What will be placed in the output parameter if the function fails?

devblogs.microsoft.com/oldnewthing/20200417-00

April 17, 2020

Raymond Chen

A customer wanted to know what, if anything, can be said about the value in the output parameter of the `OpenProcessToken` function if the call fails.

According to the ground rules, the contents of output parameters are unspecified on failure. There is a special rule for COM which says that the contents of output parameters on failure must be valid values for their type. For example, if the output parameter is a COM interface pointer, then it must contain a valid value for a COM interface pointer on return, even if the call failed.[1]

However, as a concession to the widespread use of RAII types, the *de facto* rule is that if the output parameter is a resource that requires management (disposal, release, etc.), then the value on failure is either unchanged or set to a “no resource” value.

If you use an RAII type, then you are going to pass an empty[2] RAII object to the function because the function may put a valid object into the RAII object, and you don’t want to leak the old object. Therefore, if the function that fails leaves the RAII object empty or explicitly sets it to empty, the RAII object can be destructed without harm. On the other hand, if the function put garbage in the output parameter, then the RAII type is going to try to clean up a garbage resource, which is unlikely to end well.

In other words, for types that require cleanup, “empty stays empty” on failure. (On the other hand, if the original object was not empty, it’s not specified whether the result is empty or not.)

Now, if the output doesn’t require resource management, then this rule doesn’t apply. For example, if the output is a string buffer, the buffer could very well be filled with garbage on failure. You need to free the buffer, and a buffer of garbage can be freed just as easily as a buffer with a valid string in it.

Examples:

```
BOOL GetSomething(HSOMETHING* result);
```

Does the `HSOMETHING` need to be destroyed in a special way? If so, then on failure, `GetSomething` will either leave the `*result` unchanged, or set it to `nullptr`. If you are using an RAII type to hold the `HSOMETHING`, then you will have arranged for `*result` to contain `nullptr` before calling `GetSomething`, which means that on failure, `*result` will still be `nullptr`.

```
HRESULT GetWidgetName(HWIDGET widget, wchar_t buffer[], size_t size);
```

Since `wchar_t` does not require any special resource management, the `buffer` could be filled with garbage if `GetWidgetName` fails.

```
HRESULT GetWidgetName(HWIDGET widget, PWSTR* name);
```

This case is different. Let's say that the `name` needs to be freed with a function like `CoTaskMemFree`. In that case, the function cannot set `*name` to a garbage pointer, because that would cause `CoTaskMemFree` to crash. On failure, `*name` will be unchanged or set to `nullptr`.

Bonus chatter: WIL has a number of helper functions for manipulating tokens.

[1] For COM interface pointers, the value is typically `nullptr` on failure. Exceptions are called out in the function documentation if the result is anything else. For example, a function might return `E_PENDING` and put a provisional answer in the output pointer, with a complete answer provided when the operation completes.

[2] The word *empty* is a term of art which refers to the case where an object of an RAII type is not managing anything.

Raymond Chen

Follow