Why does RegOpenKey sometimes (but not always) fail if I use two backslashes instead of one?

devblogs.microsoft.com/oldnewthing/20121004-00

October 4, 2012

Raymond Chen

A customer reported that on Windows XP, they observed that their program would very rarely get the error `ERROR_INVALID_ARGUMENT` when they passed two backslashes instead of one to the `RegOpenKeyEx` function:

```
RegOpenKeyEx(hk, L"Blah\\\\Oops", ...);
```

After removing C++ escapes, the resulting string passed to `RegOpenKeyEx` is

```
Blah\\Oops
```

The failure was very sporadic and not reproducible under controlled conditions.

Well, first of all, doubled backslashes are not legal in registry key paths in the first place, so the first recommendation is *stop doubling the backslashes*. Once you fix that, the problem will go away.

But the next question is why the error was detected sometimes but not always.

When an application tries to open a registry key, the registry code first consults a cache of recently-opened keys, since registry accesses exhibit very high locality of reference. If a match is found in the cache, then the cached result is used. Otherwise, it's a cache miss, and the registry tree is searched in the old-fashioned way. The registry tree search rejects the double-backslash since it interprets the path `Blah\\Oops` as "Look for a subkey called `"Blah"`, then a subkey called `""`, then a subkey called `"Oops"`." The "subkey called `""` " step fails because key cannot have an empty string as their name.

On the other hand, the code that checks the cache has a different search algorithm which happens to have the effect of collapsing consecutive backslashes, so the path `Blah\\Oops` is interpreted as "Look for a subkey called `"Blah"`, then a subkey called `"Oops"`." (Note: "has the effect of". There is no explicit "collapse backslashes" step; it just turns out that the way the path is parsed, consecutive backslashes end up being treated as if they were single backslashes.)

In the customer's case, therefore, the key in question is in the cache most of the time, which is why the doubled backslash is silently corrected to a single backslash. But every so often, the key is not in the cache, and the old-fashioned search is performed. And the old-fashioned search rejects the double-backslash as an invalid path.

The discrepancy in the two parsing algorithms was resolved in Windows Vista, so you'll see this issue only on Windows XP and earlier.

But this historical tidbit does highlight one of the hidden gotchas of optimization: If your optimized version differs from the unoptimized version in cases that are theoretically anyway illegal, you may find yourself chasing elusive bugs when somebody accidentally stumbles into those cases and managed to get away with it… until now.

Raymond Chen

Follow