More undocumented behavior and the people who rely on it: Output buffers

devblogs.microsoft.com/oldnewthing/20050901-17

September 1, 2005

Raymond Chen

For functions that return data, the contents of the output buffer if the function fails are typically left unspecified. If the function fails, callers should assume nothing about the contents.

But that doesn't stop them from assuming it anyway.

I was reminded of this topic after reading Michael Kaplan's story of one customer who wanted the output buffer contents to be defined even on failure. The reason the buffer is left untouched is because many programs assume that the buffer is unchanged on failure, even though there is no documentation supporting this behavior.

Here's one example of code I've seen (reconstructed) that relies on the output buffer being left unchanged:

```
HKEY hk = hkFallback;
RegOpenKeyEx(…, &hk);
RegQueryValue(hk, …);
if (hk != hkFallback) RegCloseKey(hk);
```

This code fragment starts out with a fallback key then tries to open a "better" key, assuming that if the open fails, the contents of the `hk` variable will be left unchanged and therefore will continue to have the original fallback value. This behavior is not guaranteed by the specification for the `RegOpenKeyEx` function, but that doesn't stop people from relying on it anyway.

Here's another example from actual shipping code. Observe that the `CRegistry::Restore` method is documented as "If the specified key does not exist, the value of 'Value' is unchanged." (Let's ignore for now that the documentation uses registry terminology incorrectly; the parameter specified is a value name, not a key name.) If you look at what the code actually does, it loads the buffer with the original value of "Value", then calls the `RegQueryValueEx` function twice and ignores the return value both times! The real work happens in the `CRegistry::RestoreDWORD` function. At the first call, observe that it

initializes the `type` variable, then calls the `RegQueryValueEx` function and assumes that it does not modify the `&type` parameter on failure. Next, it calls the `RegQueryValueEx` function a second time, this time assuming that the output buffer `&Value` remains unchanged in the event of failure, because that's what `CRegistry::Restore` expects.

I don't mean to pick on that code sample. It was merely a convenient example of the sorts of abuses that Win32 needs to sustain on a regular basis for the sake of compatibility. Because, after all, people buy computers in order to run programs on them.

One significant exception to the "output buffers are undefined on failure" rule is output buffers returned by COM interface methods. COM rules are that output buffers are always initialized, even on failure. This is necessary to ensure that the marshaller doesn't crash. For example, the last parameter to the IUnknown::QueryInterface method must be set to NULL on failure.

Raymond Chen

Follow