How can I control the directory from which my delay-loaded DLL is loaded?

devblogs.microsoft.com/oldnewthing/20170126-00

January 26, 2017

Raymond Chen

A customer had a DLL that is a COM in-process server. This DLL gets loaded by arbitrary client applications, and it also uses the `/DELAYLOAD` linker flag to delay-load many of the DLLs which it depends upon. The customer observed that these delay-loaded DLLs were being loaded according to the standard DLL-loading search algorithm.[1] The problem is that the DLL which they are dependent upon is not in the standard search path; it's in its own private directory.

The customer explained that they are working around this problem by providing a custom delay-load helper function which calls `SetDllDirectory` to add the private directory to the DLL search path.

The customer wanted to know whether `SetDllDirectory` affects the DLL search path for the entire process. The customer doesn't want to affect the DLL search path for the entire process because the DLL is a guest in the host process, so it shouldn't be changing the carpet. (Hey, at least they're not calling in a demolition team.) "If it affects only the DLL we need to load, then it looks like `SetDllDirectory` is what we need. But if it affects the entire process, then we would have to switch to `AddDllDirectory`."

Yes, the `SetDllDirectory` function affects the DLL search path for the entire process. It's not clear what the customer's mental model is for "affects only the DLL we need to load", seeing as you don't actually pass `SetDllDirectory` the name of the DLL you need to load, so it has no idea which DLL to apply this path to.

The customer's proposed alternative of using `AddDllDirectory` doesn't solve the problem, because it too affects the DLL search path for the entire process. Maybe they were thinking of calling `AddDllDirectory` to add the private directory, then calling `RemoveDllDirectory` to remove it at some unspecified point in the future. But that creates a window in which the process DLL path has the private directory, and if another thread also calls `LoadLibrary`, it will see that other private directory, which is presumably unwanted.

The customer is making things too hard for themselves by manipulating the DLL search paths. Since they know what directory the DLL is in, they don't need to do any searching at all. When the notification handler is called, it is given a few pieces of information.

- The reason why the handler is being called.
- Information about the DLL being loaded.

Since the customer already has a custom handler, they can just write their custom handler like this:

```
FARPROC WINAPI delayHook(unsigned dliNotify, PDelayLoadInfo pdli)
{
 if (dliNotify == dliNotePreLoadLibrary &&
     StrCmpIC(pdli->szDll, "special.dll") == 0)
 {
  return LoadTheSpecialDll();
 }
 ...
}

HMODULE LoadTheSpecialDll()
{
 .. calculate the full path to the special DLL in its
 .. private directory
 return LoadLibrary(fullPathToSpecialDll);
}
```

If the notification handler is being told that we are about to load `special.dll`, then load the special DLL using whatever custom algorithm you need, and return that handle. The delay-load library will use that module instead of trying to load via the standard DLL search directory. There's no need to mess around with `Get` / `SetDllDirectory`, which is a good thing, since that avoids applying a global solution to a local problem.

[1] This is explained in the documentation, because it says that the default delay-load helper function calls the `LoadLibrary` function, which is subject to the standard DLL search path. Though technically, it calls `LoadLibraryEx` and passes a flags value of 0, which is functionally equivalent. You can see this and more in the file `delayhlp.cpp` in the `VC\Include` directory.

Raymond Chen

Follow