



Developer Network

[MSDN subscriptions](#)[Get tools](#)[Sign in](#)[Technologies](#)[Downloads](#)[Programs](#)[Community](#)[Documentation](#)[Samples](#)[Follow us](#)[Collapse All](#)[Export \(0\)](#)[Print](#)[MSDN Library](#)[Development Tools and Languages](#)[Visual Studio 2013](#)[Visual C++](#)[MFC and ATL](#)[ATL COM Desktop Components](#)[Concepts](#)[Dual Interfaces and ATL](#)[Implementing a Dual Interface](#)**Multiple Dual Interfaces**[nonextensible Attribute](#)[Dual Interfaces and Events](#)

Multiple Dual Interfaces

Visual Studio 2013 [Other Versions](#) ▼

You may want to combine the advantages of a dual interface (that is, the flexibility of both vtable and late binding, thus making the class available to scripting languages as well as C++) with the techniques of multiple inheritance.

Although it is possible to expose multiple dual interfaces on a single COM object, it is not recommended. If there are multiple dual interfaces, there must be only one **IDispatch** interface exposed. The techniques available to ensure that this is the case carry penalties such as loss of function or increased code complexity. The developer considering this approach should carefully weigh the advantages and disadvantages.

Exposing a Single IDispatch Interface

It is possible to expose multiple dual interfaces on a single object by deriving from two or more specializations of **IDispatchImpl**. However, if you allow clients to query for the **IDispatch** interface, you will need to use the [COM_INTERFACE_ENTRY2](#) macro (or [COM_INTERFACE_ENTRY_IID](#)) to specify which base class to use for the implementation of **IDispatch**.

C++

```
COM_INTERFACE_ENTRY2(IDispatch, IMyDualInterface)
```

Because only one **IDispatch** interface is exposed, clients that can only access your objects through the **IDispatch** interface will not be able to access the methods or properties in any other interface.

Combining Multiple Dual Interfaces into a Single Implementation of IDispatch

ATL does not provide any support for combining multiple dual interfaces into a single implementation of **IDispatch**. However, there are several known approaches to manually combining the interfaces, such as creating a templated class that contains a union of the separate **IDispatch** interfaces, creating a new object to perform the **QueryInterface** function, or using a typeid-based implementation of nested objects to create the **IDispatch** interface.

These approaches have problems with potential namespace collisions, as well as code complexity and maintainability. It is not recommended that you create multiple dual interfaces.

See Also

Concepts

[Dual Interfaces and ATL](#)

Was this page helpful?

Your feedback about this content is important.
Let us know what you think.

Yes

No

Have a suggestion to improve MSDN Library?

Visit our [UserVoice Page](#) to submit
and vote on ideas!

Make a suggestion

Dev centers

[Windows](#)

[Office](#)

[Visual Studio](#)

[Nokia](#)

[Microsoft Azure](#)

[More...](#)

Learning resources

[Microsoft Virtual Academy](#)

[Channel 9](#)

[Interoperability Bridges](#)

[MSDN Magazine](#)

Programs

[BizSpark \(for startups\)](#)

[DreamSpark](#)

[Imagine Cup](#)

Community

[Forums](#)

[Blogs](#)

[Codeplex](#)

Support

[Self support](#)

[Other support options](#)

[United States \(English\)](#)

[Newsletter](#)

[Privacy & cookies](#)

[Terms of use](#)

[Trademarks](#)

© 2014 Microsoft