Building Ice for C++ for Windows Applications

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 Windows installer that contains pre-compiled debug and release libraries, executables, and everything else necessary to build Ice applications on Windows.

On this page:

Windows Build Requirements

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 (OpenSSL 1.0 or later recommended)
- bzip2 1.0
- mcpp 2.7.2 (with patches)

You do not need to build these packages yourself, as ZeroC supplies a Windows installer that contains release and debug libraries for all of the third-party dependencies.

If you intend to build the third-party dependencies from source, 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.

Monotonic Clock

Ice uses the QueryPerformanceCounter Windows API function to measure time with a monotonic clock. If you are experiencing timing or performance issues, there are two knowledgebase articles that may be relevant for your system:

- KB 896256
- KB 895980

Compiling and Testing Ice for C++ on Windows

Building Ice

Using your favorite Zip tool, unzip the Ice source archive anywhere you like.

Open a command prompt that is configured for your target architecture. For example, when using Visual Studio 2010 or later, you have several alternatives:

- Visual Studio Command Prompt
- Visual Studio x64 Win64 Command Prompt
- Visual Studio x64 Cross Tools Command Prompt

Using the first configuration produces 32-bit binaries, while the second and third configurations produce 64-bit binaries.

You must be using a Windows x64 platform when compiling a 64-bit version of Ice.

Change the working directory to `Ice-3.5.1`. For example:

```
> cd C:\Ice-3.5.1\cpp
```

Edit `config\Make.rules.mak` to establish your build configuration. The comments in the file provide more information.

Now you're ready to build Ice:

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

Running the Test Suite

Python is required to run the test suite.

Open a command prompt and change to the top-level Ice directory.

Add the `bin` directory of the third-party libraries to your `PATH`.

For x86 builds:

```
> set PATH=ThirdPartyHome\bin;%PATH%
```

For x64 builds:

```
> set PATH=ThirdPartyHome\bin\x64;%PATH%
```

You can now run the test suite. At the command prompt, execute:

```
> 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, which contains the Ice DLLs and executables.

x64 Platform

Building Ice on x64 with the Visual Studio C++ compiler is like building Ice on x86. You just need to perform the build in a "Visual Studio x64 Win64 Command Prompt", and not in a regular "Visual Studio Command Prompt".

You must be using a Windows x64 platform when compiling a 64-bit version of Ice.

Installing a C++ Source Build on Windows

Simply run nmake /f Makefile.mak install. This will install Ice in the directory specified by the `prefix` variable in `config\Make.rules.mak`.

After installation, if you plan to use the Visual Studio IDE for your Ice project, make sure to add the `bin` directory to the Visual C++ "Executable files", the `include` directory to the "Include files" and the `lib` directory to the "Library files" in the IDE.

If you built a 64-bit version of Ice, the binaries are installed in the `bin\x64` directory and the libraries are installed in the `lib\x64` directory.