The long and sad story of the Shell Folders key

devblogs.microsoft.com/oldnewthing/20031103-00

November 3, 2003

Raymond Chen

When you are attempting to architect an operating system, backwards compatibility is one of the ones you just have to accept. But when new programs **rely on** app hacks designed for old programs, that makes you want to scream. Once upon a time, in what seems like a galaxy far far away (a Windows 95 beta release known as "M3"), we documented a registry key called "Shell Folders" that programs could read to obtain the locations of various special folders like the Fonts folder or the My Documents folder. The developers who received Windows 95 M3 Beta followed the documentation and used that key. In the meantime, Windows 95 work continued, and we realized that a registry key was the wrong place to store this information. In part, because a lot of things (like the Control Panel) aren't disk directories so they wouldn't be expressible there. And in another part, because we had forgotten to take into account a feature of Windows NT called roaming user profiles, where your user profile can move around from place to place, so a hard-coded path in the registry is no good. So we created the function `SHGetSpecialFolderLocation`, and updated the documentation to instruct developers to use this new function to obtain the locations of various special folders. The documentation on the old "Shell Folders" key was removed. But to ease the transition from the M3 documentation to the RTM documentation, we left the old "Shell Folders" registry key around, "temporarily", but it was no longer the location where this information was kept. It was just a shadow of the "real" data stored elsewhere ("User Shell Folders"). We shipped Windows 95 RTM with this "temporary" key, because there were still a small number of programs (let's say four) that hadn't finished converting to the new `SHGetSpecialFolderLocation` function. But the support for this registry key was severely scaled back, so it was just barely good enough for those four programs. After all, this was just a backwards compatibility hack. All new programs should be using `SHGetSpecialFolderLocation`. In other words, **the "Shell Folders" key exists solely to permit four programs written in 1994 to continue running on the RTM version of Windows 95**. You can guess what happened next. Windows 95 came out and everybody and their brother wanted to write programs for it. But reading documentation is a lot of work. So when there's some setting you want to retrieve, and you don't want to read documentation, what do you do? You search the registry! (Sound familiar? People still do this today.) So now there were hundreds, thousands of programs which didn't call `SHGetSpecialFolderLocation`; they just went directly for the "Shell Folders" key. But they

didn't realize that the support for "Shell Folders" was only barely enough to keep those four original programs working. For example, did you know that if you never open your Fonts folder, and if no program ever calls `SHGetSpecialFolderLocation(CSIDL_FONTS)`, then there will not be a "Fonts" entry in the "Shell Folders" key? That's because those entries are created only if somebody asks for them. If nobody asks for them, then they aren't created. No point setting up an app hack until it is needed. Of course, when you're testing your program, you don't reformat your hard disk, install Windows 95 clean, then run your program. You just put your program on a Windows 95 machine that has been running for months and see what happens. And what happens is that, since at some point during all those months you opened your Font folder at least once, the "Fonts" entry exists and you are happy. And then back in our application compatibility labs, your program gets a "Fail" grade because our lab reformats the computer before installing each application to make sure there is nothing left over from the previous program before installing the next one. And then the core development team gets called in to figure out why this program is getting a "Fail" grade, and we find out that in fact this program, when faced with a freshly-formatted machine, **never worked in the first place**. Philosophical question: If a program never worked in the first place, is it still a bug that it doesn't work today? Now there are those of you who are licking your lips and saying, "Wow, there's this 'User Shell Folders' key that's even cooler than the 'Shell Folders' key, let me go check it out." I implore you to exercise restraint and not rely on this new key. Just use the function `SHGetFolderPath`, which returns the path to whatever folder you want. Let the "User Shell Folders" key rest in peace. Because in Longhorn, we're doing even more stuff with user profiles and I would personally be very upset if we had to abandon the "User Shell Folders" key as "lost to backwards compatiblity" and set up shop in a new "Real User Shell Folders" key. I strongly suspect that of those four original programs for which the "Shell Folders" key was originally created, not a single one is still in existence today.

Raymond Chen

Follow