Building Ice for C++ on OS X

This page describes the Ice source distribution, including information about compiler requirements, third-party dependencies, and instructions for building and testing the distribution. If you prefer, you can download a binary package that contains pre-compiled libraries, executables, and everything else necessary to build Ice applications on OS X.

On this page:

C++ Build Requirements for OS X

Ice for C++ was extensively tested using the operating systems and compiler versions listed on our platforms page.

Third-party Libraries

Ice has dependencies on a number of third-party libraries:

- Berkeley DB 5.3
- expat 2.0
- OpenSSL 0.9.8 or later
- bzip2 1.0
- mcpp 2.7.2 (with patches)

Some of these packages may have been included with your system. For those packages that are not installed or have an older version than what is listed above, we recommend downloading the Ice third-party source archive. This archive contains the source distributions for each of the third-party dependencies, as well as required source patches and configuration instructions.

Compiling and Testing Ice for C++ on OS X

Extract the Ice archive in any directory you like (for example, in your home directory):

```
$ tar xvfz Ice-3.5.1.tar.gz
```

Change the working directory to `Ice-3.5.1/cpp`:

```
$ cd Ice-3.5.1/cpp
```

Edit `config/Make.rules` to establish your build configuration. The comments in the file provide more information. Pay particular attention to the variables that define the locations of the third-party libraries.

Now you're ready to build Ice:

```
$ make
```

This will build the Ice core libraries, services, tests and examples.

Python is required to run the test suite. After a successful build, you can run the tests as follows:

```
$ make test
```

This command is equivalent to:

```
$ python allTests.py
```

If everything worked out, you should see lots of "ok" messages. In case of a failure, the tests abort with "failed".

If you want to try out any of the demos, make sure to update your `PATH` environment variable to add the `bin` directory, and your `DYLD_LIBRARY_PAT` environment variable to add the `lib` directory:

```
$ export PATH=`pwd`/bin:$PATH
$ export DYLD_LIBRARY_PATH=`pwd`/lib:$DYLD_LIBRARY_PATH
```

Installing a C++ Source Build on OS X

Simply run `make install`. This will install Ice in the directory specified by the `prefix` variable in `config/Make.rules`.

After installation, make sure that the *`prefix`*`/bin` directory is in your `PATH`. For C++11 builds, binaries are installed in *`prefix`*`/bin/c++11`.

If you choose to not embed a runpath into executables at build time (see your build settings in `cpp/config/Make.rules`) or did not create a symbolic link from the runpath directory to the installation directory, you also need to add the prefix/lib directory to your DYLD_LIBRARY_PATH. For C++11 builds, use *`prefix`*`/lib/c++11` instead of *`prefix`*`/lib`.

When compiling Ice programs, you must pass the location of the *`prefix`*`/include` directory to the compiler with the `-I` option, and the location of the *`prefix`*`/lib` directory with the `-L` option. If you are using C++11, use *`prefix`*`/lib/c++11` instead of *`prefix`*`/lib` for the `-L` option.