

Building Ice for Ruby

This page describes how to build and install Ice for Ruby from source code. If you prefer, you can also download [binary distributions](#) for the supported platforms.

On this page:

- [Ruby Build Requirements](#)
 - [Ruby Prerequisites for Linux](#)
 - [Ruby Prerequisites for Windows](#)
- [Building the Ruby Extension](#)
 - [Building Ice for Ruby on Linux](#)
 - [Building Ice for Ruby on Windows](#)
- [Installing the Ruby Extension](#)
 - [Installing Ice for Ruby on Linux](#)
 - [Installing Ice for Ruby on Windows](#)
- [Configuring your Environment for Ruby](#)
 - [Using Ice for Ruby on Linux](#)
 - [Using Ice for Ruby on Windows](#)
- [Running the Ruby Tests](#)
- [Running the Ruby Demos](#)
- [SELinux Notes for Ruby](#)

Ruby Build Requirements

Ice for Ruby is expected to build and run properly on Windows and any recent Linux distribution for x86 and x86_64, and was extensively tested using the operating systems and Ruby versions listed on our [platforms page](#).



Ruby currently does not implement server-side functionality, therefore Ice for Ruby can only be used in client applications. If you have a need for this feature and wish to sponsor its development, please contact us at info@zeroc.com.

Ruby Prerequisites for Linux

To build Ice for Ruby you must have the following:

- Ice 3.5.1 development kit for C++
- Ruby 1.8.1 or later

You can use a source or binary installation of Ruby. If you use an RPM installation, the following packages are required:

```
ruby
ruby-devel
ruby-libs (RHEL)
```

Ruby Prerequisites for Windows

To build Ice for Ruby you must have the following:

- Ice 3.5.1 development kit for C++
- Ruby 1.9.3
- Ruby Development Kit 4.5.2

The Ruby distribution for Windows uses the MinGW compiler, therefore MinGW is the only compiler supported by Ice for Ruby.

The instructions in this file make the following assumptions about your build environment:

1. You have installed the Ice 3.5.1 distribution using the [ZeroC installer](#). The default installation directory is C:\Program Files\ZeroC\Ice-3.5.1.
2. You have installed [Ruby 1.9.3](#) using the Windows installer. The default installation directory is C:\Ruby193.
3. You have installed the [Ruby Development Kit 4.5.2](#).

If you selected different installation directories, you will need to modify the relevant path names in the steps below to match your configuration.

Building the Ruby Extension

Building Ice for Ruby on Linux

The instructions for compiling the Ice extension assume that you have already installed Ruby.

If you installed Ruby in a non-standard location, set the `RUBY_HOME` environment variable to the installation directory. For example:

```
$ export RUBY_HOME=/opt/ruby-1.8.6
```

If you have not built Ice for C++ from the `cpp` subdirectory or if you have installed the Ice for C++ development kit in a non-standard location, set the `ICE_HOME` environment variable to the installation directory. For example:

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

Review the settings in `config/Make.rules` and adjust as necessary. For example, you may wish to enable `OPTIMIZE`.

Run `make` to build the extension.

Building Ice for Ruby on Windows

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

Open a Windows command prompt and add Ruby to your environment:

```
> C:\Ruby193\bin\setrbvars.bat
```

Add MinGW from the Ruby Development Kit to your `PATH`:

```
> C:\RubyDevKit-4.5.2\devkitvars.bat
```

Change to the Ice for Ruby source directory:

```
> cd Ice-3.5.1/rb
```

If you have not [built Ice for C++](#) from the `cpp` subdirectory, set the `ICE_HOME` environment variable to point to your Ice installation. This path must contain forward slashes (/) as directory separators, and cannot contain any space. If your Ice installation's path contains any space, use the DOS 8-character name as a work-around, for example:

```
> set ICE_HOME=C:/PROGRA~2/Ice-3.5.1
```

You can use `dir /x` to get this DOS name, for example:

```
> cd c:\
> dir /x
```

```
Directory of c:\
```

```
...
```

```
03/03/2014 03:37 PM <DIR> PROGRA~1 Program Files
07/25/2014 12:54 PM <DIR> PROGRA~2 Program Files (x86)
```

Then run `make` to build the extension:

```
> make
```

Installing the Ruby Extension

Installing Ice for Ruby on Linux

You can perform an automated installation with the following command:

```
$ make install
```

This process uses the `prefix` variable in `config/Make.rules` as the installation's root directory. The subdirectory `prefix/ruby` is created as a copy of the local `ruby` directory and contains the Ice for Ruby extension library (`IceRuby.so`) as well as Ruby source code. Using this installation method requires that you modify your environment as described in "Using Ice for Ruby" below.

Another option is to copy the contents of the local `ruby` directory to your Ruby installation's `site_ruby` directory. For example, if you installed Ruby via RPM, you can use the steps below:

```
# cd <Ice source directory>/rb/ruby
# sudo tar cf - * | (cd /usr/lib/ruby/site_ruby/1.8/i386-linux; tar xvf -)
```

On x86_64 systems, change the last command to:

```
# sudo tar cf - * | (cd /usr/lib64/ruby/site_ruby/1.8/x86_64-linux; tar xvf -)
```

There is no need to modify your environment if you use this approach.

Installing Ice for Ruby on Windows

You can perform an automated installation with the following command:

```
> make install
```

This process uses the `prefix` variable in `config\Make.rules` as the installation's root directory. The subdirectory `prefix\ruby` is created as a copy of the local `ruby` directory and contains the Ice for Ruby extension library (`IceRuby.so`) as well as Ruby source code. Using this installation method requires that you modify your environment as described below.

Configuring your Environment for Ruby

Using Ice for Ruby on Linux

The Ruby interpreter must be able to locate the Ice extension. If you used the automated installation described above, you need to define the `RUBYLIB` environment variable as follows:

```
$ export RUBYLIB=/opt/Ice-3.5.1/ruby:$RUBYLIB
```

This example assumes that your Ice for Ruby installation is located in the `/opt/Ice-3.5.1` directory.

You must also modify `LD_LIBRARY_PATH` to include the directory `/opt/Ice-3.5.1/lib`:

```
$ export LD_LIBRARY_PATH=/opt/Ice-3.5.1/lib:LD_LIBRARY_PATH
```

To verify that Ruby can load the Ice extension successfully, open a command window and start the interpreter using `irb`:

```
> irb
irb(main):001:0>
```

At the prompt, enter

```
require "Ice"
```

If the interpreter responds with the value `true`, the Ice extension was loaded successfully. Enter `exit` to quit the interpreter.

Using Ice for Ruby on Windows

The Ruby interpreter must be able to locate the Ice extension. One way to configure the interpreter is to define the `RUBYLIB` environment variable as follows:

```
> set RUBYLIB=C:\Ice-3.5.1\ruby
```

This example assumes your Ice for Ruby installation is located in the `C:\Ice-3.5.1` directory.

In addition, you must modify your `PATH` environment variable to include the following directories:

```
C:\Program Files\ZeroC\Ice-3.5.1\bin
C:\Ice-3.5.1\bin
```

At a command prompt, you can set your `PATH` as shown below:

```
> set PATH=C:\Program Files\ZeroC\Ice-3.5.1\bin;C:\Ice-3.5.1\bin;%PATH%
```

Running the Ruby Tests

The test subdirectory contains Ruby implementations of the core Ice test suite. [Python](#) is required to run the test suite.

The test suites require that the Ice for C++ tests be built in the `cpp` subdirectory of this source distribution.

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".

Running the Ruby Demos

The demo directory contains Ruby versions of the Ice sample programs. Note that only clients are provided, since Ice for Ruby does not support server-side activities. In order to run a sample client, you must first start its corresponding server from another Ice language mapping, such as C++.

As an example, let's run the hello application in `demo/Ice/hello` using the C++ server. Assuming that you have already compiled the sample C++ programs in the `cpp` subdirectory, we begin by starting the server:

```
> cd Ice-3.5.1/cpp/demo/Ice/hello
> server
```

In a separate window, start the client:

```
> cd Ice-3.5.1/rb/demo/Ice/hello
> ruby Client.rb
```

Some demo directories contain `README` files if additional requirements are necessary.

SELinux Notes for Ruby

If [SELinux](#) is enabled on your RHEL system, you may encounter this error message when Ruby attempts to load the Ice extension:

```
cannot restore segment prot after reloc: Permission denied
```

There are two ways to solve this problem:

- Change the default security context for the Ice extension using the following command:

```
$ chcon -t texrel_shlib_t /opt/Ice-3.5.0/ruby/IceRuby.so
```

Replace `/opt/Ice-3.5.1` with your installation directory.

- Disable SELinux completely by adding the following line to your `/etc/sysconfig/selinux` file:

```
SELINUX=disabled
```