This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework 4

**How to: Sign an Assembly with a Strong Name**

The Windows Software Development Kit (SDK) provides several ways to sign an assembly with a strong name:

* Using the [Assembly Linker (Al.exe)](http://msdn.microsoft.com/en-us/library/c405shex.aspx) provided by the Windows SDK.
* Using assembly attributes to insert the strong name information in your code. You can use either the [AssemblyKeyFileAttribute](http://msdn.microsoft.com/en-us/library/system.reflection.assemblykeyfileattribute.aspx) or the [AssemblyKeyNameAttribute](http://msdn.microsoft.com/en-us/library/system.reflection.assemblykeynameattribute.aspx), depending on where the key file to be used is located.

|  |
| --- |
| **Description: NoteNote** |
| In the .NET Framework version 2.0, some compilers issue warning messages when attributes are used. |

* Using compiler options such **/keyfile** or **/delaysign** in C# and Visual Basic, or the **/KEYFILE** or **/DELAYSIGN** linker option in C++. (For information on delay signing, see [Delay Signing an Assembly](http://msdn.microsoft.com/en-us/library/t07a3dye.aspx).)

|  |
| --- |
| **Description: NoteNote** |
| In Visual Studio 2005, the development environment provides tools for signing assemblies. See [Managing Assembly and Manifest Signing](http://msdn.microsoft.com/en-us/library/ms247066.aspx) and [Signing Page, Project Designer](http://msdn.microsoft.com/en-us/library/0k50fs3b.aspx). |

You must have a cryptographic key pair to sign an assembly with a strong name. For more information about creating a key pair, see [How to: Create a Public/Private Key Pair](http://msdn.microsoft.com/en-us/library/6f05ezxy.aspx).

**To create and sign an assembly with a strong name using the Assembly Linker**

* At the command prompt, type the following command:

**al** **/out:**<*assembly name*> *<module name>* **/keyfile:**<*file name*>

In this command, *assembly name* is the name of the assembly to sign with a strong name, *module name* is the name of the code module used to create the assembly, and *file name* is the name of the container or file that contains the key pair.

The following example signs the assembly MyAssembly.dll with a strong name using the key file sgKey.snk.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl10_code');" \o "Copy Code)

al /out:MyAssembly.dll MyModule.netmodule /keyfile:sgKey.snk

**To sign an assembly with a strong name using attributes**

* In a code module, add the **AssemblyKeyFileAttribute** or the **AssemblyKeyNameAttribute**, specifying the name of the file or container that contains the key pair to use when signing the assembly with a strong name.

The following code example uses the **AssemblyKeyFileAttribute** with a key file called keyfile.snk, located in the directory where the assembly is compiled.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl11_code');" \o "Copy Code)

<Assembly:AssemblyKeyFileAttribute("keyfile.snk")>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl12_code');" \o "Copy Code)

[assembly:AssemblyKeyFileAttribute("keyfile.snk")]

Visual C++

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl13_code');" \o "Copy Code)

[assembly:AssemblyKeyFileAttribute("keyfile.snk")];

|  |
| --- |
| **Description: NoteNote** |
| In development environments such as Visual Studio, the assembly might not be compiled in the project directory. For example, some versions of Visual Studio compile C# projects in a bin\Debug subdirectory. In that case, the path in the code example would be "..\\..\\keyfile.snk". In Visual Studio 2005 the key file for C# can be specified in project settings. |

You can also delay sign an assembly when compiling. For more information, see [Delay Signing an Assembly](http://msdn.microsoft.com/en-us/library/t07a3dye.aspx).

When signing an assembly with a strong name, the [Assembly Linker (Al.exe)](http://msdn.microsoft.com/en-us/library/c405shex.aspx) looks for the key file relative to the current directory and to the output directory. When using command-line compilers, you can simply copy the key to the current directory containing your code modules.

Note:  
**csc /t:module Myclass.cs**   
gives you a Myclass.netmodule

Then use  
**al /out:Myclass.dll Myclass.netmodule /keyfile:sgKey.snk** (assuming you don't have Myclass.dll previously)  
to generate the signed Myclass.dll (which references Myclass.netmodule, kind of inconvenience: you need both together anywhere when other classes reference your Myclass.dll)