Can CoCreateGuid ever return GUID_NULL?

devblogs.microsoft.com/oldnewthing/20140326-00

March 26, 2014

Raymond Chen

A customer asked whether the `CoCreateGuid` function can ever return `GUID_NULL`. Their code uses `GUID_NULL` for special purposes, and it would be bad if that was ever returned as the GUID for an object. "Can we assume that `CoCreateGuid` never returns `GUID_NULL`? Or should we test the return value against `GUID_NULL`, and if it is equal, then call `CoCreateGuid` and try again?" Some people started running `CoCreateGuid` a bunch of times and observing that it was spitting out type 4 GUIDs, which will always have a 4 in the version field. Then other people started wondering whether the use of Algorithm 4 was contractual (it isn't). Then still other people went back to read the RFCs which cover UUIDs to see whether those documents provided any guidance. And then I had to step in and stop the madness. It is very easy to show that any UUID generator which generates `GUID_NULL` has failed to meet the requirement that the generated UUID be unique in space and time: If it's equal to `GUID_NULL`, then it isn't unique! The uniqueness requirement is that the generated GUID be different from any other valid GUID. And if it generated `GUID_NULL`, then it wouldn't be different from `GUID_NULL`! (And `GUID_NULL` is a valid GUID, specifically identified in RFC4122 section 4.1.7.) If you're so worried about `CoCreateGuid` generating a duplicate `GUID_NULL`, why aren't you worried about `CoCreateGuid` generating a duplicate `IID_IUnknown` or `GUID_DEVCLASS_1394` or any of the other GUIDs that have already been generated in the past? In other words, no valid implementation of `CoCreateGuid` can generate `GUID_NULL` because the specification for the function says that it is not allowed to generate any GUID that has been seen before.

One of my colleagues cheekily remarked, "And even if it did generate `GUID_NULL` for some reason, uniqueness would require that it do so only once! (So you should try to force this bug to occur in test, and then you can be confident that it will never occur in production.)"

Raymond Chen

Follow