ASTAs. That type of re-entrancy is blocked due to deadlock risk. However, the call cannot be delayed until the ASTA returns to normal, since it is part of a call chain that involves the original ASTA, and that would result in a deadlock. Instead, COM rejects the call with the error `RO_E_BLOCKED_CROSS_ASTA_CALL`.

There are special rules inside COM to allow certain types of re-entrancy in violation of the above rules. For example, `QueryInterface` is always allowed. And if you are using the `IContextCallback::ContextCallback` method, you can specify whether you want your callback to be subject to ASTA re-entrancy rules.

Raymond Chen

Follow