# Why does OpenProcess succeed even when I add three to the process ID?

devblogs.microsoft.com/oldnewthing/20080606-00

June 6, 2008

Raymond Chen

A customer noticed that if you add three to a process ID and pass it to the `OpenProcess` function, it still succeeds. Why is that? Well, first of all, I need to say up front that the behavior you're seeing is an artifact of the implementation and is not part of the contract. You're passing intentionally invalid parameters, what did you expect? The context of this question is "We're seeing this behavior and we can't explain it," not "We're using this trick and want confirmation that it's okay." Now, you actually know the answer to this already.

As we saw earlier, for convenience, the Windows NT kernel uses the handle manager to parcel out process and thread IDs, and the handle manager ignores the bottom two bits of handles. Therefore, adding three has no effect on the process-id-to-object mapping. This mechanism is peculiar to kernels based on Windows NT. Versions of Windows derived from the Windows 95 kernel have a different mechanism for mapping process IDs to processes, and that mechanism is unflinchingly rigid. If you add three, the `OpenProcess` function will reject your process ID as invalid. And I don't know how Windows CE handles it. Again, I wish to emphasize that the behavior you see in Windows NT-based kernels is just an implementation artifact which can change at any time. Who knows, maybe once they read this entry, the kernel folks will go in and change `OpenProcess` to be even more strict.

**Pre-emptive Yuhong Bao comment**: "Process IDs on Windows 95 are a pointer to an internal data structure XORed with a constant to obfuscate them."

Raymond Chen

**Follow**