

DLL2 should implement interface defined in DLL1 – how?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.atl/2007-07/msg00048.html>

- *From:* Axel Bock <axel.bock.news@xxxxxxxxxxxxxxxx>
 - *Date:* Tue, 10 Jul 2007 04:39:28 -0700
-

Hi list,

I have this VERY annoying thing at my hands. I don't know COM too much, so please forgive me if that's something every noob should know.

So.

I have two DLL's with COM components. DLL one defines an interface IMyInterface. I know want to have a 2nd DLL to have a class which implements that interface, too.

The IDL of the second DLL looks like this:

```
[
object,
uuid(982F014C-579C-464E-BA9E-584419041811),
dual,
nonextensible,
helpstring("IMyOtherInterface-Schnittstelle"),
pointer_default(unique)
]
interface IMyOtherInterface : IDispatch{
};

[
uuid(A10B957D-305F-401A-B12F-58EE7D192F66),
version(1.0),
]
library MySecondDLLLibrary
{
importlib("stdole2.tlb");
importlib("../myfirstproject/Debug/firstDLL.tlb");

[
uuid(0AC402BF-7310-4C56-8E29-48EAF9D5BA76),
helpstring("MyOtherInterface Class")
```

DLL2 should implement interface defined in DLL1 – how?

```
]
coclass MyOtherInterface
{
[default] interface IMyOtherInterface;
interface IMyInterface;
};
};
```

Now that compiles just fine. I also checked with some MS DLLs in the Windows directory, they look just the same.

The problem is now that I don't know how to do this on C++ level. What I did until now was:

- I added the IDL files from the IMyInterface to the 2nd project
- I compiled the 2nd project without any modifications – works.
- I checked the code, the code looks like this:

```
// ADDED BY ME
#include "IMyInterfaceProject.idl"
// THE REST IS WIZARD CODE

// CMyOtherInterface

class ATL_NO_VTABLE CMyOtherInterface :
public CComObjectRootEx<CComSingleThreadModel>,
// TRIED TO ADD THE NEXT LINE
// public IMyInterface,
public CComCoClass<CMyOtherInterface,
&CLSID_MyOtherInterface>,
public IDispatchImpl<IMyOtherInterface,
&IID_IMyOtherInterface, &LIBID_Icw_Sdk_Module_Configuration_C, /
*wMajor =*/ 1, /*wMinor =*/ 0>
{
public:
CMyOtherInterface()
{
}
}

DECLARE_REGISTRY_RESOURCEID(IDR_MyOtherInterface)

DECLARE_NOT_AGGREGATABLE(CMyOtherInterface)

BEGIN_COM_MAP(CMyOtherInterface)
COM_INTERFACE_ENTRY(IMyOtherInterface)
// TRIED TO ADD THE NEXT LINE
// COM_INTERFACE_ENTRY(IMyInterface)
COM_INTERFACE_ENTRY(IDispatch)
END_COM_MAP()
```

DLL2 should implement interface defined in DLL1 – how?

DLL2 should implement interface defined in DLL1 – how?

```
DECLARE_PROTECT_FINAL_CONSTRUCT()

HRESULT FinalConstruct()
{
return S_OK;
}

void FinalRelease()
{
}

public:
};

OBJECT_ENTRY_AUTO(__uuidof(MyOtherInterface),
CMOtherInterface)
```

Well. This is all fine, VS-wizard-generated code. But I missed the references to `IMyInterface`, so I tried including them into the header (the commented out lines). I also found an article in the internet saying exactly that: <http://vcfaq.mvps.org/com/7.htm> .

Alas as soon as I include references to `IMyInterface` in the inheritance chain, I get the following compiler error:
c:\.....\myotherinterface.h (36) : error C2594: 'static_cast' : ambiguous conversions from 'CMOtherInterface::_ComMapClass *' to 'IDispatch *'

I CAN import the IDL files into the 2nd project, but if I do that I cannot use both at the same time from a third one – both will export the `IMyInterface` definition, and the stub compiler – rightfully – complains about redefinition, and I guess this is not the right way to go. A look with `OLEView` at for example `SRCHADM.DLL` and `MSSITLB.IDL` shows that this HAS to be possible :-). But I only don't know how ...

Any ideas? Anyone? :-)? I hope this was enough information to get the idea across.

Greetings & thanks,
Axel.

.

DLL2 should implement interface defined in DLL1 – how?