DLL forwarding is not the same as delay-loading

devblogs.microsoft.com/oldnewthing/20080204-00

February 4, 2008

Raymond Chen

As I noted earlier, when you create a forwarder entry in an export table, the corresponding target DLL is not loaded until somebody links to the forwarder entry. It looks like some people misread this statement to suggest some sort of delay-loading so I'm going to state it again with an example in mind in the hopes of clearing up any confusion (and risking creating more confusion than I clear up).

Suppose that you have a DLL called `A.DLL` that has a forwarder entry to `B.DLL`:

```
; A.DEF
EXPORTS
  Dial = B.Call
  Pour
  Refill
```

This specifies that if somebody wants the function `Dial` from `A.DLL`, they will actually get the function `Call` from `B.DLL`. The delay-load-like behavior is that `B.DLL` is not loaded until somebody asks for the `Dial` function.

I will use the notation `DLLNAME!FunctionName` to mean "the function `FunctionName` from the DLL named `DLLNAME`." This is the notation used by the `ntsd` debugger.

Consider this program:

```
POURME.EXE
 Imports from A.DLL
  Pour
  Refill
```

The `POURME` program will not result in `B.DLL` being loaded since it never links to `A!Dial`. Of course `A.DLL` will get loaded because the program wants the functions `A!Pour` and `A!Refill`. This is the "delay-load-like behavior" I mentioned in the original entry: If you don't call a function that forwards to `B.DLL`, then `B.DLL` won't get loaded.

Alternative, you could have used this method to do the forwarding:

```
; A2.DEF
EXPORTS
 Dial
 Pour
 Refill
/* a2.c */
// Forward Dial to B!Call
HRESULT Dial()
{
 return Call();
}
```

This pseudo-forwarder is not a forwarder in the linker sense; it is an attempt to emulate linker forwarding in code. Now let's look at the corresponding alternate `POURME` program:

```
POURME2.EXE
 Imports from A2.DLL
  Pour
  Refill
```

Even though `POURME2` doesn't call `A2!Dial`, the file `B.DLL` will nevertheless be loaded when `POURME2` runs because `A2.DLL` contains a dependency on `B.DLL` in its own import table:

```
; dump of headers of A2.DLL
 Imports from B.DLL
  Call
```

Loading `A2.DLL` will cause `B.DLL` to be loaded since `B.DLL` is listed as one of `A2`'s dependencies.

Commenter bruteforce got off on the wrong foot by calling the above mechanism a delay-loading feature.

> I tried to take advantage of the delay-loading feature described above for the forwarder DLLs…

The mechanism is not delay-loading and I never said that it was. The quasi-delay-load behavior is that a forwarded-to DLL is not loaded until somebody links to it. The term delay-loading typically is used to apply to delaying the load of a module until a function in that module is called. But import resolution happens at load time, not run time.

Commenter bruteforce tried to create a forwarder to a nonexistent function, and then tried to link to the forwarder DLL. As we saw above, this triggers an attempt to resolve the forward by loading the forwarded-to DLL and looking for the function. If this fails, then the original import request is declared to have failed. This all happens as part of the import resolution process. And as we saw many years ago, Win32 fails a module load if an import cannot be resolved. Since the forwarder cannot be resolved, the load fails. Import forwarding

functionality is completely unsuitable for functions whose presence you wish to detect and respond to at runtime. As with all imports, an import failure is considered a fatal error. If you want delay-loading, then you need to do delay-loading. Forwarding is not delay-loading.

Raymond Chen

Follow