The oracle always tells the truth, even when it is wrong: COM method calls with a user-defined type as a return value

devblogs.microsoft.com/oldnewthing/20220113-00

January 13, 2022

Raymond Chen

COM method calls follow the `STDMETHODCALLTYPE` calling convention, which is defined to be `__stdcall`. And the COM ABI for the layout of the virtual method call table was carefully chosen so that it matched the code generation of the Microsoft Visual C++ compiler. That way, you could access COM objects from C++ as if they were regular C++ objects.

One of the rules for the `__stdcall` calling convention on x86 is that if the return value is a structure, then the caller passes a hidden first parameter that is the address of an uninitialized structure in which to place the result. However, if that structure is 8 bytes or smaller, then a *small structure* special case kicks in, and instead of using the hidden first parameter, the result is returned directly in the `edx:eax` register pair.

It turns out that the Microsoft Visual C++ compiler doesn't use the small structure special case for C++ methods. When I brought it up to them, the Visual Studio folks investigated this quirk and discovered through source code spelunking that it has been this way since the beginning: If the return value of a C++ method is a user-defined type, then it is always returned via a hidden parameter, which for method calls comes after the hidden `this` parameter and before any formal parameters.

The compiler team probably figured that C++ method calls can't be ABI method calls because the Win32 ABI isn't defined in terms of C++ method calls; it's defined in terms of flat C-style functions. It was safe to follow a nonstandard convention for C++ method calls because those methods are never exposed via the Win32 ABI.

Except when they are implementations of COM methods that are placed in the COM vtable, which is defined by the Win32 ABI.

Oops.

This deviation explains why attempting to use the C bindings for methods like `ID2D1HwndRenderTarget::GetPixelFormat` fails spectacularly.

Okay, so what can we do about it? The operating system has been implementing the COM methods with the wrong calling convention for decades. Existing applications have been calling the COM methods with the (same) wrong calling convention for just as long. If we went and "fixed" the compiler to be conformant, it would break the world.

This is a case where the compiler is the *de facto* ABI. The calling convention in practice for COM methods is that the small structure special case is not in effect. Whether that's what the calling convention *should be* is beside the issue. The code has spoken, and the code says that's how it works. Changing the calling convention at this point would be an ABI breaking change from the compiler's point of view, and those are really scary.

One consolation for all of this is that COM methods that return a structure are very rare. Nearly all COM methods return `HRESULT`. For one thing, lacking an `HRESULT` means that there is no way to report a marshalling error.

The behavior of the Microsoft Visual C++ compiler was acknowledged as the *de facto* COM vtable ABI, and the MIDL compiler was updated to generate new C bindings that are compatible with it. I believe the revised MIDL compiler is in the Windows Insider SDK, waiting to go out with the next full SDK release. You can activate these bindings by passing the `/cstruct_out` flag to the MIDL compiler.

Bonus sad chatter: The Direct2D header files are generated by a custom tool, not by the MIDL compiler. The Direct2D team regret this decision. They regret it so much that they tell me that they deprecated and then ultimately deleted the C-style headers from the SDK.

Raymond Chen

Follow