Why does the OpenThread function behave differently when the target thread belongs to another process?

devblogs.microsoft.com/oldnewthing/20140822-00

August 22, 2014

Raymond Chen

A customer discovered strange behavior in the `OpenThread` function and wondered whether it was expected.

> We use the `OpenThread` function to obtain a thread handle with `THREAD_QUERY_LIMITED_INFORMATION`, passing in a valid thread ID. We later pass this handle to `GetExitCodeThread` to get the thread exit code. We have found that the function succeeds if the thread in question belongs to another process, provided the thread is still running (has not yet exited). On the other hand, if the thread belongs to our own process, then the call always succeeds regardless of whether the thread is running or not. Is this expected behavior? And can we assume that if `OpenThread` fails with `ERROR_INVALID_PARAMETER`, then it means that the target thread has already exited?

The `OpenThread` function fails if you pass it an invalid thread ID. Thread IDs go invalid when the corresponding thread object is destroyed, and thread objects are destroyed when the thread exits and there are no open handles to the thread. Once a thread object is destroyed, its thread ID becomes invalid and may be re-used by a future thread. Whether the thread belongs to the same process or a different process does not play a rôle in this determination. My guess is that the reason the call succeeds if the target thread belongs to the same process, even if the target thread has already exited, is something much more mundane: They have a thread handle leak in their application. The customer never wrote back after receiving this explanation, so we'll never know whether my guess was correct.

Bonus chatter: If you aren't sure whether you are passing a valid thread ID to `OpenThread`, then you most likely already have a bug. Since thread IDs can be reused, if you haven't taken other steps to ensure that the thread you want still exists, then it's possible that the thread you want has already exited, the corresponding thread object has been destroyed, and the thread ID has been reused by some other thread. Your `OpenThread` call will now *succeed*, but it will refer to some totally unrelated thread. Your program will most likely get very confused at this point.

Raymond Chen

Follow