Upgrading your Application from Ice 3.5.0

Ice 3.5.1 maintains binary compatibility with Ice 3.5.0, therefore it is not necessary for you to recompile your Slice files or your program code, nor is it necessary to relink your application. (Custom transports are an exception to this rule.) The database formats used by Ice services such as IceGrid and IceStorm have not changed, therefore no database migration is required. Finally, this release has not removed any APIs. Generally speaking, you are free to use any combination of Ice 3.5.0 and Ice 3.5.1 applications and Ice services.

The subsections below provide additional information about upgrading to Ice 3.5.1, including administrative procedures for the supported platforms.

On this page:

Custom Transport Plug-ins

Changes to the internal Ice transport API in Ice 3.5.1 break binary and source compatibility for custom transport plug-ins that used Ice 3.5.0. ZeroC does not guarantee backward compatibility for internal APIs, therefore transport developers will need to modify their plug-ins and recompile them with Ice 3.5.1.

Upgrading from Ice 3.5.0 on Linux

For RPM and Ubuntu installations, you can use the Ice 3.5.1 packages to upgrade an existing installation of Ice 3.5.0.

For a Java application, no additional steps are necessary if your `CLASSPATH` refers to `/usr/share/java/Ice.jar`, which is a symbolic link that points to the actual version-specific JAR file.

For a Mono application, the Mono package for Ice 3.5.1 installs the updated Ice run time assemblies into the Global Assembly Cache (GAC) along with policy assemblies that enable backward compatibility with Ice 3.5.0.

Upgrading from Ice 3.5.0 on Windows

The file names of the Ice for C++ run time DLLs do not contain the patch number of a release. For example, the core Ice DLL uses the same name (`ice35.dll`) for Ice 3.5.0 and 3.5.1. As a result, you can simply substitute the 3.5.1 DLLs for the 3.5.0 DLLs. If you install the 3.5.1 DLLs in a different directory, you will typically need to adjust the `PATH` setting of a C++ application so that it can locate the new libraries. This also applies to Python, Ruby, and PHP applications because they use the Ice for C++ DLLs.

For a Java application, you can replace the existing `Ice.jar` file with the one from Ice 3.5.1, or you can adjust the `CLASSPATH` setting to point to the new JAR file.

For a .NET application, Ice for .NET includes policy assemblies that supply the .NET run time with the required compatibility information. Policy assemblies have names of the form `policy.3.5.`*`package`*`.dll`. For example, the policy assembly for IceBox is `policy.3.5.IceBox.dll`. One way to upgrade an existing .NET application to a new patch release while maintaining binary compatibility is to install the policy assemblies into the Global Assembly Cache (GAC) using one of the following methods:

- Open Windows Explorer and navigate to the directory `C:\WINDOWS\assembly`. Next, drag and drop (or copy and paste) the assemblies into the right-hand pane to install them in the GAC.
- Use `gacutil` from the command line:

```
> gacutil -i <policy.dll>
```

Another option is to modify the `.config` file of your application to add `bindingRedirect` directives, as explained in the links below:

http://msdn.microsoft.com/en-us/library/7wd6ex19.aspx
http://msdn.microsoft.com/en-us/library/yx7xezcf.aspx

For example, in the `.config` file of an application you can modify the configuration of the Ice and IceBox assemblies as follows:

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

<dependentAssembly>
  <assemblyIdentity name="Ice" culture="neutral" publicKeyToken="cdd571ade22f2f16"/>
  <bindingRedirect oldVersion="3.5.0.0" newVersion="3.5.1.0"/>
</dependentAssembly>

<dependentAssembly>
  <assemblyIdentity name="IceBox" culture="neutral" publicKeyToken="cdd571ade22f2f16"/>
  <bindingRedirect oldVersion="3.5.0.0" newVersion="3.5.1.0"/>
</dependentAssembly>

</assemblyBinding>
</runtime>
</configuration>
```

This `.config` file can be used for `iceboxnet.exe` (as `iceboxnet.exe.config`), to load a simple IceBox service built with Ice 3.5.0 with `icebox net` 3.5.1.

Note that `cdd571ade22f2f16` is the token corresponding to ZeroC's public key for signing the assemblies in binary distributions. If you built Ice from sources, your assemblies were signed using the development key instead, which you can find in `config/IceDevKey.snk`. The token for the development key is `1f998c50fec78381`.

The advantage of installing the policy assemblies into the GAC is that they establish binary compatibility for all Ice applications, whereas modifying a `.config` file must be done for each application individually.

On a development system, it is not necessary to remove your existing Ice installation prior to installing Ice 3.5.1 unless you intend to install Ice 3.5.1 in the same directory as your existing installation. You may need to update your `PATH` setting and modify your Visual C++ directory configurations to reflect the installation directory for Ice 3.5.1.

Upgrading from Ice 3.5.0 on OS X

The Ice installer installs Ice in `/Library/Developer/Ice-3.5.1` and updates the `/Library/Developer/Ice-3.5` symbolic link to point to the Ice 3.5.1 installation. If your binaries were built with an RPATH set to `/Library/Developer/Ice-3.5/lib`, no additional steps are necessary to use Ice 3.5.1. If they were built without an RPATH or if the RPATH is set to the Ice 3.5.0 installation directory, you will need to re-build them or you will need to set the `DYLD_LIBRARY_PATH` environment variable:

```
export DYLD_LIBRARY_PATH=/Library/Developer/Ice-3.5.1/lib
```

No additional steps are necessary for a Java application if its `CLASSPATH` refers to the JAR file via the symbolic link:

```
export CLASSPATH=/Library/Developer/Ice-3.5/lib/Ice.jar
```

For Python no additional steps are necessary, the Ice installer replaced the Ice for Python files from `/Library/Python/2.7/site-packages` with the Ice 3.5.1 version.

Upgrading from Ice 3.5.0 on Solaris

The Using the Solaris Binary Distribution page describes how to configure your environment so that the embedded path names in the Ice for C++ shared libraries are resolved correctly. For example, if you extracted the binary distribution for Ice 3.5.0 into `/opt/Ice-3.5.0`, it instructs you to create the following symbolic link:

```
/opt/Ice-3.5 -> /opt/Ice-3.5.0
```

To upgrade to Ice 3.5.1, you simply extract the binary distribution archive into `/opt/Ice-3.5.1` and reset the symbolic link to point to the new installation:

```
/opt/Ice-3.5 -> /opt/Ice-3.5.1
```

No additional steps are necessary for a Java application if its `CLASSPATH` refers to the JAR file via the symbolic link:

```
export CLASSPATH=/opt/Ice-3.5/lib/Ice.jar
```

This also applies for Python and Ruby applications:

```
export PYTHONPATH=/opt/Ice-3.5/python
export RUBYLIB=/opt/Ice-3.5/ruby
```

Upgrading from Ice 3.5.0 with a Source Distribution (Linux, OS X, Solaris)

If you compiled an Ice 3.5.0 source distribution and installed it via `make install`, the default installation directory uses a version-specific name such as `/opt/Ice-3.5.0`. Consequently, you can build and install the Ice 3.5.1 source distribution without affecting your existing Ice 3.5.0 installation.

You may need to update your application's build settings to use the new installation directory for locating include and library files. You may also need to update your shared library search path.

The relevant environment variables for each language mapping are detailed below.

- C++, Ruby, Python, PHP

  ```
  export LD_LIBRARY_PATH=/opt/Ice-3.5.1/lib (32-bit Linux & 32-bit Solaris)
  export LD_LIBRARY_PATH=/opt/Ice-3.5.1/lib64 (64-bit Linux)
  export LD_LIBRARY_PATH_64=/opt/Ice-3.5.1/lib/amd64 (64-bit Solaris Intel)
  export LD_LIBRARY_PATH_64=/opt/Ice-3.5.1/lib/sparcv9 (64-bit Solaris SPARC)
  export DYLD_LIBRARY_PATH=/opt/Ice-3.5.1/lib (OS X)
  ```

- Python

  ```
  export PYTHONPATH=/opt/Ice-3.5.1/python
  ```

- Ruby

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

- Java

  ```
  export CLASSPATH=/opt/Ice-3.5.1/lib/Ice.jar:/opt/Ice-3.5.1/lib/Freeze.jar
  ```

- Mono

  ```
  export MONO_PATH=/opt/Ice-3.5.1/bin
  ```

Deprecated APIs in Ice 3.5.1

No APIs were deprecated in Ice 3.5.1.