The various ways of moving between C++/WinRT and classic COM

devblogs.microsoft.com/oldnewthing/20210830-00

August 30, 2021

Raymond Chen

Windows Runtime language projections create parallel universes with types that correspond to the ABI types, but which are expressed in a projection-specific way. For the C++ projections (C++/WinRT and C++/CX), these projected types go even further: Not only do they correspond to the ABI types, but their internal layouts are identical to the ABI types. This means that you can use reinrepret-casting to convert between them.

Suppose we have a Windows Runtime `Contoso.Widget` with its associated default interface `Contoso.IWidget`.

C++/WinRT and C++/CX set up parallel universes like this:

C++/WinRT	ABI	C++/(
`winrt::Contoso::Widget`	`ABI::Contoso::Widget*`	`Cont`
`winrt::Contoso::IWidget`	`ABI::Contoso::IWidget*`	`Cont`
`winrt::Windows::Foundation::IInspectable`	`::IInspectable*`	`Plat`
`winrt::Windows::Foundation::IUnknown`	`::IUnknown*`	
`winrt::hstring`	`::HSTRING`	`Plat`
`winrt::Contoso::SomeStruct`	`ABI::Contoso::SomeStruct`	`Cont`

`winrt::Contoso::SomeEnum`	`ABI::Contoso::SomeEnum`	`Cont`

All of the entities in a block have the same internal representation and are therefore interchangeable at the ABI. They are just wrapped in different types.

It's not too difficult to move up and down the columns, since you are staying inside the same family of types. But moving between columns is usually much uglier, since you're travelling between universes.

A case where you may find yourself needing to move between columns is when you want to use a classic COM interface exposed by a Windows Runtime object. Classic COM interfaces exist only in the ABI world; there is no corresponding version in C++/WinRT or C++/CX.

One way to move between the C++/WinRT and ABI columns is to use C++/WinRT ABI-interop functions.

For getting ABI pointers out of C++/WinRT reference types and strings:

Expression	`x`	`v` receives	Notes
`v = get_abi(x)`	Unchanged	Non-refcounted pointer	Lifetime controlled by `x`
`copy_to_abi(x, v)`	Unchanged	Refcounted pointer	New refcount added
`v = detach_abi(x)`	Emptied	Refcounted pointer	Ownership transfer

And for putting ABI pointers into C++/WinRT reference types and strings:

Expression	`x` 's old value	`v`	Notes
`*put_abi(x) = v`	Released	No longer owning	Ownership transfer
`copy_from_abi(x, v)`	Released	Still owning	Reference-counted copy
`attach_abi(x, v)`	Released	No longer owning	Ownership transfer

A customer had a C++/WinRT `Contoso.Widget` and wanted to get the raw ABI `::IUnknown*` from it, so that they could pass it to `CoSetProxyBlanket`. Here's one way to do it:

```
winrt::check_hresult(
    CoSetProxyBlanket(
        reinterpret_cast<::IUnknown*>(winrt::get_abi(thing)),
        RPC_C_AUTHN_DEFAULT,
        RPC_C_AUTHZ_DEFAULT,
        COLE_DEFAULT_PRINCIPAL,
        RPC_C_AUTHN_LEVEL_DEFAULT,
        RPC_C_IMP_LEVEL_IMPERSONATE,
        nullptr /*pAuthInfo*/,
        EOAC_NONE));
```

Another way to do it is to use `winrt::com_ptr`, which gives you a little bridge to the classic COM world, provided you include `<unknwn.h>` before including any C++/WinRT header files.

```
winrt::check_hresult(
    CoSetProxyBlanket(
        thing.as<::IUnknown>().get(),
        RPC_C_AUTHN_DEFAULT,
        RPC_C_AUTHZ_DEFAULT,
        COLE_DEFAULT_PRINCIPAL,
        RPC_C_AUTHN_LEVEL_DEFAULT,
        RPC_C_IMP_LEVEL_IMPERSONATE,
        nullptr /*pAuthInfo*/,
        EOAC_NONE));
```

If you're the sort of person who worries about such things, do note that `as()` performs a `QueryInterface` call, whereas `get_abi` is basically free because it just reaches in and gives you the embedded pointer.

Structures and enumerations are simpler to manage, since there is no reference count. You can just `reinterpret_cast` between the types:

```
winrt::Contoso::SomeStruct s1;
ABI::Contoso::SomeStruct s2 = reinterpret_cast<ABI::Contoso::SomeStruct&>(s1);

winrt::Contoso::SomeEnum e1;
ABI::Contoso::SomeEnum e2 = reinterpret_cast<ABI::Contoso::SomeEnum>(e1);
```

Bonus chatter: Really, you can just `reinterpret_cast<T&>` to move between columns, even for reference types, since the internal representations are the same.

```
winrt::Anything a1;
ABI::Anything a2;
::Anything a3;

a1 = reinterpret_cast<winrt::Anything&>(a2);
a1 = reinterpret_cast<winrt::Anything&>(a3);

a2 = reinterpret_cast<ABI::Anything&>(a1);
a2 = reinterpret_cast<ABI::Anything&>(a3);

a3 = reinterpret_cast<::Anything&>(a1);
a3 = reinterpret_cast<::Anything&>(a2);
```

Raymond Chen

Follow