# The unfortunate interaction between LOAD_LIBRARY_AS_DATAFILE and DialogBox

devblogs.microsoft.com/oldnewthing/20051006-09

October 6, 2005

Raymond Chen

Some people have noticed that if you load a DLL with the `LOAD_LIBRARY_AS_DATAFILE` flag, you sometimes get strange behavior if you then pass that `HINSTANCE` to a dialog box function.

The problem here is that since the bottom 16 bits of a proper `HINSTANCE` are always zero, different components have "borrowed" those bits for different purposes. The kernel uses the bottom bit to distinguish modules loaded by having been mapped into memory as sections (i.e., loaded normally) from those who have been mapped as one giant block (loaded as a datafile). It needs to know this so that the various resource-management functions such as the `FindResource` function know how to interpret the data in order to locate the resource in question. Although everybody now knows that the `HINSTANCE` is the base address of the DLL, in principle, it is an opaque value (and in the 16-bit world, the value was indeed opaque).

Meanwhile, the window manager has its own problems. In order to support 16-bit applications running seamlessly on the desktop (not in a virtual machine, as discussed earlier), as well as thunking between 16-bit and 32-bit code, it needs to accept both 32-bit `HINSTANCE` values as well as 16-bit `HINSTANCE` values. Since memory allocation granularity is 64KB, the window manager knows that valid 32-bit `HINSTANCE`s have zero in the bottom 16 bits, whereas 16-bit `HINSTANCE` values are nonzero there.

Perhaps you see the conflict now.

If you pass the instance handle of a DLL loaded as a datafile, the kernel will set the bottom bit as a signal to itself to locate its resources in the flat datafile manner rather than in the mapped DLL manner. But the window manager sees that the bottom 16 bits are not all zero and assumes that it has been given a 16-bit `HINSTANCE` value.

Amazingly, this doesn't cause a problem most of the time because the things that need to be handled differently between 32-bit and 16-bit `HINSTANCE`s are relatively minor. The one that is most likely to bite you is the dialog instance data segment.

In 16-bit Windows, a dialog box came with its own data segment, which was used as the local data segment for controls hosted by that dialog box. Most controls didn't need a lot of storage in the local data segment, so the issue of where it came from wasn't really important. The big exception was edit controls, since they can contain multiple kilobytes of text. a dozen kilobytes of text may very well not fit in the application's data segment. Therefore, creating a new data segment gave the edit controls on the dialog a new 64KB block of memory to store their data in. Programs were expected to extract the data from the edit control via mechanisms such as the `GetWindowText` function and store the result someplace that had the capacity to handle it (outside the cramped local data segment).

In order to maintain compatibility with 16-bit programs who are expecting this behavior to continue, the window manager, when it sees a 16-bit `HINSTANCE`, dutifully creates a 16-bit data segment in which to store the data for the edit controls, using a helper function provided by the 16-bit emulation layer. But if you aren't really a 16-bit program, then the 16-bit emulation layer is not active, and consequently it never got a chance to tell the window manager how to create one of these compatibility segments. Result: Crash.

The solution is to add the `DS_LOCALEDIT` style to your dialog box styles. This flag means "Do not create a dialog box data segment; just keep using the data segment of the caller." Therefore, when your `LOAD_LIBRARY_AS_DATAFILE` `HINSTANCE` is mistaken for a 16-bit dialog template, the dialog manager won't try to create a dialog box data segment and therefore won't call that function that doesn't exist.

I believe this issue has been resolved in Windows XP SP 2. The window manager uses a different mechanism to detect that it is being asked to create a dialog box on behalf of a 16-bit program and is no longer faked out by the faux-`HINSTANCE`s produced by the `LoadLibraryEx` function.

Raymond Chen

**Follow**