

Alternate Property Stores

In addition to regular files, Ice also supports storing property settings in the Windows registry and Java resources.

On this page:

- [Loading Properties from the Windows Registry](#)
- [Loading Properties from Java Resources](#)

Loading Properties from the Windows Registry

You can use the Windows registry to store property settings. Property settings must be stored with a key underneath HKEY_LOCAL_MACHINE. To inform the Ice run time of this key, you must set the `Ice.Config` property to the key. For example:

```
$ client --Ice.Config=HKLM\MyCompany\MyApp
```

The Ice run time examines the value of `Ice.Config`; if that value begins with HKLM, the remainder of the property is taken to be a key to a number of string values. For the preceding example, the Ice run time looks for the key HKEY_LOCAL_MACHINE\MyCompany\MyApp. The string values stored under this key are used to initialize the properties.

The name of each string value is the name of the property (such as `Ice.Trace.Network`). Note that the value must be a string (even if the property setting is numeric). For example, to set `Ice.Trace.Network` to 3, you must store the string "3" as the value, not a binary or DWORD value.

String values in the registry can be regular strings (REG_SZ) or expandable strings (REG_EXPAND_SZ). Expandable strings allow you to include symbolic references to environment variables (such as %ICE_HOME%).



Depending on whether you use 32-bit or 64-bit binaries, you must set the registry keys in the corresponding 32-bit or 64-bit registry. See <http://support.microsoft.com/kb/305097> for more information.

Loading Properties from Java Resources

The Ice run time for Java supports the ability to load a configuration file as a class loader resource, which is especially useful for deploying an Ice application in a self-contained JAR file. For example, suppose we define `ICE_CONFIG` as shown below:

```
$ export ICE_CONFIG=app_config
```

During the creation of a property set (which often occurs implicitly when initializing a new communicator), Ice asks the Java run time to search the application's class path for a file named `app_config`. This file might reside in the same JAR file as the application's class files, or in a different JAR file in the class path, or it might be a regular file located in one of the directories in the class path. If Java is unable to locate the configuration file in the class path, Ice attempts to open the file in the local file system.

The class path resource always takes precedence over a regular file. In other words, if a class path resource and a regular file are both present with the same path name, Ice always loads the class path resource in preference to the regular file.

The path name for a class path resource uses a relative Unix-like format such as `subdir/myfile`. Java searches for the resource relative to each JAR file or subdirectory in an application's class path.

See Also

- [Using Configuration Files](#)