# **Using Configuration Files**

The ability to configure an application's properties externally provides a great deal of flexibility: you can use any combination of command-line options and configuration files to achieve the desired settings, all without having to modify your application. This page describes two ways of loading property settings from a file.

On this page:

- Prerequisites for Using Configuration Files
- The ICE\_CONFIG Environment Variable
- The Ice.Config Property

## Prerequisites for Using Configuration Files

The lce run time automatically loads a configuration file during the creation of a *property set*, which is an instance of the *lce::Properties* interface. Every communicator has its own property set from which it derives its configuration. If an application does not supply a property set when it calls *lce::initialize* (or the equivalent in other language mappings), the lce run time internally creates a *property set* for the new communicator.

Note however that Ice loads a configuration file automatically only when the application creates a property set using an argument vector. This occurs when the application passes an argument vector to create a property set explicitly, or when the application passes an argument vector to Ice:: initialize.

Both of the mechanisms described below can also retrieve property settings from additional sources.

### The ICE\_CONFIG Environment Variable

Ice automatically loads the contents of the configuration file named in the ICE\_CONFIG environment variable (assuming the prerequisites are met). For example:

```
$ export ICE_CONFIG=/usr/local/filesystem/config
$ ./server
```

This causes the server to read its property settings from the configuration file in /usr/local/filesystem/config.

If you use the ICE\_CONFIG environment variable together with command-line options for other properties, the settings on the command line override the settings in the configuration file. For example:

```
$ export ICE_CONFIG=/usr/local/filesystem/config
$ ./server --Ice.MessageSizeMax=4096
```

This sets the value of the Ice.MessageSizeMax property to 4096 regardless of any setting of this property in /usr/local/filesystem/config.

You can use multiple configuration files by specifying a list of configuration file names separated by commas. For example:

```
$ export ICE_CONFIG=/usr/local/filesystem/config,./config
$ ./server
```

This causes property settings to be retrieved from /usr/local/filesystem/config, followed by any settings in the file config in the current directory; settings in ./config override settings /usr/local/filesystem/config.

### The Ice.Config Property

The Ice.Config property has special meaning to the Ice run time: it determines the path name of a configuration file from which to read property settings. For example:

```
$ ./server --Ice.Config=/usr/local/filesystem/config
```

This causes property settings to be read from the configuration file in /usr/local/filesystem/config.

The --Ice.Config command-line option overrides any setting of the ICE\_CONFIG environment variable, that is, if the ICE\_CONFIG environment variable is set and you also use the --Ice.Config command-line option, the configuration file specified by the ICE\_CONFIG environment variable is ignored.

If you use the --Ice.Config command-line option together with settings for other properties, the settings on the command line override the settings in the configuration file. For example:

\$ ./server --Ice.Config=/usr/local/filesystem/config --Ice.MessageSizeMax=4096

This sets the value of the Ice.MessageSizeMax property to 4096 regardless of any setting of this property in /usr/local/filesystem/config . The placement of the --Ice.Config option on the command line has no influence on this precedence. For example, the following command is equivalent to the preceding one:

\$ ./server --Ice.MessageSizeMax=4096 --Ice.Config=/usr/local/filesystem/config

Settings of the Ice.Config property inside a configuration file are ignored, that is, you can set Ice.Config only on the command line.

If you use the --Ice.Config option more than once, only the last setting of the option is used and the preceding ones are ignored. For example:

\$ ./server --Ice.Config=file1 --Ice.Config=file2

#### This is equivalent to using:

\$ ./server --Ice.Config=file2

You can use multiple configuration files by specifying a list of configuration file names separated by commas. For example:

```
$ ./server --Ice.Config=/usr/local/filesystem/config,./config
```

This causes property settings to be retrieved from /usr/local/filesystem/config, followed by any settings in the file config in the current directory; settings in ./config override settings /usr/local/filesystem/config.

#### See Also

- Alternate Property Stores
- The Properties Interface