

Re: operator new[] fails in DLL when being called from within VB6 process

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.kernel/2005-10/msg00376.htm>

- *From:* "Skywing" <skywing_NO_SPAM_@xxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 22 Oct 2005 10:21:34 -0400
-

The reason why it is forbidden is that depending on how the loader builds its initialization graph, it may work one day, and the next day you'll start having weird and hard to diagnose failures because of situations like your DllMain being called before ole32's DllMain.

And, of course, there are probably things CoInitialize does internally that are also going to fail on some occasions in other strange and hard to debug ways.

Stay away from doing things in DllMain if there is any remotely feasible alternative and you'll save yourself many hours of debugging "impossible" bugs that you probably won't catch in QA.

"Jo Siffert" <jo.siffert@xxxxxxx> wrote in message
<news:%23JvfIct1FHA.3560@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

> Hi,

>

> thanks for your answer. However, I noticed that the problem causing line
> of code was another one – not the memory allocation. So I have solved the
> problem now.

> However, I still do not know why VS.Net showed me a completely wrong stack
> when the exception was raised though all symbols should have been
> correct – but this is another topic.

>

> But btw – calling CoInitialize() in DllMain() is actually very forbidden
> ;) Though it works (I once used it by myself), you are actually only
> allowed to call functions in kernel32.dll.

>

> Thanks,

> Jo

>

> Hector Santos wrote:

>

>> In lieu of showing code, you might need to call CoInitialize() or
>> CoInitializeEx() when the DLL is first loaded.

>>

>> I can't remember the details, and our DLL is a complex multi-threaded RPC

Re: operator new[] fails in DLL when being called from within VB6 process

>> client/server wire handler for client applications, but I recall running
>> into similar memory related GPFs when 3rd party VB6 applications were
>> importing our standard client/server WINAPI based dlls. Every native
>> language applications (C/C++, Delphi, etc) worked fine except VB6-based
>> applets.
>>
>> You would call CoInitialize() in DLL_PROCESS_ATTACH and you would call
>> CoUninitialize() in DLL_PROCESS_DETACH.
>>
>> PS: You didn't mention it, but if your DLL uses threads, you definitely
>> need
>> to call CoInitialize().
>>
>> --
>> Hector Santos, Santronics Software, Inc.
>> <http://www.santronics.com>
>>
>>
>> "Jo Siffert" <jo.siffert@xxxxxxx> wrote in message
>> news:eVdyCYm1FHA.1028@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>
>>>Hi,
>>>
>>>I have written a DLL in C/C++ which is used by a VB6 program. Everything
>>>works fine unless the DLL uses the operator new[] – e.g.
>>>
>>> m_pBuffer = new CHAR[dwApproxPayloadLen];
>>>
>>>then an Access Violation Exception is thrown within ntdll!_RtlAllocHeap.
>>>
>>>Calling
>>> CFoo *f = new Cfoo()
>>>works just fine – only arrays will not work.
>>>
>>>The same code works just fine when being called from a C/C++ program.
>>>
>>>Is there something special about calling CRT functions when my lib is
>>>loaded into a VB6 process? Maybe VB6 does not initialize the CRT as it
>>>does not use it or something like that?
>>>
>>>Thanks in advance,
>>> Jo
>>>

• **References:**

◆ **[operator new\[\] fails in DLL when being called from within VB6 process](#)**

◇ From: Jo Siffert

Re: operator new[] fails in DLL when being called from within VB6 process

◆ **Re: operator new[] fails in DLL when being called from within VB6 process**

◇ From: Hector Santos

◆ **Re: operator new[] fails in DLL when being called from within VB6 process**

◇ From: Jo Siffert

- Prev by Date: **Re: "Manually" load an Exe**
- Next by Date: **Re: operator new[] fails in DLL when being called from within VB6 process**
- Previous by thread: **Re: operator new[] fails in DLL when being called from within VB6 process**
- Next by thread: **Re: operator new[] fails in DLL when being called from within VB6 process**
- Index(es):
 - ◆ **Date**
 - ◆ **Thread**