Be very careful if you decide to change the rules after the game has ended

devblogs.microsoft.com/oldnewthing/20060410-14

April 10, 2006

Raymond Chen

One suggestion for addressing the network compatibility problem was returning an error code like `ERROR_PLEASE_RESTART` which means "Um, the server ran into a problem. Please start over." This is basically the same as the "do nothing" option, because the server is already returning an error code specifically for this problem, namely, `STATUS_INVALID_LEVEL`. Now, sure, that error doesn't actually mean "Please try again," It actually means "Sorry, I can't do that." This is the error code that is supposed to come if you ask a server to go into fast mode and it doesn't support fast mode. But the effect from a coding standpoint is the same. "If `FindNextFile` return the error `xyz`, then the server ran into a problem and you should start over." Call `xyz` "`ERROR_PLEASE_RESTART`" or "`STATUS_INVALID_LEVEL`" or "`PURPLE_LILACS`". No matter what you pick, the net effect is the same: Existing code must be changed to specifically check for this new error code and react accordingly. Programs that aren't updated will behave strangely. And that's the issue faced by today's topic: When do you decide that a problem requires you to change the rules of the game after it has ended? Programs out there were written to one of may sets of rules. Most of them were written to Windows XP's rules. Some were written to Windows 2000's rules. Even older programs may have been written to Windows 95's or Windows 3.1's rules. One aspect of backwards compatibility is accomodating programs that broke the rules and got away with it. But here, the issue is not fixing broken programs; it's keeping correct programs correct. If you introduce a new error code and specify an unusual recommended action (i.e., something other than "fail the operation"), then all programs written prior to the introduction of this rule have suddenly become "wrong" through no fault of their own. Depending on how "wrong" they are, the severity of the problem can range from inconvenient to fatal. In the Explorer case, the directory comes up wrong the first time but fixes itself if you refresh. But if a .NET object's enumerator suddenly threw a new `ServerFailedMustRestartEnumeration` exception, you're probably going to see lots of programs crash with unhandled exception failures. At this point, the usual suspects come to the surface: How will users get updated programs that conform to the new rules? The original program's author may no longer be alive. The source code may have been lost. Or the knowledge necessary to understand the source code may have been lost. ("This program was written by an outside contractor five years ago. We have the source code but nobody here can

make heads nor tails of it.") Or the program's author may simply not consider updating that program to Windows Vista to be a priority. (After all, why bother updating version 1.0 of a program when version 2.0 is available?)

Mind you, Microsoft does change the rules from time to time. Pre-emptive multi-tasking changed many rules. The new power management policiies in Windows Vista certainly changed the rules for a lot of programs. But even when the rules change, an effort is usually made to continue emulating the old rules for old programs. Because those programs are following a different set of rules, and it's not nice to change the rules after the game has ended.

Raymond Chen

Follow