Using the Mac OS X Binary Distribution

This page provides important information for users of the Ice binary distribution for Mac OS X. You can obtain this distribution at the ZeroC web site.

On this page:

Overview of the Mac OS X binary distribution

The binary distribution of Ice for Mac OS X includes the following components:

- The Ice run time, including executables for the Ice services, and Slice files.
- Run time libraries for C++, Java, and Python. These libraries enable you to execute Ice applications.
- Tools and libraries for developing Ice applications.

The binary distribution was compiled on Mac OS X 10.6 using the default C++ compiler, GCC 4.2.1. The binaries in this distribution are fat binaries with support for both Intel 32-bit and Intel 64-bit architectures.

The Ice extension for Python included in this distribution requires Python 2.6.1, installed with Mac OS X 10.6.

The Ice extension for Python is not binary-compatible with Python 2.7, which is the default version in Mac OS 10.7. To use Ice for Python in this environment, you must either rebuild the extension from source for Python 2.7, or modify your environment to use Python 2.6 instead.

Setting up your Mac OS X environment to use Ice

After installing Ice, read the relevant language-specific sections below to learn how to configure your environment and start programming with Ice.

General requirements

In order to use Ice services and tools such as Slice translators, you need to add the location of the Ice binaries to your `PATH` as shown in the bash command below:

```
$ export PATH=<Ice installation directory>/bin:$PATH
```

The install name of Ice shared libraries in this distribution contain the `/opt/Ice-3.4/lib` path. In order to run Ice services and tools, you can do one of the following:

- Create a symbolic link `/opt/Ice-3.4` that points to your Ice installation:

  ```
  # ln -s <Ice installation directory> /opt/Ice-3.4
  ```

- Add the Ice `lib` directory to your `DYLD_LIBRARY_PATH` environment variable:

  ```
  $ export DYLD_LIBRARY_PATH=<Ice installation directory>/lib:$DYLD_LIBRARY_PATH
  ```

If you run applications that load the IceSSL plug-in (such as the Ice demos) or the IceStorm service, you need to set the `DYLD_LIBRARY_PATH` environment variable as shown above.

C++

When compiling Ice for C++ programs, you must pass the Ice include directory to the compiler with the `-I` option, and the Ice library directory with the `-L` option. Furthermore, a C++ program needs to link with at least `libIce` and `libIceUtil`, so a typical link command would look like this:

```
$ c++ -I <Ice installation directory>/include -o myprogram myprogram.o \
  -L<Ice installation directory>/lib -lIce -lIceUtil
```

Additional libraries are necessary if you are using an Ice service such as IceGrid or Glacier2.

To build fat binaries or binaries using an architecture that differs from the default architecture, you can specify the GCC `-arch` compiler flag. For example, use `-arch i386 -arch x86_64` to build Intel 32-bit and 64-bit fat binaries.

Java

To use Ice for Java, you must add `Ice.jar` to your `CLASSPATH`, as shown below:

```
$ export CLASSPATH=<Ice installation directory>/lib/Ice.jar:$CLASSPATH
```

If you intend to use Freeze for Java, you must include `Freeze.jar` in your `CLASSPATH` along with `Ice.jar`:

```
$ export CLASSPATH=<Ice installation directory>/lib/Freeze.jar:$CLASSPATH
```

Note that Freeze requires Berkeley DB. Freeze.jar contains a manifest that automatically loads the Berkeley DB classes (`db.jar`), which means you do not need to include this file in your `CLASSPATH` when executing a Freeze application. However the JVM does require that the directory containing Berkeley DB's native libraries be listed in `java.library.path`, therefore you must add this directory to your `DYLD_LIBRARY_PATH`.

When building a Java application that uses Freeze, you will need to add the Berkeley DB JAR file to your `CLASSPATH`:

```
$ export CLASSPATH=<Ice installation directory>/lib/db.jar:$CLASSPATH
```

Ice includes ant tasks for translating Slice to Java. The ant tasks allow `slice2java` and `slice2freezej` to be invoked from the ant build system. These tasks require one of the following:

- Specify the location of the Ice installation containing the translators with the `ice.home` property:

  ```
  ant -Dice.home=/home/bill/Ice-3.4.2
  ```

- Set the `ICE_HOME` environment variable to specify the location of the Ice installation containing the translators:

  ```
  $ export ICE_HOME=/home/bill/Ice-3.4.2
  ```

- If neither `ice.home` nor `ICE_HOME` is available, the ant tasks will simply invoke the translator without an absolute path, relying on the translators being in a directory in your `PATH` for successful execution.

Ice for Java supports protocol compression using the bzip2 classes included with ant. Compression is automatically enabled if these classes are present in your `CLASSPATH`. You can either add `ant.jar` to your `CLASSPATH`, or download only the bzip2 classes from:

http://www.kohsuke.org/bzip2/

Note that these classes are a pure Java implementation of the bzip2 algorithm and therefore add significant latency to Ice requests.

Eclipse Development

ZeroC has created a Slice2Java plug-in for Eclipse that automates the translation of your Slice files. If you use Eclipse, we strongly recommend using this plug-in for your own development.

The Slice2Java plug-in is required if you intend to build any of the Android projects included in the sample programs.

For installation instructions, please refer to the ZeroC web site. The manual provides more information about configuring the plug-in and using it in your projects.

Android

Ice requires Android 2.1 or later. Aside from that, there are no other special requirements for using Ice in an Android application. We strongly recommend installing our Slice2Java plug-in for Eclipse to automate the compilation of your Slice definitions.

Python

To use Ice for Python, the `PYTHONPATH` environment variable must be updated so that the interpreter can load the Ice extension and supporting Python files:

```
$ export PYTHONPATH=<Ice installation directory>/python:$PYTHONPATH
```

Starting the IceGrid Administrative Console on Mac OS X

A Java-based graphical tool for administering IceGrid applications is included in this distribution. The Java archive file is installed as

<Ice installation directory>`/lib/IceGridGUI.jar`

With a suitable Java environment, you can execute the application using the following command:

```
$ java -jar IceGridGUI.jar
```

Using the sample programs on Mac OS X

Sample programs are provided in a separate archive, which can be downloaded from the ZeroC web site.

Please refer to the `README.DEMOS` file included in that package for more information.

Third-party packages for Mac OS X

The binary distribution for Mac OS X includes the following third-party packages as separate binary libraries:

- Berkeley DB 4.8.30 (C/C++ and Java run time)
- Expat 2.0.1 (C run time)

The IceGrid and IceStorm SQL database plug-ins depend on the Qt 4.6.2 framework, which is not included in this distribution. You can download binaries for the Qt framework at the Nokia web site.