

Passing the incorrect object type for a handle, how bad is it?

 devblogs.microsoft.com/oldnewthing/20141105-00

November 5, 2014



Raymond Chen

A customer asked a somewhat strange question: “We are calling `SetEvent` but passing the handle to a waitable timer. Application Verifier reports, “Incorrect object type for handle.” But the code works well. We want to know the risks of passing the wrong object type to `SetEvent`. Is the recommendation only to pass handles of type “Event” to `SetEvent`? Let’s answer those questions in reverse order. Yes, the recommendation is only to pass handles of type “Event” to `SetEvent`, just as the recommendation is only to pass handles of type “Semaphore” to `ReleaseSemaphore`, and more generally, only to pass valid parameters to functions. What is the risk of passing the wrong object type? You’re lucky that the kernel does object type validation before proceeding, so your error is caught during parameter validation and the function fails with the error `ERROR_INVALID_HANDLE` (or status code `STATUS_OBJECT_TYPE_MISMATCH`, if the function returns status codes instead of error codes). Of course, if you are encountering this problem only because you are using a handle after closing it (and then the handle got recycled as a timer handle), then you merely got lucky. Maybe tomorrow you won’t be so lucky, and the handle will get recycled as another unrelated event. Tomorrow, your `SetEvent` call will succeed and *set some other guy’s event*. This will probably cause that other guy to get really confused. “This event is set when the modulator has finished calibrating. But the event is getting signaled before the calibration is complete, so my code ends up using an uncalibrated modulator! I set a breakpoint on my `SetEvent` call, and it never fires, yet the event is set. Help me debug this. I’ve spent a week trying to figure out what’s wrong!”

As to the final remark, “But the code works well,” it’s not clear what the customer meant by that. What does “works well” mean in this context? Do they mean, “The event is successfully set even though it’s not an event”? How can you successfully perform an event operation on something that isn’t an event? Or perhaps they mean, “Our code seems to work okay in spite of this mistake.” The operative phrase there is “seems to”. It may seem to work well, but someday it won’t, and at the most inconvenient time.

Raymond Chen

Follow

