Is there a built-in way in C++/WinRT to get the string name for a Windows Runtime enum?

devblogs.microsoft.com/oldnewthing/20240626-00

June 26, 2024

Raymond Chen

Say you have a value from a Windows Runtime enumeration and you want to convert it to a corresponding string. You can do this in C# with the `ToString()` method.

```
void LogStatus(AsyncStatus status)
{
    Log("Status: " + status.ToString());
}
```

But C++/WinRT doesn't provide a `ToString()` method.

C++/WinRT produces standard C++, and at least as of C++20, there is no standard for reflection, so there is no standard C++ way for converting a value to a string.[1]

C++/WinRT could have included value-to-string conversion in its code generation, but it doesn't. Partly because it's hoping that the C++ language will provide reflection. And partly because experience has told us that value-to-string conversion is rarely useful in production.

For one thing, you would never want to do a value-to-string conversion for a string to display to the user. Those strings are going to be in English (which may not be appropriate for your audience), and they will require domain-specific knowledge to interpret (which is definitely not appropriate for your audience).[2]

In practice, stringification is useful only for logging purposes, and people who have done this say that generally regret it and wish they had just logged as plain integers.

One reason is that if new values are added to the enum after the program is compiled, the program won't know how to convert those new values to strings, so it'll presumably log them as integers. And then a newer version of the program is compiled which understands these new values, and those new values are now logged as strings. The same value is now being logged in two different ways, and the back end processing has to merge those two columns into one. Better to just leave them as integers and let the back end do the stringification, so that it’s consistent. If a new enum value is added, you just add it to the back end and the strings are all updated, even for values coming from older versions of the program that predated the new value.

Another reason is that it is much easier to parse integers on the back end. If you logged them as strings, then the back end would have to have a reverse parser to convert them back to values if it wanted to do further processing.

You can do the value-to-string conversion on your back end, probably as a final step before displaying them on a dashboard.

[1] That hasn’t stopped people from coming up with clever hacks that work under certain conditions. The most ambitious of those clever hacks is probably magic_enum.

A proposal for reflection has made significant progress for C++26 and may even have been approved by the time you read this.

[2] The exception to this would be a diagnostic tool whose audience is other software developers, not the general user population. If you are really motivated, you could write a tool that uses the xlang library (the same library that C++/WinRT itself uses) to parse the winmd files and produce stringification functions.

Alternatively, you could write a C# program that ingests the winmd files and uses C# reflection to produce the stringifiers. That’s what my colleague Alexander Sklar did: CppWinRT.Builders.