

## Re: HELP !! pb calling eVC++ DLL from VB.NET on compact framework

**Source:**

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.embedded.vc/2004-12/0036.html>

---

**From:** Paul G. Tobey [eMVP] (*ptobey*)

**Date:** 11/30/04

Date: Tue, 30 Nov 2004 12:55:38 -0700

\*\*\*\*\*

This message is a reply to the orphan thread of the same name in the Compact Framework newsgroup, where Ilya's response is found.

\*\*\*\*\*

No. In fact, you don't have to *\*do\** anything to the buffer to see the error. I was surprised, too. Try the code, but with no modifications or accesses to the buffer local variable at all and you'll still get the exception when the function is called. The allocation of the variable on the stack itself is the problem. Changing only the way in which the buffer is allocated, from a local automatic to a dynamic allocation (same size and everything), causes the problem to go away.

Here's the code from my DLL used to duplicate the problem (note that the buffer is a local which is never accessed):

-----

```
extern "C" char __declspec(dllexport) TestProc(char voie,double * pValeurs,
int
*nbValeur)
{
// UCHAR *buffer = new UCHAR[ 120000 ]; // [100 /* 120000 */];
UCHAR buffer[ 120000 ];
long sizeBuf=0;

DEBUGMSG( 1, ( TEXT( "In TestProc\r\n" ) ) );

// Do something here.
TCHAR s[ 1024 ];
_sprintf( s, _T( "voie = %d, pValeurs = 0x%x, nbValeur = %d, *nbValeur =
%d" ),
(int)voie, (unsigned)pValeurs, (unsigned)nbValeur, *nbValeur );
```

```
DEBUGMSG( 1, ( TEXT( "In TestProc, calling MessageBox\r\n" ) ) );  
  
MessageBox( NULL, s, _T( "TestProc" ), MB_OK );  
  
DEBUGMSG( 1, ( TEXT( "In TestProc, leaving\r\n" ) ) );  
  
// delete [] buffer;  
  
return voie;  
}
```

-----

When the buffer is dynamically allocated (same size and all), the code works fine.

And here's the managed code in VB which calls it:

-----

```
Public Declare Function TestProc Lib "TestProc.dll" (ByVal voie As Byte,  
ByVal pValeurs() As Double, ByVal nbValeur As Integer) As Byte
```

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Button1.Click  
    Dim pValeurs(256) As Double  
    Dim taille As Integer = 256  
    TestProc(4, pValeurs, taille)  
End Sub
```

-----

Paul T.

""Ilya Tumanov [MS]"" <ilyatum@online.microsoft.com> wrote in message  
news:bTOS9Dx1EHA.768@cpmsftngxa10.phx.gbl...

> I don't think it's related to VB and pages. That seems to be a very common

> bug in your native code called "buffer overflow".

> Here's an example:

> UCHAR buffer[10];

> ...

> bufferr[100] = 0; // Oops.

> memcpy( buffer, NULL, 100); // Oops and... oops.

>

> Most likely memcpy() in ReceiveData\_GetLast() copies more than 120000

> bytes for some reason or uses invalid pointer(s).

> Add some verification code to ReceiveData\_GetLast() to verify array

> boundaries/pointers and pop up an assert in case of a problem.

> That would help you figure out what's wrong. Also, it's a good idea to add

> some check and return an error if data appears to be incorrect.

>

```
> You should pass buffer size to ReceiveData_GetLast() so it won't copy too  
> much:  
>  
> UCHAR buffer[120000];  
> ..  
> long sizeBuf=sizeof(buffer);  
> ..  
>  
> In ReceiveData_GetLast():  
>  
> size = ... ; // Where it's coming from? Can it be wrong?  
> source = ... ; // Where it's coming from? Can it be wrong?  
>  
> ASSERT (buffer != NULL);  
> ASSERT (source != NULL);  
> ASSERT (sizeBuf > size);  
>  
> // consider adding check and return an error instead of crashing. ..  
> memcpy (buffer, source, size);  
> Best regards,  
>  
> Ilya  
>  
> This posting is provided "AS IS" with no warranties, and confers no  
> rights.  
>
```