Using the Solaris Binary Distribution

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

On this page:

Overview of the Solaris binary distribution

The binary distribution of Ice for Solaris 11 on SPARC includes the following components:

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

The distribution contain executables and libraries in both 32-bit and 64-bit format. The 32-bit executables are in the `bin` directory and the 32-bit libraries are in the `lib` directory. The 64-bit executables are located in `bin/sparcv9`, and the 64-bit libraries are in `lib/64`.

The C++ binaries were created using Sun Studio 12.3 and its default standard library.

The following Solaris 11 binary packages are required to use this distribution:

- `compress/bzip2`
- `library/expat`
- `library/security/openssl`

Ice for Java requires J2SE 1.6.0 or later; Java binary packages can be obtained from Oracle. The IceGrid Admin graphical tool is included in the distribution but the metrics graph feature is not supported because JavaFX is not available on this platform.

If you are using Java 7, see this Solaris Java 7 issue.

Setting up your Solaris 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
```

To use the 64-bit version of these services and tools, add the `sparcv9` subdirectory to `PATH`, making sure it appears before `bin`:

```
$ export PATH=<Ice installation directory>/bin:$PATH
$ export PATH=<Ice installation directory>/bin/sparcv9:$PATH
```

Ice shared libraries and executables in this distribution contain `/opt/Ice-3.5/lib` as the embedded runpath (the runpath for 64-bit libraries and executables is `/opt/Ice-3.5/lib/64`). In order to run Ice services and tools, you can do one of the following:

- Create a symbolic link `/opt/Ice-3.5` that points to your Ice installation:

```
$ ln -s <Ice installation directory> /opt/Ice-3.5
```

- Add the Ice `lib` directory to your `LD_LIBRARY_PATH` and `LD_LIBRARY_PATH_64` environment variables. For example:

```
$ export LD_LIBRARY_PATH=<Ice installation directory>/lib:$LD_LIBRARY_PATH
$ export LD_LIBRARY_PATH_64=<Ice installation directory>/lib/64:$LD_LIBRARY_PATH_64
```

C++

When compiling Ice for C++ programs, you must pass a `-I` option specifying the Ice include directory, along with several other options:

```
CC -I <Ice installation directory>/include -c -mt +p -errtags -erroff=wvarhidenmem,wvarhidemem,notemsource -
m32 myprogram.cpp
```

Replace `-m32` with `-m64` to build 64-bit binaries.

A C++ program needs to link with at least `libIce` and `libIceUtil`, so a typical link command would look like this:

```
$ CC -mt -m32 -o myprogram myprogram.o -L<Ice installation directory>/lib -lIce -lIceUtil -lpthread
```

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

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` and `db.jar` in your `CLASSPATH` along with `Ice.jar`:

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

Classes for the other Ice services are provided in separate JAR files:

- `Glacier2.jar`
- `IceBox.jar`
- `IceGrid.jar`
- `IcePatch2.jar`
- `IceStorm.jar`

If your application uses any of these services, add the appropriate JAR files to your `CLASSPATH` as shown above.

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-3.5.1
```

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

```
$ export ICE_HOME=/home/bill/Ice-3.5.1
```

- 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.

Using the sample programs on Solaris

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

Third-party packages for Solaris

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

- Berkeley DB 5.3.21 (C/C++ and Java run time)