

## Re: difference between a .dll and .ocx????

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2007-01/msg00008.html>

---

- *From:* "Peter Duniho" <NpOeStPeAdM@xxxxxxxxxxxxxxxxxxxx>
  - *Date:* Mon, 1 Jan 2007 15:24:39 -0800
- 

"Scott M." <s-mar@xxxxxxxxxxxxxxxx> wrote in message  
<news:Oozi3AdLHHA.320@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

All .dll's  
and .exe's (prior to .NET) as well as .ocx's conform to this  
standard

That's not true.

How so? If (prior to .NET), you built an .exe or .dll (that runs on  
Windows), then it must be a COM library since, that's the architecture  
that Windows is built on.

You are basing your conclusion on false assumptions. Windows is NOT built  
solely on COM, nor would the architecture on which Windows is built  
necessarily define what some .exe or .dll not part of Windows (even if it  
runs on Windows) is or is not.

The main difference between a .exe and a .dll is that the .exe is entirely  
self-contained with a well-defined entry point where a program can start  
executing. There have been .exe files since long before Windows. A .dll  
also has a well-defined entry point, but it doesn't define where execution  
of a program begins...instead, it's a function that is called when the DLL  
is loaded (there are also special functions called when the DLL is unloaded  
and for other events).

Other than these few special functions (which have nothing to do with COM),  
a DLL is not required to implement anything else. It always does, of  
course, since otherwise it wouldn't be useful. But what it does implement  
is up to the person authoring the DLL. Most DLLs are simply a collection of  
useful functions. For example, the Visual Studio C runtime library, which  
is very much like a statically-linked CRT except that it can be shared by  
all the various executables that use it. It has nothing to do with COM, and  
it is not a COM library.