Using the Windows Binary Distribution

This page provides important information for users of the Ice binary distribution on Windows platforms.

On this page:

Overview of the Windows binary distribution

The Ice binary distribution for Windows provides all Ice run time services and development tools to build Ice applications on Windows:

- in C++, using Visual Studio 2008 SP1, Visual Studio 2010 or C++Builder 2010
- in .NET, using Visual Studio 2008 SP1 or Visual Studio 2010
- in Java, using Java 5 or Java 6
- in Android, using Android 2.1 or later
- in Python, using Python 2.6.4
- in Ruby, using Ruby 1.8.6
- in PHP, using PHP 5.3.1

You only need the development environment for your target programming language to use this distribution. For example, if you want to build Ice applications in Java, you need to install a JDK, but do not need to install Visual Studio or Python.

Setting up your Windows 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.

C++

To use Ice for C++ you need to add one or more of the Ice `bin` directories to your `PATH`. The changes you must make depend on your compiler and architecture, as listed below:

- Visual Studio 2008SP1, 32-bit

```
> set PATH=<Ice installation root directory>\bin;%PATH%
```

- Visual Studio 2008SP1, 64-bit

```
> set PATH=<Ice installation root directory>\bin;%PATH%
> set PATH=<Ice installation root directory>\bin\x64;%PATH%
```

  The `x64` directory must come first in your `PATH`.

- Visual Studio 2010, 32-bit

```
> set PATH=<Ice installation root directory>\bin\vc100;%PATH%
```

- Visual Studio 2010, 64-bit

```
> set PATH=<Ice installation root directory>\bin\vc100;%PATH%
> set PATH=<Ice installation root directory>\bin\vc100\x64;%PATH%
```

The `x64` directory must come first in your `PATH`.

- C++Builder 2010

```
> set PATH=<Ice installation root directory>\bin;%PATH%
> set PATH=<Ice installation root directory>\bin\bcc10;%PATH%
```

The `bcc10` directory must come first in your `PATH`.

If you don't want to rely on the `PATH` environment variable to locate the Ice for C++ DLLs, you can also copy the DLLs into the same directory as your executable.

To compile Ice for C++ applications with Visual C++, you should use the Ice Visual Studio Add-In.

To compile with Visual C++ Express where the Ice Visual Studio Add-in is not available, you need to configure Visual Studio manually. This involves adding the locations of the Ice header files, libraries, and executables to Visual Studio's configuration. Follow these steps:

1. In the IDE, choose *Tools* / *Options* / *Projects and Solutions* / *VC++ Directories*
2. Select `Include files`
3. Add <*`Ice installation root directory`*>`\include`
4. Select `Library files` and add the appropriate library directory for your compiler:

VS2008SP1, 32-bit	<*`Ice installation root directory`*>`\lib`
VS2008SP1, 64-bit	<*`Ice installation root directory`*>`\lib\x64`
VS2010, 32-bit	<*`Ice installation root directory`*>`\lib\vc100`
VS2010, 64-bit	<*`Ice installation root directory`*>`\lib\vc100\x64`

5. Select `Executable files` and add the appropriate binary directory for your compiler:

VS2008SP1, 32-bit	<*`Ice installation root directory`*>`\bin`
VS2008SP1, 64-bit	<*`Ice installation root directory`*>`\bin\x64`
VS2010, 32-bit	<*`Ice installation root directory`*>`\bin\vc100`
VS2010, 64-bit	<*`Ice installation root directory`*>`\bin\vc100\x64`

.NET

Locating the .NET assemblies

To use Ice for .NET, you can either copy the .NET assemblies to the directory of your executable or add the .NET assemblies to the Global Assembly Cache (GAC).

Copying the Ice for .NET assemblies to the executable directory is the simplest solution. You can set up your Visual Studio projects to copy the assemblies by setting the `Copy Local` property to `True`. To access this property in the Solution Explorer, open the `References` folder of your project and click on the assembly to access its properties in the `Properties` panel.

You can also add the Ice for .NET assemblies to the GAC. To do this, open Windows Explorer and navigate to the directory `C:\WINDOWS\assembly`. Next, drag and drop (or copy and paste) the .NET assemblies from the <*`Ice installation root directory`*>`\bin` directory into the right-hand pane to install them in the cache.

Alternatively, you can achieve the same result by using `gacutil` from the command line:

```
> gacutil /i <library.dll>
```

The `gacutil` tool is included with your Visual C# installation. For example, if you have installed Visual C# 9.0 in `C:\Program Files`, the path to `gacutil` is

```
C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\gacutil.exe
```

Once installed in the GAC, the assemblies will always be located correctly without having to set environment variables or copy them into the same directory as an executable.

Line numbers for stack traces

If you want line numbers for stack traces, you must also install the PDB (.pdb) files in the GAC. Unfortunately, you cannot do this using Explorer, so you have to do it from the command line. Open a command shell window and navigate to `C:\WINDOWS\assembly\GAC_MSIL\Ice` (assuming `C:\WINDOWS` is your system root). Doing a directory listing there, you will find a directory named `3.4.2.0__<UUID>`, for example:

```
3.4.2.0__cdd571ade22f2f16
```

Change to that directory (making sure that you use the correct version number for this release of Ice). In this directory, you will see the `Ice.dll` you installed into the GAC in the preceding step. Now copy the `Ice.pdb` file into this directory:

```
> copy <\path\to\Ice.pdb> .
```

Using protocol compression

The Ice for .NET run time implements protocol compression by dynamically loading the native library `bzip2.dll` from a directory in your `PATH`. Ice disables the protocol compression feature if it is unable to load the `bzip2` library successfully.

This DLL is included in your Ice distribution, therefore the Ice `bin` directory must be added to your `PATH`:

```
> set PATH=<Ice installation root directory>\bin;%PATH%
```

On 64-bit Windows, use the following setting instead:

```
> set PATH=<Ice installation root directory>\bin\x64;%PATH%
```

If the wrong `PATH` is set, the Ice run time prints a warning to the console when it detects a `bzip2.dll` format mismatch during start-up.

.NET Compact Framework

The Ice assembly for the .NET Compact Framework is installed as *<Ice installation root directory>*`\bin\cf\Ice.dll`. The Visual Studio Add-in detects a Smart Device project and automatically configures it to use the .NET CF version of the Ice assembly.

Java

To use Ice for Java, you must add `Ice.jar` to your `CLASSPATH`, as shown below:

```
> set CLASSPATH=<Ice installation root directory>\lib\Ice.jar;%CLASSPATH%
```

If you intend to use Freeze for Java, you must include `Freeze.jar` in your `CLASSPATH` along with `Ice.jar`:

```
> set CLASSPATH=<Ice installation root directory>\lib\Freeze.jar;%CLASSPATH%
```

Furthermore, to use a Freeze demo the JVM requires that the directory containing Berkeley DB's native libraries be included in `java.library.path`, therefore you must add this directory to your `PATH`:

```
> set PATH=<Ice installation root directory>\bin;%PATH%
```

For a 64-bit JVM, use the following setting instead:

```
> set PATH=<Ice installation root directory>\bin\x64;%PATH%
```

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.

Eclipse Development

ZeroC has created a Slice2Java plug-in for Eclipse that automates the translation of your Slice files. If you use Eclipse, we strongly recommend using this plug-in for your own development.

The Slice2Java plug-in is required if you intend to build any of the Android sample projects included in this distribution.

For installation instructions, please refer to the ZeroC web site. The manual provides more information about configuring the plug-in and using it in your projects.

Android

Ice requires Android 2.1 or later. Aside from that, there are no other special requirements for using Ice in an Android application. We strongly recommend installing our Slice2Java plug-in for Eclipse to automate the compilation of your Slice definitions.

Python

To use Ice for Python, you must add the Ice `bin` directory to your `PATH` and set `PYTHONPATH` so that the Python interpreter is able to load the Ice extension. For a 32-bit Python installation, use these settings:

```
> set PATH=<Ice installation root directory>\bin;%PATH%
> set PYTHONPATH=<Ice installation root directory>\python
```

For a 64-bit Python installation, use these settings instead:

```
> set PATH=<Ice installation root directory>\bin\x64;%PATH%
> set PYTHONPATH=<Ice installation root directory>\python\x64
```

Ruby

To use Ice for Ruby, you must add the Ice `bin` directory to your `PATH`:

```
> set PATH=<Ice installation root directory>\bin;%PATH%
```

You must also set `RUBYLIB` so that the Ruby interpreter is able to load the Ice extension:

```
> set RUBYLIB=<Ice installation root directory>\ruby;%RUBYLIB%
```

The Ruby installer includes versions of the OpenSSL DLLs that are not compatible with the ones supplied with Ice. If you intend to use SSL in your Ice for Ruby applications, you will need to remove or rename the following files in the Ruby installation directory:

```
libeay32.dll
ssleay32.dll
```

If you used the default installation directory, these files are located in `C:\ruby\bin`.

Also be aware that the Ruby installer inserts `C:\ruby\bin` at the beginning of the system `PATH`, therefore the DLLs listed above can also have an adverse impact on other Ice language mappings that use OpenSSL, such as C++ and Python.

PHP

The binary distribution of PHP 5.3.6 for Windows is compiled with Visual Studio 2008 (Visual C++ 9) and therefore is not compatible with the Apache binaries provided by the Apache Software Foundation, which are compiled with Visual C++ 6.

The Ice extension for PHP included in this installer is also compiled with VC9 for compatibility with the PHP binary distribution. To use this extension, you will need a compatible PHP binary distribution as well as a compatible Web server. If you wish to use Apache, you can obtain a VC9 build of Apache from alternate sources.

If you require a version of the Ice extension for a different environment, you will need to compile the extension from source. Download the Ice source distribution and review the `php/INSTALL` file for details.

The PHP documentation describes how to configure the Apache servers for PHP, and the PHP installer may have already performed the necessary steps. We provide instructions below for configuring PHP to use the Ice extension. These instructions make several assumptions:

- Apache 2.2 is installed and configured to load PHP
- PHP is installed in `C:\Program Files\PHP`
- Ice is installed in `C:\Ice`

If you have a different installation, you will need to make the appropriate changes as you follow the instructions.

1. Verify your PHP installation

With Apache running, verify that PHP has been loaded successfully by creating a file in Apache's document directory (`htdocs`) called `phpinfo.php` that contains the following line:

PHP

```
<?php phpInfo();?>
```

Open the file in your browser using a URL such as

`http://127.0.0.1/phpinfo.php`

If you have configured PHP correctly, you should see a long page of PHP configuration information. If you do not see this page, or an error occurs, check Apache's error log as well as the Windows event log for more information. Also note that you may need to restart Apache if it was running at the time you installed PHP.

2. Determine the location of your PHP configuration file

Review the settings on the browser page for an entry titled `Loaded Configuration File`. It will have a value such as

```
C:\Program Files\PHP\php.ini
```

As an administrator, open this file in a text editor and append the following line:

```
extension = php_ice.dll
```

The file `php_ice.dll` contains the Ice extension for PHP.

3. Install the Ice extension DLL

Look for the `extension_dir` setting in the browser page or in PHP's configuration file. It typically has the following value by default:

```
extension_dir = "C:\Program Files\PHP\ext"
```

If instead the `extension_dir` setting contains a relative path, it it is resolved relative to the working directory of the Apache process (Apache's working directory is usually its installation directory).

Copy the DLL for the Ice extension to PHP's extension directory:

```
> copy C:\Ice\bin\php_ice.dll "C:\Program Files\PHP\ext"
```

4. Verify that Apache can load dependent libraries

Regardless of the location of PHP's extension directory, the Ice extension's dependent libraries must be located in Apache's executable search path.

The Ice extension depends on the following libraries:

- `bzip2.dll`
- `ice34.dll`
- `iceutil34.dll`
- `slice34.dll`

All of these files can be found in the `bin` subdirectory of your Ice installation (e.g., `C:\Ice\bin`). Apache must be able to locate these DLLs during startup, and several alternatives are available:

- Add the Ice installation directory to the system `PATH`. Using the `System` control panel, change the system `PATH` to include `C:\Ice\bin`. Note that Windows must be restarted for this change to take effect.
- Copy the dependent libraries to Apache's installation directory.
- Copy the dependent libraries to the Windows system directory (`C:\WINDOWS\system32`). We do not recommend this option.

If Apache cannot find or access a DLL, Apache startup may fail with an access violation, or the PHP module may ignore the Ice extension and continue its initialization. Consequently, a successful Apache startup does not necessarily mean that the Ice extension has been loaded. Unfortunately, the message reported by PHP in Apache's error log is not very helpful; the error might imply that it cannot find `php_ice.dll` when in fact it was able to open `php_ice.dll` but a dependent DLL was missing or inaccessible.

5. Review access rights

Review the access rights on PHP's extension directory, the Ice extension DLL, and its dependent libraries. When running as a Windows service, Apache runs in the `Local System` account (also known as `NT Authority\SYSTEM`). You can use the `cacls` utility in a command window to view and modify access rights. For example, run the following commands to review the current access rights of the Ice extension:

```
> cd \Program Files\PHP\ext
> cacls php_ice.dll
```

6. Restart Apache

Restart Apache and verify that the PHP module and the Ice extension have been loaded successfully. After reloading the `phpinfo.php` page in your browser, scan the entries for a section titled `ice`. The presence of this section indicates that the extension was loaded.

If Apache does not start, check the Windows Event Viewer as well as Apache's log files for more information. The most likely reasons for Apache to fail at startup are missing DLLs (see step 4) or insufficient access rights (see step 5).

7. Locate the Ice run time files

Your application will also need to include at least some of the Ice for PHP run-time source files (installed in `C:\Ice\php`). To make these files available to your application, you can either modify PHP's include path or copy the necessary files to a directory that is already in the interpreter's include path. You can determine the current include path by loading the `phpinfo.php` page in your browser and searching for an entry named `include_path`.

If you want to make the Ice run-time files available to all PHP applications on the host, you can modify the `include_path` setting in `php.ini` to add the installation directory:

```
include_path = C:\Ice\php;...
```

Another option is to modify the include path from within your script prior to including any Ice run-time file:

PHP

```
ini_set('include_path', ini_get('include_path') . PATH_SEPARATOR . 'C:/Ice/php')
require 'Ice.php'; // Load the core Ice run time definitions.
```

`iceboxnet` with .NET 4.0

The `iceboxnet.exe` executable included in the Windows binary distribution was built with .NET 3.5 and requires extra configuration to load IceBox services that rely on the .NET 4.0 run time.

The solution is to create a file named `iceboxnet.exe.config` in the same directory as `iceboxnet.exe`:

XML

```
<?xml version="1.0"?>
<configuration>
 <startup>
   <supportedRuntime version="v4.0" />
 </startup>
</configuration>
```

With this configuration, `iceboxnet` uses the .NET 4.0 run time and is able to load IceBox services built with .NET 4.0.

Managed code in Ice for .NET

The main Ice for .NET assembly (`Ice.dll`) included in the Windows binary distribution uses unmanaged code. If you require only managed code then you can download the Ice source distribution and build Ice for .NET in a purely managed version. Note that the managed version of Ice for .NET omits support for protocol compression and for signal handling in the `Ice.Application` class.

You can download the source distribution at the ZeroC web site.

Using the sample programs on Windows

During installation, sample programs for all supported programming languages are installed by default in a sub-folder of the `Documents` or `My Documents` folder of the user who performed this installation. You'll find a shortcut to this directory in your Start menu. You can also download a ZIP archive of these sample programs, and extract this archive in a folder of your choice.

The sample programs are in source form only. Please refer to the `README.txt` file in the main demos folder for instructions on how to build and execute them.

Configuration files for IceGrid and Glacier2 services

The `config` subdirectory of your Ice installation includes sample configuration files for the Glacier2 router, IceGrid node, and IceGrid registry. These files provide a good starting point on which to base your own configurations, and they contain comments that describe the settings in detail.

If you intend to edit one of the configuration files but you have installed your distribution into `\Program Files` or `\Program Files (x86)` on Windows Vista or later, be aware that Windows makes it difficult to make permanent modifications to files in these directories. We recommend copying the files to the location of your choice. For more information on this topic, refer to the **Virtualization** section of New UAC Technologies for Windows Vista.

The Ice manual provides more information on installing and running the IceGrid registry, IceGrid node, and Glacier2 router as Windows services.

Starting the IceGrid Administrative Console on Windows

You can launch the IceGrid Administrative Console using the shortcut that the Ice installer created in your Start menu as **IceGrid GUI**. The console is a Java program and requires JRE 5.0 or later.

Ice Visual Studio Add-In installation notes

The Ice binary distribution includes an add-in for Visual Studio that helps to create and manage projects with Slice files. The Ice Visual Studio Add-In supports C++, .NET, and .NET Compact Framework projects.

The add-in does not support Visual Studio Express editions because Microsoft does not permit add-ins to be written for Express editions of Visual Studio.

The installer for the Ice binary distribution detects if Visual Studio 2008 or Visual Studio 2010 is installed on the target computer and configures the corresponding Ice add-in for all users. If you install Visual Studio 2008 or Visual Studio 2010 *after* installing Ice, you will have to re-run the Ice installer and choose **Repair** to install the add-in.

On a machine with multiple users, be aware that the installers for Ice 3.4.0 and Ice 3.4.1 placed a file in the documents directory of the user that executed the installer. For example, when using Visual Studio 2008 on Windows 7, this file is installed as

```
C:\Users\user\My Documents\Visual Studio 2008\Addins\Ice-VS2008.AddIn
```

This file is automatically removed if the same user executes the Ice 3.4.2 installer. However, if a different user installs Ice 3.4.2, the `.AddIn` file will not be removed from the original user's documents directory and that user will continue to use the old version of the add-in. In order for that user to use the add-in for Ice 3.4.2, the `.AddIn` file must be removed manually. After removing the file and starting Visual Studio for the first time, you may notice that the `Ice Configuration` menu option is not present and an error dialog appears when you close Visual Studio. Restart Visual Studio once more to begin using the Ice 3.4.2 add-in.

Ice installer registry key

The Ice installer adds information to the Windows registry to indicate where it was installed. Developers can use this information to locate the Ice files in their applications.

The registration key used by this installer is:

`HKEY_LOCAL_MACHINE\Software\ZeroC\Ice 3.4.2`

On 64-bit machines this key is added to the 64-bit registry, but not the 32-bit registry.

The install location is stored as a string value named `InstallDir`.

Third-party packages for Windows

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

- Berkeley DB 4.8.30 (C/C++ and Java run time)
- Bzip2 1.0.6 (C run time)
- Expat 2.0.1 (C run time)
- OpenSSL 0.9.8r (C run time)
- QtCore and QtSql 4.5.3 with SQLite driver built-in (C++ run time)
- STLPort 4.6.3 (C++ run time)