Why is regsvr32 exiting with code 3?

devblogs.microsoft.com/oldnewthing/20180921-00

September 21, 2018

Raymond Chen

A customer had a script to set up a virtual machine, but this call was failing:

```
regsvr32 /s /n /i:u Awesome.dll
```

The DLL failed to register, and `regsvr32` exited with code 3.

Last time, we saw exit code 3 means that the `LoadLibrary` call failed. The customer reported that the error was not consistent, and they've been working around it by waiting a little while and retrying the operation. But sometimes, even after a few retries, the operation still fails.

The were running `regsvr32` in silent mode, so no error messages were displayed to the user.

According to the table from last time, step 3 is the `LoadLibrary` step. Since the problem was random and sometimes cleared up after a few retries, this ruled out systematic errors like copying the file to the wrong directory, or copying the wrong version of the file. Those types of errors would result in the operation failing consistently, rather than randomly.

I suspected that the `LoadLibrary` failed because the file was still in use, either because it was still being copied to the VM, or because it was being scanned or blocked by anti-malware software running in the VM.

One option for digging further is to run `regsvr32` one last time in non-silent mode, so that the error details are on the screen. They could write an automation client that scrapes the message before dismissing the dialog box. If they go the automation client route, they may as well *always* run `regsvr32` in non-silent mode.

If the team doesn't have experience with writing automation, they could just set a watchdog on `regsvr32`. Pick a generous amount of time to cover typical running time of `regsvr32` in the success cases. If `regsvr32` has not returned by then, then take a screen shot and then terminate the `regsvr32` proces.

Or they could write their own program that tries to `LoadLibrary` their DLL and captures the `GetLastError`. Run the custom program once the first `regsvr32` fails. They could even turn on loader snaps to get extremely detailed information about the `LoadLibrary` operation; that information will pinpoint exactly where it went wrong.

Another option is to run `regsvr32` under the debugger with loader snaps enabled and tell the debugger to log all output to a file.

```
cdb -Ggx -logo log.txt regsvr32 /s /n i:u Awesome.dll
```

If the DLL registers successfully, then delete the log file. If it fails, then save the log file somewhere for analysis.

Yet another possibility is that the exit code of 3 is a red herring. Perhaps something went wrong in a way that led to the C runtime calling the `abort()` function, which exits the program with code 3.

Raymond Chen

Follow