Building Ice for Python

This page describes how to build and install Ice for Python from source code. If you prefer, you can also download binary distributions for the supported platforms.

On this page:

Python Build Requirements

Ice for Python is expected to build and run properly on Windows, OS X, Solaris, and any recent Linux distribution for x86 and x86_64, and was extensively tested using the operating systems and Python versions listed on our platforms page.

Ice for Python supports Python versions 2.6, 2.7, or 3.3. Note however that your Python installation must have been built with a C++ compiler that is compatible with the one used to build Ice for C++.

You will also need the Ice 3.5.1 development kit for C++, which you can install as a binary distribution or compile from source yourself.

Depending on your platform, you may need to download the Python source distribution and make your own Python build.

Building Ice for Python on Linux/Solaris/OS X

Follow the steps below to build the Ice extension for Python.

Change to Ice for Python source subdirectory:

```
$ cd Ice-3.5.1/py
```

If you have not built Ice for C++ in the `cpp` subdirectory, set `ICE_HOME` to the directory of your Ice for C++ installation. For example:

```
$ export ICE_HOME=/opt/Ice-3.5.1
```

Edit `config/Make.rules`, modify the installation prefix (if necessary), and review the comments describing the `PYTHON_VERSION` variable.

Execute `python -V` to verify that the correct Python interpreter is in your executable search path.

Run `make` to build the extension.

Upon successful completion, run `make install`. You may need additional user privileges to install in the directory specified by `config/Make.rules`.

Refer to Configuring your Environment for Python for more information.

Building Ice for Python on Windows

The Python interpreter is readily available on Windows platforms. You can build it yourself using Microsoft Visual C++, or obtain a binary distribution from the Python web site. The Python 3.3.x binary distribution is compiled with Visual C++ 10.0, and you should use this binary distribution if you want to compile the Ice extension with Visual C++ 10.0.

Follow the steps below to build the Ice extension for Python.

Open a command prompt that supports command-line compilation with Visual C++. For example, you can execute the Visual C++ batch file `vcvars32.bat` to configure your environment. Alternatively, you can start a *Visual Studio Command Prompt* by selecting the appropriate entry from the Visual Studio program group in your Start menu.

Change to the Ice for Python source subdirectory:

```
> cd Ice-3.5.1\py
```

If you have not built Ice for C++ from the `cpp` subdirectory, set `ICE_HOME` to the directory of your Ice for C++ installation. For example:

```
> set ICE_HOME=C:\Ice-3.5.1
```

Edit `config\Make.rules.mak` and review the settings. In particular you must set `CPP_COMPILER` to the appropriate compiler.

Run nmake:

```
> nmake /f Makefile.mak
```

Upon completion, the Ice extension is created as `python\IcePy.pyd`.

Normally you should build with `OPTIMIZE=yes`. If you wish to build a debug version of the Ice extension, set `OPTIMIZE=no`. In this case, you will also need to build a debug version of the Python interpreter from sources.

Configuring your Environment for Python

On Unix, modify your `PYTHONPATH` environment variable to include the Ice extension for Python. Assuming you installed the extension in the default directory (`/opt/Ice-3.5.1`), you would modify your environment as shown below:

```
$ export ICEPY_HOME=/opt/Ice-3.5.1
$ export PYTHONPATH=$ICEPY_HOME/python:$PYTHONPATH
```

On Windows, modify your environment to allow Python to find the Ice extension for Python. The interpreter must be able to locate the extension DLL as well as the Python source files in the `python` subdirectory. This is normally accomplished by setting the `PYTHONPATH` environment variable to contain the necessary subdirectory. For example, if the Ice for Python extension is installed in `C:\Ice-3.5.1`, you could configure your environment as follows:

```
> set ICEPY_HOME=C:\Ice-3.5.1
> set PYTHONPATH=%ICEPY_HOME%\python
```

Running the Python Tests

To run the tests, open a command window and change to the top-level directory. At the command prompt, execute:

```
> python allTests.py
```

You can also run tests individually by changing to the test directory and running this command:

```
> python run.py
```

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

Solaris Notes for Python

Python needs to be configured and built with the following options in order to successfully load the Ice extension:

```
$ ./configure --enable-shared --with-cxx-main=CC --without-gcc ...
```

After running `configure`, edit the generated `makefile` to comment out the following line:

```
BASECFLAGS= -OPT:Olimit=0
```