The Windows Runtime PassArray is a read-only array, even though it isn't declared const

devblogs.microsoft.com/oldnewthing/20210825-00

August 25, 2021

Raymond Chen

As I noted some time ago, the Windows Runtime PassArray pattern passes a read-only non-owning counted array which is nevertheless not declared as `const`.

Indeed, if you try to force the array type to be `const` in your IDL declaration:

```
HRESULT SetData([in] UINT32 dataSize, [in, size_is(dataSize)] const INT32* data);
```

The `const` is ignored, and the resulting metadata declares the parameter as non-`const`.

There are a few reasons for this, partly intentional, and partly a technicality.

The technicality is that the `const` attribute is lost because Windows Runtime methods are described by metadata that physically takes the form of an ECMA-335 assembly (though restricted to a very limited subset of full ECMA-335), and ECMA-335 does not have `const`. Therefore Windows Runtime metadata cannot have `const`.

Mind you, this is an unsatisfying explanation since it's semi-circular. Windows Runtime metadata doesn't have `const` because the designers chose a format that doesn't support `const`, and it's okay to have chosen a format that doesn't support `const` because Windows Runtime metadata doesn't use `const`.

But really, if they really wanted `const`, then they would have chosen some other file format that *does* support `const`.

The Windows Runtime does not have `const` because the concept cannot be expressed in most programming languages,[1] and the Windows Runtime intends to be language-independent. Limiting the feature set of the Windows Runtime type system makes it more likely that it can be consumed by a broad range of programming languages.

[1] Indeed, it's really only C, C++ and now Rust that have such a concept. The C++ projections do represent the array as `const`: It is a `const Platform::Array` in C++/CX, and it is a `winrt::array_view<T const>` in C++/WinRT. Similarly, the Rust projection represents

the array as an immutable reference.

Raymond Chen

Follow