Building Ice for C++ for WinRT 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 for WinRT.

On this page:

C++ Build Requirements for WinRT

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

Compiling and Testing Ice for WinRT

Building Ice

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

To build Ice for WinRT you first need to build Ice for Windows. The build of Ice for Windows is necessary to create the Slice translators that we need to build Ice for WinRT.

Open a command prompt that is configured for your target architecture. For example, Visual Studio gives you several alternatives:

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

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

Change the working directory:

```
> 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. In particular, you must set `WINRT` to `yes` in `Make.rules.mak` or in your environment:

```
> set WINRT=yes
```

Now you're ready to build Ice:

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

After the build has completed, you must register the Ice SDK in the Windows registry:

```
> nmake /f Makefile.mak register-sdk
```

> This command must be executed in a command prompt that has administrative privileges because it requires write access to the registry.

Running the Test Suite

The WinRT test suite is composed of a set of dynamic libraries (one for each client/server test) and a GUI application that loads and runs the tests in the dynamic libraries.

You need to build the dynamic libraries first. Change the working directory:

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

Run `nmake` to build the test libraries:

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

In Visual Studio, open this solution file:

```
cpp/test/WinRT/TestSuite.sln
```

When using Visual Studio 2013, the dynamic libraries built in the previous step will automatically target Windows 8.1. The solution must also be updated in this case, so choose *Project > Retarget Windows Store projects to Windows 8.1* before proceeding.

Now select the configuration that matches the settings in `config\Make.rules.mak` that you used to build the dynamic libraries. For example, if you built the test libraries for x86 and debug, you must select Win32 Debug.

After selecting the appropriate configuration, build the solution by choosing "Build Solution" in the "Build" menu.

After the build completes, you can deploy the application using "Deploy Solution" in the "Build" menu. Once deployed, you can start the application from the WinRT Desktop by clicking the "Ice Test Suite" icon.

If you want to run the tests with SSL enabled, you must use servers from another language mapping as WinRT does not support server-side SSL.

To run a test with SSL, open a command window and change to the test directory. At the command prompt, execute:

```
> python run.py --winrt --protocol=ssl
```

Then open the "Ice Test Suite" Windows Store application, check "Enable SSL", and click the "Run" button.

To run the tests on a remote device such as the Surface, you will need to install the Remote Debugger Tools.

> Some tests might fail if you run the tests with the debugger attached. You should choose "Start without Debugging" from the Debug menu to run the tests.

Installing a C++ Source Build for WinRT

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

> This command must be executed in a command prompt that has administrative privileges because it requires write access to the registry.