

# Re: COM Singleton object and static members lifetime

---

*Source:* <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.atl/2007-01/msg00174.html>

---

- *From:* "Alexander Nickolov" <[agnickolov@xxxxxxxx](mailto:agnickolov@xxxxxxxx)>
  - *Date:* Mon, 15 Jan 2007 09:55:38 -0800
- 

You do realize that COM singletons are always purely optional? You never need to have a COM singleton, it's just a convenience of not writing a flyweight object for your internal C++ singleton object.

--

=====  
Alexander Nickolov  
Microsoft MVP [VC], MCS D  
email: [agnickolov@xxxxxxxx](mailto:agnickolov@xxxxxxxx)  
MVP VC FAQ: <http://vcfaq.mvps.org>  
=====

<[joproulx@xxxxxxxxxxxx](mailto:joproulx@xxxxxxxxxxxx)> wrote in message  
[news:1168873581.566666.320800@xx](mailto:news:1168873581.566666.320800@xx)

Thanks for the reply,  
Actually, the singleton I am referring to in my post is indeed a COM singleton. You are mentioning that COM singletons cannot be hosted in DLL, can you explain a bit more on that? From what I have read so far, implementing singletons in a DLL can be problematic (threading models...), but not impossible as long as the singleton is accessed in the same process. In fact, this was working fine before we modified our code to compile with VS2005 and ATL 8.0.

From what I understand, the COM singleton gets destroyed when its Class

Factory is destroyed, which is when the DLL unloads. Do you know if a static member can be destroyed before this actually happens?

Modifying this design impacts a lot of code so I just want to be sure that it is really necessary before doing anything.

See reference:  
<http://www.kbalertz.com/201321/Alternative.Implementation.Singleton.aspx>

Re: COM Singleton object and static members lifetime

Best regards,  
Jonathan

Alexander Nickolov wrote:

Static C++ members have the same lifetime as global objects. And global objects are destroyed in random order as far as you are concerned – you should never allow one global object to depend on another global object in its destructor. So what happens in your case is your static members are destroyed before your singleton (e.g. global object). Then you try to access them in that object's destructor and you crash.

Since this is a singleton, why would you want to use static members in the first place? Make them regular members and access them through the singleton instance instead. Note I'm assuming your singleton is a regular C++ object. If you've attempted a COM singleton, now is the time to get rid of it – COM singletons cannot be hosted in DLLs.

--

=====  
Alexander Nickolov  
Microsoft MVP [VC], MCSO  
email: agnickolov@xxxxxxxx  
MVP VC FAQ: <http://vcfaq.mvps.org>  
=====

<joproulx@xxxxxxxxxxxx> wrote in message  
[news:1168637760.624713.16700@xx](mailto:news:1168637760.624713.16700@xx)

Hello everybody  
Here is a question about COM singleton object and static members lifetime.

The problem:  
I have a COM Singleton object in a DLL that is instantiated in a EXE application. When I close the EXE app, the DLL is unloaded and a call to FinalRelease() function is executed for a COM Singleton object before it is destroyed . The problem is that when the code tries to access some static members during this time, the application crashes.

The question:

## Re: COM Singleton object and static members lifetime

What is the lifetime of the static members in this case? Can we assume that static members are still valid when the COM singleton object's FinalRelease() function is called?

More explanation:

We were using a previous version of ATL before and it was working fine. But the problem appeared I think when we ported the code to VS2005 with ATL 8.0.

Here is a description of what is happening. You can see the stack below this description.

1) OmniDir.exe is the EXE application. You can see in the stack that it is closing, then some calls are made in msvcr80d.dll and other DLLs.

2) After that, my DLL GxResource.dll is unloaded when DllMainCRTStartup() is called.

3) This eventually calls FinalRelease() on my COM singleton object CResource before it is destroyed

4) The CDisplayHelper::Terminate() static function is finally called and this function tries to access static members which in turn crash the application because these members are invalid at this point.

```
CDisplayHelper::Terminate()
CAppResources::Terminate()
COMnicastAppResources::Terminate()
CResources::DestroyResources()
CResources::FinalRelease()
ATL::CComObjectCached<CResources>::~~CComObjectCached<CResources>()
ATL::CComObjectCached<CResources>::~`scalar deleting
destructor'()
TL::CComObjectCached<CResources>::Release()
ATL::CComPtrBase<IUnknown>::~~CComPtrBase<IUnknown>()
ATL::CComPtr<IUnknown>::~~CComPtr<IUnknown>()
ATL::CComClassFactorySingleton<CResources>::~~CComClassFactorySingleton<CResource
ATL::CComObjectCached<ATL::CComClassFactorySingleton<CResources>
```

Re: COM Singleton object and static members lifetime

```
::~CComObjectCached<ATL::CComClassFactorySingleton<CResources>>  
>()
```

```
ATL::CComObjectCached<ATL::CComClassFactorySingleton<CResources>
```

```
::`scalar deleting destructor'()
```

```
ATL::CComObjectCached<ATL::CComClassFactorySingleton<CResources>
```

```
::Release()
```

```
ATL::CAtlComModule::Term()
```

```
ATL::CAtlComModule::~CAtlComModule()
```

```
ATL::`dynamic atexit destructor for '_AtlComModule'()
```

```
_CRT_INIT(void * hDllHandle=0x07710000, [...])
```

```
__DllMainCRTStartup(void * hDllHandle=0x07710000,  
[...])
```

```
__DllMainCRTStartup(void * hDllHandle=0x07710000, [...])
```

```
[...]
```

```
msvcr80d.dll!__crtCorExitProcess(int status=0)
```

```
msvcr80d.dll!__crtExitProcess(int status=0)
```

```
msvcr80d.dll!doexit(int code=0, int quick=0, int retcaller=0)
```

```
msvcr80d.dll!exit(int code=0)
```

```
OmniDir.exe!__tmainCRTStartup()
```

```
OmniDir.exe!wWinMainCRTStartup()
```

Any idea on what is happening here?

Best regards,

Jonathan