How do I destroy an ABI pointer that I extracted from a C++/WinRT object?

devblogs.microsoft.com/oldnewthing/20250310-00

Raymond Chen March 10, 2025

A customer was working with C++/WinRT↔ABI interop. They had extracted the ABI pointer from a C++/WinRT object via `detach_abi`, and now they were finished with it and wanted to know how best to destroy it.

```
winrt::Contoso::Widget w = ⟦ something ⟧;
winrt::hstring s = ⟦ something ⟧;

void* w_abi = winrt::detach_abi(w);
void* s_abi = winrt::detach_abi(s);
// or
void* w_abi;
winrt::copy_to_abi(w, w_abi);
void* w_abi;
winrt::copy_to_abi(s, s_abi);
```

This code obtains a `Widget` and an `hstring` from somewhere, not important where. The first variation extracts the ABI pointers from those objects into `w_abi` and `s_abi` using the `detach_abi` function, which means that the ownership of the ABI pointer has been transferred out of the source object (leaving it null). The second variation uses `copy_to_abi`, which means that the ABI pointer has been copied out of the source object, increasing the reference count. Either way, so the `w_abi` and `s_abi` are responsible for cleaning up the pointer when they are no longer needed.

The customer didn't say why they switched to the ABI. Maybe they needed to pass the pointer to some other function that consumed it as an ABI pointer. Maybe they wanted to pass it as reference data to a callback, and the callback supports only raw pointers.

Anyway, these ABI raw pointers are owning pointers, so when when they were done with them, they have to clean them up. How do you clean up a raw ABI pointer?

The tautological answer is that you clean up ABI raw pointers using whatever ABI mechanism applies. For example, the `w_abi` is a raw pointer to the ABI `IWidget` interface, so you can do this:

```
auto abi_widget = reinterpret_cast<ABI::Contoso::IWidget*>(w_abi);
```

and then clean it up like any other ABI pointer to a COM object.

```
if (abi_widget != nullptr) abi_widget->Release();
```

Similarly, you clean up the ABI `HSTRING` by calling `WindowsDeleteString()`:

```
WindowsDeleteString(reinterpret_cast<HSTRING>(s_abi));
```

You might simplify this by using an existing ABI helper library like WRL or wil.

```
// WRL
Microsoft::WRL::ComPtr<ABI::Contoso::IWidget> widget;
widget.Attach(reinterpret_cast<ABI::Contoso::IWidget*>(w_abi));
// allow the widget to destruct

Microsoft::WRL::Wrappers::HString s;
s.Attach(reinterpret_cast<HSTRING>(s_abi));
// allow the HString to destruct
```

I mean, you presumably took ownership of the ABI pointer because you wanted to become the one responsible for its lifetime. And part of the “take responsibility for its lifetime” is knowing how to end the lifetime. Why ask for something if you don’t know what to do with it? It’s like demanding that somebody transfer ownership of their car to you, even though you don’t know how to take care of a car. “You should have thought of that before you demanded a car.”

If you don’t want to be responsible for its lifetime (for example, if you are just passing it to another function that does not take ownership), then maybe don’t extract it at all.

```
winrt::Contoso::Widget w = ⟦ something ⟧;
winrt::hstring s = ⟦ something ⟧;

void* w_abi = winrt::get_abi(w);
ABIFunction(reinterpret_cast<ABI::Contoso::IWidget*>(w_abi));

void* s_abi = winrt::get_abi(s);
ABIFunction(reinterpret_cast<HSTRING>(s_abi));
```

Bonus chatter: Perhaps the simplest way to do it is to simply tell C++/WinRT to take ownership back!

```
winrt::Widget{ w_abi, winrt::take_ownership_from_abi };
winrt::hstring{ s_abi, winrt::take_ownership_from_abi };
```

We take the raw ABI pointers and create C++/WinRT objects that take ownership from them, and then let C++/WinRT's destructor do the work.

Author

Raymond has been involved in the evolution of Windows for more than 30 years. In 2003, he began a Web site known as The Old New Thing which has grown in popularity far beyond his wildest imagination, a development which still gives him the heebie-jeebies. The Web site spawned a book, coincidentally also titled The Old New Thing (Addison Wesley 2007). He occasionally appears on the Windows Dev Docs Twitter account to tell stories which convey no useful information.