Visual Studio 2010 - Visual C#

**/link (C# Compiler Options)**

Causes the compiler to make COM type information in the specified assemblies available to the project that you are currently compiling.

/link:fileList

// -or-

/l:fileList

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifArguments

*fileList*

Required. Comma-delimited list of assembly file names. If the file name contains a space, enclose the name in quotation marks.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifRemarks

The **/link** option enables you to deploy an application that has embedded type information. The application can then use types in a runtime assembly that implement the embedded type information without requiring a reference to the runtime assembly. If various versions of the runtime assembly are published, the application that contains the embedded type information can work with the various versions without having to be recompiled. For an example, see [Walkthrough: Embedding Types from Managed Assemblies (C# and Visual Basic)](http://msdn.microsoft.com/en-us/library/dd409610.aspx).

Using the **/link** option is especially useful when you are working with COM interop. You can embed COM types so that your application no longer requires a primary interop assembly (PIA) on the target computer. The **/link** option instructs the compiler to embed the COM type information from the referenced interop assembly into the resulting compiled code. The COM type is identified by the CLSID (GUID) value. As a result, your application can run on a target computer that has installed the same COM types with the same CLSID values. Applications that automate Microsoft Office are a good example. Because applications like Office usually keep the same CLSID value across different versions, your application can use the referenced COM types as long as .NET Framework 4 or later is installed on the target computer and your application uses methods, properties, or events that are included in the referenced COM types.

The **/link** option embeds only interfaces, structures, and delegates. Embedding COM classes is not supported.

|  |
| --- |
| **Description: NoteNote** |
| When you create an instance of an embedded COM type in your code, you must create the instance by using the appropriate interface. Attempting to create an instance of an embedded COM type by using the CoClass causes an error. |

To set the **/link** option in Visual Studio, add an assembly reference and set the **Embed Interop Types** property to **true**. The default for the **Embed Interop Types** property is **false**.

If you link to a COM assembly (Assembly A) which itself references another COM assembly (Assembly B), you also have to link to Assembly B if either of the following is true:

* A type from Assembly A inherits from a type or implements an interface from Assembly B.
* A field, property, event, or method that has a return type or parameter type from Assembly B is invoked.

Like the [/reference](http://msdn.microsoft.com/en-us/library/yabyz3h4.aspx) compiler option, the **/link** compiler option uses the Csc.rsp response file, which references frequently used .NET Framework assemblies. Use the [/noconfig](http://msdn.microsoft.com/en-us/library/8hww4s6c.aspx) compiler option if you do not want the compiler to use the Csc.rsp file.

The short form of **/link** is **/l**.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifGenerics and Embedded Types

The following sections describe the limitations on using generic types in applications that embed interop types.

**Generic Interfaces**

Generic interfaces that are embedded from an interop assembly cannot be used. This is shown in the following example.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl00_code');" \o "Copy Code)

// The following code causes an error if ISampleInterface is an embedded interop type.

ISampleInterface<SampleType> sample;

**Types That Have Generic Parameters**

Types that have a generic parameter whose type is embedded from an interop assembly cannot be used if that type is from an external assembly. This restriction does not apply to interfaces. For example, consider the [Range](http://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel.range.aspx) interface that is defined in the [Microsoft.Office.Interop.Excel](http://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel.aspx) assembly. If a library embeds interop types from the [Microsoft.Office.Interop.Excel](http://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel.aspx) assembly and exposes a method that returns a generic type that has a parameter whose type is the [Range](http://msdn.microsoft.com/en-us/library/microsoft.office.interop.excel.range.aspx) interface, that method must return a generic interface, as shown in the following code example.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl05_code');" \o "Copy Code)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Microsoft.Office.Interop.Excel;

public class Utility

{

// The following code causes an error when called by a client assembly.

public List<Range> GetRange1() {

...

}

// The following code is valid for calls from a client assembly.

public IList<Range> GetRange2() {

...

}

}

In the following example, client code can call the method that returns the [IList](http://msdn.microsoft.com/en-us/library/system.collections.ilist.aspx) generic interface without error.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl29_ctl00_ctl07_code');" \o "Copy Code)

public class Client

{

public void Main()

{

Utility util = new Utility();

// The following code causes an error.

List<Range> rangeList1 = util.GetRange1();

// The following code is valid.

List<Range> rangeList2 = (List<Range>)util.GetRange2();

}

}

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExample

The following code compiles source file OfficeApp.cs and reference assemblies from COMData1.dll and COMData2.dll to produce OfficeApp.exe.

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl30_ctl00_ctl00_code');" \o "Copy Code)

csc /link:COMData1.dll,COMData2.dll /out:OfficeApp.exe OfficeApp.cs