# Silverlight and .NET Compact Framework Support

Ice for .NET includes support for Silverlight and the .NET Compact Framework (.NET CF).

On this page:



## Using Ice for Silverlight and .NET CF

These versions of Ice differ from Ice for .NET in several ways, therefore you must re-compile Ice to target Silverlight or .NET CF. The Ice installer for Windows includes the Silverlight and .NET CF versions of the Ice run time in *`install-dir`*`\bin\sl\Ice.dll` and *`install-dir`*`\bin\cf\Ice.dll`, respectively. The Ice Visual Studio Add-in automatically selects the appropriate DLL for your project. To build Ice for .NET CF in a source distribution, enable `COMPACT` in `cs\config\Make.rules.mak.cs`; to build Ice for Silverlight, enable `SILVERLIGHT` in the same file.

## Limitations of Ice for Silverlight and .NET CF

This table lists the Ice for .NET features that may not be available when using Silverlight or .NET CF:

| Feature | Silverlight | .NET CF |
|---|---|---|
| Protocol compression | No | No |
| Serializable objects | No | No |
| IceSSL transport plug-in | No | No |
| `ICE_CONFIG` environment variable | No | No |
| `Ice.Application` class | No | Yes (without signal support) |
| `Glacier2.Application` class | No | Yes |
| Thread priorities | No | Yes |
| Server-side support | No | Yes |
| Bi-directional callbacks | Yes | Yes |
| Loading properties from Windows registry | No | Yes |
| `Ice.StdOut` and `Ice.StdErr` properties | No | Yes |
| `Ice.PrintProcessId` property | No | Yes |
| `Ice.LogFile` property | No | Yes |
| `Ice.TCP.SndSize` and `Ice.TCP.RcvSize` properties | Yes | No |
| Multi-homed DNS addresses | No | Yes |
| Dynamic loading of class and exception factories | No | No |

With respect to multi-homed DNS addresses, the Silverlight API never returns more than one IP address when performing a DNS lookup, even for a multi-homed host name that is associated with multiple IP addresses. As a result, the Ice for Silverlight run time cannot attempt to transparently establish a connection to all of the IP addresses associated with a multi-homed host name.

## Managing Factory Assemblies

When receiving a Slice user exception or a concrete Slice object-by-value, the Ice run time must be able to dynamically translate the encoded Slice type ID (such as `::MyModule::MyType`) into a .NET class name (such as `MyModule.MyType`), dynamically locate that class, and instantiate it. This is convenient for .NET applications because it requires no additional user configuration; at startup, the Ice for .NET run time recursively loads all dependent assemblies used by the program to ensure that any generated classes are available if necessary.

Neither Silverlight nor the Compact Framework allow a program to discover its dependent assemblies, so this strategy cannot work. Consequently, Ice adds the new configuration property `Ice.FactoryAssemblies` so that you can explicitly list any assemblies that contain the generated code for user exceptions or concrete classes. When searching for a class, Ice first checks in the assemblies specified by this property. If the type is not found, Ice automatically looks in the standard Ice assemblies (`Ice`, `Glacier2`, `IceBox`, `IceGrid`, `IcePatch2`, and `IceStorm`).

Note that the program itself is also considered an assembly. If you compiled the main program directly with Slice-generated code, your `Ice.FactoryAssemblies` property must include the program itself if the generated code includes user exceptions or concrete classes. For simple build scenarios in which all generated code is compiled directly into the executable, the following configuration setting is sufficient:

```
Ice.FactoryAssemblies=client
```

This example assumes the executable is named `client.exe`. On the other hand, if Slice-generated code is also compiled into a dependent assembly, your configuration might look like this instead:

```
Ice.FactoryAssemblies=client MyOtherAssembly
```

Failing to define `Ice.FactoryAssemblies` can cause the Ice run time in the receiver to raise `NoObjectFactoryException`, `UnmarshalOutOfBoundsException` or `UnknownUserException`. If you are experiencing any of these exceptions, verify that your assemblies are configured correctly.

## See Also

- Visual Studio Add-in
- `Ice.FactoryAssemblies`