

# Re: Obscure marshalling/pinvoke problem with 1, 2, 4, and 8 bytes

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.interop/2006-03/msg00204.html>

---

- *From:* whodges <[whodges@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:whodges@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Mon, 27 Mar 2006 13:27:02 -0800
- 

i wish it were that simple. unfortunately the name-mangling isn't the problem, although i do get a different error message when i use the mangled name. still the same issue though (see below).

ok, here's a complete, super-simple code example of what i'm doing (pardon the length if it seems winded). i've created a C# console app solution using VS 2005 (.NET 2.0), and have added an empty C++ DLL project to it. the C++ DLL contains Structures.h and Exports.cpp. the C# project contains Program.cs and Structure.cs. here are their contents:

--- Structures.h

---

```
#ifndef __STRUCTURES_H__  
#define __STRUCTURES_H__
```

```
struct Structure  
{  
public:  
Structure()  
{  
a = 0;  
b = 0;  
c = 0;  
d = 0;  
e = 0;  
f = 0;  
g = 0;  
h = 0;  
}
```

```
Structure(unsigned char a0, unsigned char b0, unsigned char c0,  
unsigned char d0, unsigned char e0, unsigned char f0, unsigned char g0,  
unsigned char h0)  
{  
a = a0;
```

Re: Obscure marshalling/pinvoke problem with 1, 2, 4, and 8 bytes

```
b = b0;  
c = c0;  
d = d0;  
e = e0;  
f = f0;  
g = g0;  
h = h0;  
}
```

```
public:  
unsigned char a;  
unsigned char b;  
unsigned char c;  
unsigned char d;  
unsigned char e;  
unsigned char f;  
unsigned char g;  
unsigned char h;
```

```
};
```

```
#endif
```

```
-- Exports.cpp
```

---

```
#include "Structures.h"
```

```
#define DLLEXPORT __declspec(dllexport)  
#define STDCALL __stdcall
```

```
extern "C"  
{
```

```
DLLEXPORT void STDCALL SetStructure(Structure s)  
{  
    Structure copy = s;  
    return;  
}
```

```
DLLEXPORT Structure STDCALL GetStructure()  
{  
    Structure s = Structure(8, 7, 6, 5, 4, 3, 2, 1);  
    return s;  
}  
} // extern "C"
```

```
-- Structures.cs
```

---

## Re: Obscure marshalling/pinvoke problem with 1, 2, 4, and 8 bytes

```
using System;
using System.Runtime.InteropServices;

namespace MarshallingTest
{
    [StructLayout(LayoutKind.Sequential)]
    public struct Structure
    {
        public byte a;
        public byte b;
        public byte c;
        public byte d;
        public byte e;
        public byte f;
        public byte g;
        public byte h;

        public Structure(byte a0, byte b0, byte c0, byte d0, byte
e0, byte
f0, byte g0, byte h0)
        {
            a = a0;
            b = b0;
            c = c0;
            d = d0;
            e = e0;
            f = f0;
            g = g0;
            h = h0;
        }
    }
}

— Program.cs
```

---

```
using System;
using System.Runtime.InteropServices;

namespace MarshallingTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Structure s = new Structure(1, 2, 3, 4, 5, 6, 7, 8);
            int size = Marshal.SizeOf(s);
            Console.WriteLine(String.Format("Structure size in
bytes: {0}\n",
```

## Re: Obscure marshalling/pinvoke problem with 1, 2, 4, and 8 bytes

```
size));

// 'Set' test

bool setFailed = false;
try
{
    SetStructure(s);
}
catch
{
    setFailed = true;
}

if (setFailed)
    Console.WriteLine("'Set' function failed\n");
else
    Console.WriteLine("'Set' function passed\n");

// 'Get' test

bool getFailed = false;
try
{
    s = GetStructure();
}
catch
{
    getFailed = true;
}

if (getFailed)
    Console.WriteLine("'Get' function failed\n");
else
    Console.WriteLine("'Get' function passed\n");
}

[DllImport("UnmanagedCode.dll", EntryPoint = "SetStructure")]
public static extern void SetStructure(Structure s);

[DllImport("UnmanagedCode.dll", EntryPoint = "GetStructure")]
public static extern Structure GetStructure();
}

}

--- end
```

---

now, when i call GetStructure in Program.cs, i should get a C# Structure object 's' where s.a == 8, s.b == 7, etc. all the way down to

## Re: Obscure marshalling/pinvoke problem with 1, 2, 4, and 8 bytes

s.h = 1. however, what i end up with is some form of marshalling error. if i run the code as it is above, i end up with a "cannot find entry point" exception when i call GetStructure. if i modify the code so that the 'Entry Point' attributes in the DllImport statements are set to the mangled names of the exported functions (as per Mattias's suggestion – e.g. `_GetStructure@4` and `_SetStructure@8`), i end up with this error:

```
"Managed Debugging Assistant 'PInvokeStackImbalance' has detected a problem in 'C:\Projects\MarshallingTest\MarshallingTest\bin\Debug\MarshallingTest.exe'. Additional Information: A call to PInvoke function 'MarshallingTest!MarshallingTest.Program::GetStructure' has unbalanced the stack. This is likely because the managed PInvoke signature does not match the unmanaged target signature. Check that the calling convention and parameters of the PInvoke signature match the target unmanaged signature."
```

i think the same problem is at work in both cases. now, if i use a `..def` file to export the functions instead, i run into the same issue, only now it says that the memory has been corrupted. again, same problem, slightly different error message.

here's where it gets interesting: if i comment out the 'h' member from both the C# and C++ Structure struct (i.e. so now it's 7 bytes in size rather than 8), then everything works fine. both SetStructure and GetStructure return successfully with the proper results. same thing if i were to add a member 'i' to both Structures in addition to 'h' so that i had a 9 byte structure. that works too. it works fine for any size structure \*except\* a 1, 2, 4, or 8 byte one. btw, 16 bytes and 32 bytes work fine as well, so the 'power of two' pattern doesn't continue past 8.

and to add to the weirdness that i mentioned before: if i remove the constructors from Structures.h on the C++ side, everything works fine again. 1, 2, 4, or 8 bytes – it all works as long as the constructors are gone and i just manually initialize the public members.

i'd love it if someone actually gave this code a try. i'm fairly certain you'll run into exactly the same problem. i'm open to \*any\* ideas or suggestions.

"Mattias Sjögren" wrote:

```
Make sure the exported entry points have non-mangled names (e.g. SetTest instead of _SetTest@8)
```

or

Re: Obscure marshalling/pinvoke problem with 1, 2, 4, and 8 bytes

Set the EntryPoint property of the DllImport attribute to the actual, mangled name with which the function is exported.

Mattias

--

Mattias Sjögren [C# MVP] mattias @ mvps.org

<http://www.msjogren.net/dotnet/> | <http://www.dotnetinterop.com>

Please reply only to the newsgroup.