How can I specify that my DLL should resolve a DLL dependency from the same directory that the DLL is in?

devblogs.microsoft.com/oldnewthing/20171011-00

October 11, 2017

Raymond Chen

A customer had a program that loaded two DLLs, let's call them `A.DLL` and `B.DLL`. Both of those DLLs use a common helper DLL called `C.DLL`. The catch is that the two DLLs want to use *different incompatible versions* of `C.DLL`. The two DLLs `A.DLL` and `B.DLL` reside in separate folders, and each folder has a corresponding copy of `C.DLL`.

An additional complicating factor is that `A.DLL` was written by a third party and cannot be modified.

The customer was hoping there would be some way to get the two DLLs `A.DLL` and `B.DLL` to use their respective versions of `C.DLL`. They suspect that some magic with activation contexts and manifests might do the trick, but they didn't have the expertise to figure out exactly what. (And remember that `A.DLL` came from a third party and cannot be modified.)

Eugene Talagrand explained that you can solve the problem with manifests. Embed a manifest into `B.DLL` with resource ID 2 that looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
    <file name="C.dll" />
</assembly>
```

You don't have to make any changes to the code that loads `B.DLL`. When you try to load `B.DLL`, the system will recognize the manifest, and that manifest tells the system to ignore any rogue copies of `C.DLL` and link `B.DLL` to the copy of `C.DLL` in the same directory. Furthermore, this special "`B.DLL`-specific" version of `C.DLL` is not made visible to other DLLs (unless they specifically ask for it with their own manifest), so when the program loads `A.DLL`, it will ignore the "`B.DLL`-specific" copy and look for `C.DLL` using the traditional search path.

The customer confirmed that adding the manifest to `B.DLL` worked!

Note that the manifest declaration applies to DLL dependencies resolved when `B.DLL` is loaded. If `B.DLL` performs a `LoadLibrary("C.DLL")` at run time, then it will need to make its activation context active when it loads the DLL so that the system knows to follow the instructions in the manifest. For more details, you can read more about using manifests to isolate DLLs on Eugene's blog.

Raymond Chen

Follow