Why does C++/WinRT say that first_interface is not a member of winrt::impl::interface_list<>?

devblogs.microsoft.com/oldnewthing/20220323-00

March 23, 2022

Raymond Chen

A customer was trying to use C++/WinRT to implement a classic COM interface, but was getting the error

```
error C2039: 'first_interface': is not a member of 'winrt::impl::interface_list<>'
see declaration of 'winrt::impl::interface_list<>'
see reference to class template instantiation
'winrt::impl::implements_default_interface<D,void>' being compiled
with
[
    D=Widget
]
see reference to function template instantiation 'auto winrt::make<Widget>()' being
compiled
```

from the following code:

```
struct Widget : winrt::implements<Widget, IPersist>
{
    Widget(int length) : m_length(length) { }

    // IPersist methods
    STDMETHODIMP GetClassID(CLSID* clsid) noexcept override
    { *clsid = CLSID_NULL; return S_OK; }
};
```

What is going on? What does the error mean?

The error message is saying, rather obviously, that an empty list of interfaces has no first interface. Which is true, but where is this empty list of interfaces coming from, and why do we care about it?

The list of interfaces is generated from the template arguments to the `winrt::implements` template type. C++/WinRT looks through the template type arguments (starting from the second one) and keeps the ones that look like interfaces. Those are the ones that C++/WinRT will respond to in its `QueryInterface` implementation.

In this case, there's only one candidate: `IPersist`. Why didn't C++/WinRT think this was an interface?

C++/WinRT by default considers something an interface if it derives from `winrt::Windows::Foundation::IUnknown`. Ah, but `IPersist` doesn't derive from `winrt::Windows::Foundation::IUnknown`. It derives from `::IUnknown`. That doesn't count.

Okay, but wait, I've seen people do this. So how come they can do it but I can't?

I noted that *by default* C++/WinRT requires interfaces to derive from `winrt::Windows::Foundation::IUnknown`. You can activate nondefault behavior by including the `unknwn.h` header file *before* including any C++/WinRT header files, in which case C++/WinRT will also recognize types which derives from classic COM `IUnknown` as interfaces.

Include order	Result

`#include <unknwn.h>`
`#include <winrt/base.h>`
	Classic COM support active in C++/WinRT

`#include <winrt/base.h>`
`#include <unknwn.h>`
	Classic COM support not active in C++/WinRT

A very common way to get `unknwn.h` is to include `windows.h`, since `windows.h` includes `unknwn.h` by default.

Note that all the code in your module must agree on whether or not classic COM support is active. If there is disagreement, then you have an ODR violation, and the resulting behavior is undefined.

Next time, we'll take a peek behind the curtain of C++/WinRT so we can understand how C++/WinRT detects which interfaces are implemented.

Bonus chatter: If you are using the Windows Implementation Library, then you should include `wil/cppwinrt.h` before including any other C++/WinRT headers. In addition to making sure that `unknwn.h` is included first, it also activates a bunch of other features in both WIL and C++/WinRT so that they understand each other's exceptions, for example.

Bonus bonus chatter: PR #1022 improves support for classic COM in two ways:

- For consuming classic COM interfaces, the ordering requirement between `unknwn.h` and C++/WinRT header files has been removed. You just have to include `unknwn.h` at some point, either before or after including C++/WinRT. And in practice, you will, because you need to include that header file in order to declare any classic COM interfaces in the first place!

- To implement classic COM interfaces, you still have to include `unknwn.h` before including any C++/WinRT header files, but now you get a helpful diagnostic if you forget: “To implement classic COM interfaces, you must #include <unknwn.h> before including C++/WinRT headers.”[1]

[1] It’s surprising how much work in writing a library consists of error message hacking.

Raymond Chen

Follow