Why don’t Windows Imaging Component pixel format GUIDs continue their nice pattern?

devblogs.microsoft.com/oldnewthing/20240729-00

July 29, 2024

Raymond Chen

A customer pointed out that the original pixel format GUIDs for the Windows Imaging Component (WIC) follow a pattern: The first 15 bytes are all the same, and the final byte starts at 1 and increases consecutively until `0x42`. On the other hand, the pixel formats that were added later do not follow the pattern. Why don’t they just pick up where the previous GUIDs left off?

Well, there was no promise that pixel format GUIDs would follow this pattern, so the component was well within its rights to break from the pattern. And in fact breaking from the pattern is important: Patterns are predictable, and sometimes you don’t want predictability.

Somebody might invent a custom pixel format and cleverly assign it a GUID that followed the pattern, with byte `0x43` as the final byte. They have squatted on that next GUID.

Suppose the next version of WIC introduces support for a new pixel format and assigns it the GUID that ends with `0x43`. Now, when a program asks for a decoder of that format, it might get the one defined by WIC. Or if running on an older system, it gets the one defined by that third party. Conversely, if a program that asks for the third party decoder runs on a newer system, it gets the new system format and not the third party format. In both cases, the request for the decoder succeeds, but with the *wrong* decoder (from the point of view of each program).

That’s bad.

If you create a pattern, then it's possible for somebody to anticipate the next value in the pattern and create a collision. Which sort of defeats one of the primary features of GUIDs, namely that they are unique!

Any group of bitmap formats that are introduced at the same time can follow a pattern, because nobody knows the pattern ahead of time, so they can't anticipate the next value and create a conflict.

The customer was asking about the GUID pattern because they had code that took advantage of the pattern when mapping pixel format GUIDs into its own enumeration: When given a pixel format GUID, they check the first 15 bytes, and if they match the pattern, then they use the last byte as an index into a table.

Note that even if WIC followed its pattern, that wouldn't avoid the need for the customer to issue an update to their library: They would still have to extend their table so that these new values have mappings. So it's not like following the pattern avoided a servicing event.

Now, it's true that if WIC had followed its pattern, then adding support for the new formats would have been simpler for the library. Sorry. You can use the pattern to recognize the "classic" formats, and then use another table of GUIDs for recognizing the "newcomer" formats.