What the various registry data types mean is different from how they are handled

devblogs.microsoft.com/oldnewthing/20090205-00

February 5, 2009

Raymond Chen

Although you can tag your registry data with any of a variety of types, such as `REG_DWORD` or `REG_BINARY` or `REG_EXPAND_SZ`. What do these mean, really?

Well, that depends on what you mean by *mean*, specifically, who is doing the interpreting.

At the bottom, the data stored in the registry are opaque chunks of data. The registry itself doesn't care if you lie and write two bytes of data to something you tagged as `REG_DWORD`. (Try it!) The type is just another user-defined piece of metadata. The registry dutifully remembers the two bytes you stored, and when the next person comes by asking for the data, those two bytes come out, along with the type `REG_DWORD`. Garbage in, garbage out. The registry doesn't care that what you wrote doesn't many any sense any more than the NTFS file system driver doesn't care that you wrote an invalid XML document to the file `config.xml`. Its job is just to remember what you wrote and produce it later upon request.

There is one place where the registry does pay attention to the type, and that's when you use one of the types that involve strings. If you use the `RegQueryValueA` function to read data which is tagged with one of the string types (such as `REG_SZ`), then the registry code will read the raw data from its database, and then call `WideCharToMultiByte` to convert it to ANSI. But that's the extent of its assistance.

Just as the registry doesn't care whether you really wrote four bytes when you claimed to be writing a `REG_DWORD`, is also doesn't care whether the various string types actually are of the form they claim to be. If you forget to include the null terminator in your byte count when you write the data to the registry, then the null terminator will not be stored to the registry, and the next person to read from it will not read back a null terminator.

This simplicity in design pushes the responsibility onto the code that uses the registry. If you read a registry value and the data is tagged with the `REG_EXPAND_SZ` type, then it's up to you to expand it if that's what you want to do. The `REG_EXPAND_SZ` value is just part of the secret handshake between the code that wrote the data and the code that is reading it, a secret handshake which is well-understood *by convention*. After all, if `RegQueryValueEx` automatically expanded the value, then how could you read the original unexpanded value?

Windows Vista added a new function `RegGetValue` which tries to take care of most of the cumbersome parts of reading registry values. You can tell it what data types you are expecting (and it will fail if the data is of an incompatible type), and it coerces the data to match its putative type. For example, it auto-expands `REG_EXPAND_SZ` data, and if a blob of registry data marked `REG_SZ` is missing a null terminator, `RegGetValue` will add one for you. Better late than never.

Raymond Chen

Follow