**Delay Signing an Assembly**

**.NET Framework 4.5**

An organization can have a closely guarded key pair that developers do not have access to on a daily basis. The public key is often available, but access to the private key is restricted to only a few individuals. When developing assemblies with strong names, each assembly that references the strong-named target assembly contains the token of the public key used to give the target assembly a strong name. This requires that the public key be available during the development process.

You can use delayed or partial signing at build time to reserve space in the portable executable (PE) file for the strong name signature, but defer the actual signing until some later stage (typically just before shipping the assembly).

The following steps outline the process to delay sign an assembly:

1. Obtain the public key portion of the key pair from the organization that will do the eventual signing. Typically this key is in the form of an .snk file, which can be created using the [Strong Name tool (Sn.exe)](http://msdn.microsoft.com/en-us/library/k5b5tt23.aspx) provided by the Windows Software Development Kit (SDK).
2. Annotate the source code for the assembly with two custom attributes from [System.Reflection](http://msdn.microsoft.com/en-us/library/system.reflection.aspx):
   * [AssemblyKeyFileAttribute](http://msdn.microsoft.com/en-us/library/system.reflection.assemblykeyfileattribute.aspx), which passes the name of the file containing the public key as a parameter to its constructor.
   * [AssemblyDelaySignAttribute](http://msdn.microsoft.com/en-us/library/system.reflection.assemblydelaysignattribute.aspx), which indicates that delay signing is being used by passing **true** as a parameter to its constructor. For example:

**C#**

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

[assembly:AssemblyDelaySignAttribute(true)]

1. The compiler inserts the public key into the assembly manifest and reserves space in the PE file for the full strong name signature. The real public key must be stored while the assembly is built so that other assemblies that reference this assembly can obtain the key to store in their own assembly reference.
2. Because the assembly does not have a valid strong name signature, the verification of that signature must be turned off. You can do this by using the **–Vr** option with the Strong Name tool.

The following example turns off verification for an assembly called myAssembly.dll.

sn –Vr myAssembly.dll

To turn off verification on platforms where you can’t run the Strong Name tool, such as Advanced RISC Machine (ARM) microprocessors, use the **–Vk** option to create a registry file. Import the registry file into the registry on the computer where you want to turn off verification. The following example creates a registry file for myAssembly.dll.

sn –Vk myRegFile.reg myAssembly.dll

With either the **–Vr** or **–Vk** option, you can optionally include an .snk file for test key signing.

|  |
| --- |
| **Description: Caution noteCaution** |
| Use the **-Vr** or **–Vk** option only during development. Adding an assembly to the skip verification list creates a security vulnerability. A malicious assembly could use the fully specified assembly name (assembly name, version, culture, and public key token) of the assembly added to the skip verification list to fake its identity. This would allow the malicious assembly to also skip verification. |
| **Description: NoteNote** |
| If you use delay signing during development with Visual Studio on a 64-bit computer, and you compile an assembly for **Any CPU**, you might have to apply the **-Vr** option twice. (In Visual Studio, **Any CPU** is a value of the **Platform Target** build property; when you compile from the command line, it is the default.) To run your application from the command line or from File Explorer, use the 64-bit version of the [Sn.exe (Strong Name Tool)](http://msdn.microsoft.com/en-us/library/k5b5tt23.aspx) to apply the **-Vr** option to the assembly. To load the assembly into Visual Studio at design time (for example, if the assembly contains components that are used by other assemblies in your application), use the 32-bit version of the strong-name tool. This is because the just-in-time (JIT) compiler compiles the assembly to 64-bit native code when the assembly is run from the command line, and to 32-bit native code when the assembly is loaded into the design-time environment. |

1. Later, usually just before shipping, you submit the assembly to your organization's signing authority for the actual strong name signing using the **–R** option with the Strong Name tool.

The following example signs an assembly called myAssembly.dll with a strong name using the sgKey.snk key pair.

sn -R myAssembly.dll sgKey.snk