Adding flags to APIs to work around driver bugs doesn't scale

devblogs.microsoft.com/oldnewthing/20060405-13

April 5, 2006

Raymond Chen

Some people suggested, as a solution to the network interoperability compatibility problem, adding a flag to `IShellFolder::EnumObjects` to indicate whether the caller wanted to use fast or slow enumeration.

Adding a flag to work around a driver bug doesn't actually solve anything in the long term.

Considering all the video driver bugs that Windows has had to work around in the past, if the decision had been made to surface all those bugs and their workarounds to applications, then functions like `ExtTextOut` would have several dozen flags to control various optimizations that work on all drivers except one. A call to `ExtTextOut` would turn into something like this:

```
ExtTextOut(hdc, x, y, ETO_OPAQUE |
           ETO_DRIVER_REPORTS_NATIVE_FONTS_CORRECTLY |
           ETO_DRIVER_WILL_NOT_DITHER_TEXT_DURING_BLT |
           ETO_DRIVER_DOES_NOT_LIE_ABOUT_LOCAL_TRANSFORMS |
           ETO_DRIVER_DOES_NOT_CRASH_WITH_STOCK_BRUSHES,
           &rcOpaque, lpsz, cch, NULL);
```

where each of those strange flags is there to indicate that you want to obtain the performance benefits enabled by each of those flags because you know that you aren't running on a version of the video driver that has the particular bug each of those flags was created to protect against.

And then (still talking hypothetically) with Windows Vista, you find that your program runs slower than on Windows XP: Suppose a bug is found in a video driver where strings longer than 1024 characters come out garbled. Windows Vista therefore contained code to break all strings up into 1024-character chunks, but as an optimization you could pass the `ETO_PASS_LONG_STRINGS_TO_DRIVER` flag to tell GDI not to use this workaround. Your Windows XP program doesn't use this flag, so it now runs slower on Windows Vista. You'll have to ship an update to your program just to get back to where you were.

It's not limited to flags either. By this philosophy of "Don't try to cover up for driver bugs and just make applications deal with them", you would have had the following strange paragraph in the `FindNextFile` documentation:

> If the `FindNextFile` function returns `FALSE` and sets the error code to `ERROR_NO_MORE_FILES`, then there were no more matching files. Some very old Lan Manager servers (circa 1994) report this error condition prematurely. If you are enumerating files from an old Lan Manager server and the `FindNextFile` function indicates that there are no more files, call the function a second time to confirm that there really are no more files.

Perhaps it's just me, but I don't believe that workarounds for driver issues should become contractual. I would think that one of the goals of an operating system would be to smooth out these bumps and present a uniform programming model to applications. Applications have enough trouble dealing with their own bugs; you don't want them to have to deal with driver bugs, too.

Raymond Chen

Follow