

# Re: Threading and DllImport

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2007-08/msg00495.html>

---

- *From:* "Bob Milton" <[DocBob1945@xxxxxxxxxxxxxxxxxxxx](mailto:DocBob1945@xxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 28 Aug 2007 09:40:02 -0700
- 

Seth,

That won't work – LoadLibrary simply increments the load count if the DLL has already been loaded. So all you get is the need to call FreeLibrary one more time, you do not get a separate memory space.

Bob

"Seth Gecko" <[segecko@xxxxxxxx](mailto:segecko@xxxxxxxx)> wrote in message  
<news:1188308286.597137.271450@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

On Aug 28, 10:07 am, Peter Duniho <[NpOeStPe...@xxxxxxxxxxxxxxxxxxxx](mailto:NpOeStPe...@xxxxxxxxxxxxxxxxxxxx)>  
wrote:

Seth Gecko wrote:

Hi

I am developing a complex VB.Net Windows application for an engineering firm (don't ask me why they prefer VB.Net...). All the engineering calculations are done in FORTRAN which is compiled to a non COM type DLL (meaning you can't create an Interop for it). This is called from within the .Net application using DllImport, just like using a Windows API call. This works fine. The problem arises when we create threads (using BeginInvoke) and each thread is calling the same DllImported function. This results in various unpredictable errors from within the FORTRAN code, most likely because the FORTRAN function (and its memory) is shared between threads. The FORTRAN function itself is thread safe

## Re: Threading and DllImport

(checked using Intel Thread-Checker), but my guess is that when using DllImport the function is loaded into the application process (as opposed to the thread) and thereby becomes shared.

I don't know anything about "Intel Thread-Checker". However, if the function were truly thread-safe, I don't think you would be having a problem.

Having the same `_code_` shared between threads is not a problem. Code doesn't change (or at least, it should not change...if it does, you have a problem of an entirely different nature). So having multiple threads executing the same code is fine.

What is a problem is having multiple threads accessing the same data at the same time, especially if that data changes. Presumably the "Intel Thread-Checker" is trying to make a determination along these lines, but if you are having problems with the code because of multiple threads executing the same code at the same time, then it seems likely that somewhere there is data being shared amongst those threads even though "Intel Thread-Checker" doesn't see it.

What the exact problem might be, I can't say. You didn't post any code, whether the code in the DLL or the code using the DLL. If the code uses any global variables or static variables (it's been awhile, so I forget what exactly those are called in FORTRAN, but hopefully you get the idea), then access to those variables needs to be synchronized.

Better would be to make sure all variables are local to the called function, or at least local to the thread calling the function (different languages have different mechanisms for declaring thread-specific data...maybe your FORTRAN compiler allows this). Then you never have to worry about contention between the threads for the same data.

If you do post code with the intention of receiving more specific advice, please make sure that it is a concise-but-complete example of code that reliably reproduces the problem. Emphasis on "concise"...don't just post all your code. Whittle it down to the bare minimum required to reproduce the problem.

Pete- Hide quoted text -

- Show quoted text -

You are absolutely correct. The FORTRAN code was not thread safe. We were able to reproduce the problem in a simple example. The use of

## Re: Threading and DllImport

FORTRAN common blocks is definitively not thread safe even though the values written to the common blocks are the same in our test scenario. The solution is not simple though, as there apparently are performance loss if the engineers are to remove those blocks (and performance is the whole reason for using FORTRAN in the first place). Right now we are playing with the idea of loading the FORTRAN dll several times (using LoadLibrary), ie. one for each thread. This should make sure that no memory/data is shared between threads. Any comments on this idea?

Thanks for your help.  
...Casper