

## Re: VC++2003: Methods implemented outside class body are never inlined for templates that are instantiated with `__declspec(dllimport)`

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2005-03/0512.html>

---

**From:** Larry Brasfield (*donotspam\_larry\_brasfield\_at\_hotmail.com*)

**Date:** 03/15/05

Date: Tue, 15 Mar 2005 15:06:16 -0800

"Felix I. Wyss" <felixw@inin.com> wrote in message  
news:OgBryeaKFHA.572@tk2msftngp13.phx.gbl...

> *Good Afternoon,*

Greetings.

> *I recently noticed that some very simple methods of a template declared and  
> used in a DLL library get inlined when used by the DLL itself, but not by  
> other DLLs and EXEs. After some investigating, I narrowed this down to a  
> very odd behavior (bug?) of the VC++.NET 2003 compiler: If a class that is  
> declared as `__declspec(dllimport)` derives from a template, that template's  
> methods are never inlined, even if declared with the "inline" specifier.  
> Methods that are defined within the class body are correctly inlined by the  
> compiler. The same is true to templates explicitly instantiated for a type  
> as "template class `__declspec(dllimport) Template<Type>;`".*

>

> *As this is a bit tricky to explain, consider the following code:*

[Cut 260 lines of C++, text, and assembler output.]

Whew! I think it might be simplest to just take your word for what is happening since you appear to have done your homework, (except for reducing the code to a minimal example!)

First, the 'inline' keyword is strictly an advisory to the compiler. There is no requirement in the C++ standard that the compiler inline whatever is so marked, nor is the compiler precluded from inlining code not so marked.

So 'bug?' cannot be the issue.

One might make an argument (albeit a complex one) that this is a quality of implementation issue. I will take the side of the compiler implementers on this one.

Re: VC++2003: Methods implemented outside class body are never inlined for templates that are instantiated

Why would you use a DLL if it was not supposed to be interchangeable with newer versions or (in the case of plugin architectures) a different implementation? Inlined code from class methods is first and foremost an implementation detail that is exposed only for the sake of execution efficiency. DLL's are primarily motivated by a desire to separate interface from implementation, and bind the two at nearly the latest possible moment. (See "delay loading".) So, if the compiler feature designer(s) elected to not add some syntax for saying "Bind my implementation into the DLL client code." while still allowing inlining for the benefit of the implementation of the DLL itself, I can see that as an arguably sensible decision.

If you cannot get happy with it, I would suggest that a few friends with whatever privileged access those would-be inlined methods enjoyed could be made to accomplish the same effect. That approach will degrade encapsulation somewhat, but that clearly is something you are willing to tolerate.

--

--Larry Brasfield  
email: donotspam\_larry\_brasfield@hotmail.com  
Above views may belong only to me.