Why does Windows keep showing the old indirect strings even after I update the binary?

devblogs.microsoft.com/oldnewthing/20120224-00

February 24, 2012

Raymond Chen

If your application uses indirect localized string resources, and you update the application, you may find that Windows keeps using the old string from the previous version of the application.

For example, suppose that you set the localized name for a shortcut to `@C:\Program Files\Contoso\Contoso.exe,-1`, and in version 1 of your program, you have

```
LANGUAGE LANG_ENGLISH, SUBLANG_NEUTRAL
STRINGTABLE
BEGIN
1 "Contoso Document Services"
END
LANGUAGE LANG_GERMAN, SUBLANG_NEUTRAL
STRINGTABLE
BEGIN
1 "Contoso Dokumentdienste"
END
```

For version 2, your marketing team decides that the program should really be called *Contoso Document System*, so you change the resource file to read

```
LANGUAGE LANG_ENGLISH, SUBLANG_NEUTRAL
STRINGTABLE
BEGIN
1 "Contoso Document System"
END
LANGUAGE LANG_GERMAN, SUBLANG_NEUTRAL
STRINGTABLE
BEGIN
1 "Contoso Dokumentsystem"
END
```

The user upgrades to version 2 of your program, but the shortcut on the Start menu still reads *Contoso Document Services*. What's going on?

The shell keeps a cache of indirect localized strings because loading a DLL just to read a string out of it is pretty expensive. This cache is keyed by the string location specifier, and since your string location specifier hasn't changed from its previous value of `@C:\Program Files\Contoso\Contoso.exe,-1`, the shell continues using the value it stored away in its cache, which if the user had previously been using version 1 of your program, is the string *Contoso Document Services*.

Some people, having discovered this behavior, have tried to go in and tinker with the shell's internal cache of indirect localized strings, but such a technique is doomed to failure because the location of that cache changes pretty regularly, and besides, it's an internal implementation detail. (And even if you find it and manage to fiddle with it, you only fix the problem for the current user. Other users will still have the stale cache entry.)

The best solution is to treat indirect strings as locked: Once you decide what a string should say, you can't change it. When you issue version 2 of `Contoso.exe`, you can create a second string

```
LANGUAGE LANG_ENGLISH, SUBLANG_NEUTRAL
STRINGTABLE
BEGIN
1 "Contoso Document Services" // shortcuts from version 1.0 use this
2 "Contoso Document System" // shortcuts from version 2.0 use this
END
LANGUAGE LANG_GERMAN, SUBLANG_NEUTRAL
STRINGTABLE
BEGIN
1 "Contoso Dokumentdienste" // shortcuts from version 1.0 use this
2 "Contoso Dokumentsystem" // shortcuts from version 2.0 use this
END
```

and have the installer for version 2.0 create a shortcut whose indirect localized string specifier is

```
@C:\Program Files\Contoso\Contoso.exe,-2
```

I admit that this method is rather clumsy and requires more attention on the part of the developer. Everybody wants the "cheap" way out, where the definition of "cheap" is not "cheapest for the customer" but rather "cheapest for me, the developer, because there's a new episode of *Doctor Who* tonight and I don't want to miss it."

We saw last time that the format for indirect localized string resources has room for a comment. And it's the comment that we can take advantage of here. The shell uses the entire string location specifier as the key for its cache lookup, and that string *includes the comment*. Therefore, if you simply change the comment, that results in a cache miss, and the shell will go and re-fetch the string.

```
@C:\Program Files\Contoso\Contoso.exe,-1;v2
```

By appending a `;v2` to the string, you made it different from its predecessor, which means that the string cached by the predecessor won't be used.

As I noted, this is cheap for the developer, but not necessarily cheap for the customer. Suppose the customer copied the shortcut to Contoso version 1 to their desktop, then upgraded to version 2. The upgrade replaces the shortcut in the Start menu, but the copy on the desktop remains unchanged. You now have a shortcut on the desktop whose indirect string is

```
@C:\Program Files\Contoso\Contoso.exe,-1
```

and a shortcut on the Start menu whose indirect string is

```
@C:\Program Files\Contoso\Contoso.exe,-1;v2
```

Since the shortcut on the desktop was created while version 1 was still installed on the computer, its name will read *Contoso Document Services* because that was the contents of string 1. On the other hand, the shortcut on the Start menu will read *Contoso Document System* because its use of the `;v2` forced the shell to go back and look again, and this time it sees the revised string. So far so good.

But then the user does something which causes the cache to be pruned, like, say, changing their UI language to German. The shell says, "Okay, the UI language changed, I need to go reload all these indirect strings because MUI is going to change them to the new language." The shell sees the shortcut on the Start menu, reads string 1 out of `Contoso.exe`, and gets *Contoso Dokumentsystem*. The shell then sees the shortcut on the desktop, reads string 1 out of `Contoso.exe`, and gets… *Contoso Dokumentsystem*. Not *Contoso Dokumentdienste*.

Notice that the name of the shortcut on the desktop was silently upgraded to Contoso version 2.

Even if the user changes the language back to English in an attempt to get things back to the way they were, it won't work. The shell sees the shortcut on the Start menu, reads string 1 out of `Contoso.exe`, and gets *Contoso Document System*. The shell then sees the shortcut on the desktop, reads string 1 out of `Contoso.exe`, and gets *Contoso Document System*, not *Contoso Document Service*. The original string from the first version of `Contoso.exe` is already gone; the only way to get it back is to reinstall Contoso version 1.

But at least you didn't miss your TV show.

Bonus chatter: The one case I can think of where the cheap way out is acceptable is when you are issuing a prerelease version. For your prerelease versions, you can append `;prerelease build xxxxx` to your string location specifier (where `xxxxx` is the build number), so that each time the user upgrades to a new build, the string is reloaded from

scratch. This still has the same problem described above if the user has data left over from a previous build, but since it's a prerelease build, you can just declare that as not a supported configuration.

Raymond Chen

Follow