The subtleties of CreateStreamOnHGlobal, part 1: Introduction and basic usage

devblogs.microsoft.com/oldnewthing/20210928-00

September 28, 2021

Raymond Chen

The `CreateStreamOnHGlobal` function lets you create a COM `IStream` that uses an `HGLOBAL` memory block as its backing store. It takes two input parameters: an optional `HGLOBAL` that represents the memory block to use, and a flag called `fDeleteOnRelease = FALSE` that controls whether the stream object frees the `HGLOBAL` when it is destroyed. The initial contents and size of the stream correspond to the initial contents and size of the `HGLOBAL`; if you don't pass one, then then the function will create its own from scratch and start with an empty stream.

Sounds simple, right?

It stops being simple when you start exploring the dark corners.

But let's start in the brightly-lit area: The normal case where you pass `fDeleteOnRelease = TRUE`.

Passing `fDeleteOnRelease = TRUE` represents an ownership transfer. The stream assumes ownership of the `HGLOBAL` you pass in (assuming you passed one at all), and it is free to do with it whatever it likes. When the stream is destroyed, it frees the `HGLOBAL`, too.

The rule in this case is simple: Once you give an `HGLOBAL` to the `CreateStreamOnHGlobal` function, it's not yours any more. Correct usage would be something like this:

```
HGLOBAL hglobal = CreateInitialContents();

IStream* stream = NULL;
if (SUCCEEDED(CreateStreamOnHGlobal(
        hglobal, TRUE, &stream))) {
    hglobal = NULL;     // Not our HGLOBAL any more
}
```

If you're using a smart pointer library, you would perform what is commonly known in Windows as a `Detach` operation, but which the C++ standard library calls `release`, which is not the same as a COM Release, so don't get confused.

```
wil::unique_hglobal hglobal = CreateInitialContents();

wil::com_ptr<IStream> stream;
if (SUCCEEDED(CreateStreamOnHGlobal(
        hglobal.get(), TRUE, &stream))) {
    hglobal.release();     // Not our HGLOBAL any more
}
```

The stream object assumes ownership of the `HGLOBAL` and will treat it as its own personal property. If somebody writes to the stream, it will write to the `HGLOBAL`. If the write extends beyond the end of the `HGLOBAL`, or if somebody calls `IStream::SetSize` to change the size of the stream, then it will resize the `HGLOBAL` correspondingly. But that's okay, because you gave up that `HGLOBAL` back when you created the stream. Whatever the stream does with the `HGLOBAL` is no longer your business.

That's the easy case.

Next time, we'll start looking at the hard case where you pass `fDeleteOnRelease = FALSE`.

Bonus chatter: Why does the `CreateStreamOnHGlobal` function have such dark and shady corners? This function dates back to Windows 3.1, back when a powerful computer had 4MB of memory, and your typical computer had much less. Software development kits cost thousands of dollars, and the expectation was that if you bought one, it was because you were a professional developer who understood how the system worked down to a very low level. Programming was hard because nobody expected it to be easy.

Raymond Chen

Follow