New Features in Ice 3.5

This page describes significant changes and improvements in Ice 3.5 that may affect the operation of your applications or have an impact on your source code. For a detailed list of the changes in this release, please refer to the changelog.

On this page:

Visual Studio 2013

Ice 3.5.1 adds support for Visual Studio 2013 and Windows 8.1.

SOCKS v4 proxies

Ice 3.5.1 adds support for SOCKS v4 proxies, which allows outgoing Ice connections to be routed through your organization's SOCKS v4-compatible proxy service. This feature is supported in C++, Java, .NET, and the scripting languages, but is not supported under WinRT. The new properties `Ice.SOCKSProxyHost` and `Ice.SOCKSProxyPort` identify the host and port of the proxy service, respectively.

Improved IceGrid database replication

As of Ice 3.5.1, IceGrid now performs a check when a slave registry connects to the master to prevent the slave's database from being overwritten if it contains more recent updates than the master database. For example, this situation could occur if you promote a slave with an out-of-date database to be the new master, if you restart a master with a database from an old backup, or if you restart a master with an empty database. In all of these cases, the slave's connection to the master will now be rejected and its database preserved. This check only works if both the IceGrid master and slave use Ice 3.5.1.

To correct the situation and allow the slave to connect successfully with the master, you can either update the master database to match the slave or, if you choose to discard the more recent updates recorded by the slave, you can update the slave database to match the master.

The new command-line option `--initdb-from-replica=<replica>` allows you to initialize a registry database from a given replica.

New encoding version

Ice 3.5 introduces version 1.1 of the Ice encoding, which is necessary to support several of the new features in this release.

The encoding from Ice 3.5b is not compatible with the 1.1 encoding from Ice 3.5.0. If you are still using Ice 3.5b, you should upgrade all your applications to Ice 3.5.1; a client or server using 3.5b will not interoperate with Ice 3.5.x when using the 1.1 encoding.

Metrics facility

Ice applications can now be instrumented to give remote administrative tools access to detailed metrics about the application's activity, including request success and failure rates, thread utilization, connection establishment, DNS lookups, and throughput.

Administrators can define custom views to monitor specific metrics of interest and minimize impact on the target application. A filtering and grouping mechanism enables easily monitoring specific Ice components and specify the level of detail for monitoring those objects. It is for example possible to only monitor the connections of a given object adapter. You can also decide to monitor the connections of this object adapter individually or group them based on some attributes of the connection such as the remote host of the connection or its protocol.

Using the IceGrid graphical administrative client, you can dynamically inspect deployed applications and create graphs of their activity in real time.

The Metrics facility is also supported by the Glacier2 and IceStorm services enabling monitoring Glacier2 sessions, IceStorm topics and subscribers.

Optional data members and parameters

To simplify the task of enhancing an Ice application over time, Slice classes and exceptions can now declare optional data members, and Slice interfaces can now define operations with optional parameters. The new syntax for this feature is quite simple:

Slice

```
class Account
{
    string accountNo;
    ...
    optional(1) string guardian;
    optional(2) string linkedAccountNo;
};

interface Bank
{
    Account createAccount(string name, optional(3) string guardian, ...);
};
```

Each optional member or parameter must be assigned a unique integer identifier. Ice only marshals an optional member or parameter if a value is supplied, and the receiver can test to determine whether a value is set.

Despite the simple changes in syntax, the implications for application developers are significant. It is now possible to add or remove optional data members and parameters and deploy a new version of an application without affecting backward compatibility with existing deployments.

Compact encoding for classes and exceptions

The marshaled format of Slice classes and exceptions now uses a new *compact* format by default that consumes much less space than in previous releases. The *sliced* format, which provides similar functionality to the original encoding, can still be enabled using a configuration property or metadata directive. When using the compact format, the receiver must know the most-derived type of the value; this new encoding does not include the information necessary for the receiver to "slice" the value to a known less-derived type. If your application requires this feature, you must use the sliced format instead.

Compact type IDs

You can now optionally associate a numeric identifier with a class. The Ice run time substitutes this value, known as a *compact type ID*, in place of its equivalent string type ID during marshaling to conserve space. The compact type ID follows immediately after the class name, enclosed in parentheses:

Slice

```
class CompactExample(4)
{
    // ...
};
```

Preserved slices

A limitation in version 1.0 of the Ice encoding makes it cumbersome for an intermediary server to forward an instance of a Slice class or exception from one peer to another. The intermediary must know all of the derived types that it could potentially encounter, otherwise the Ice run time in the intermediary would "slice" the value to a known less-derived type and consequently interfere with the data exchange. Ideally, the intermediary should only be required to know the base types, while still being capable of forwarding unknown derived types between the peers without slicing.

Version 1.1 of the Ice encoding adds the ability to transfer classes and exceptions with all of their slices intact using the new metadata tag `preserve-slice`. This feature can only be used with the new sliced encoding format.

Custom enumerator values

The Slice syntax for enumerations now allows you to specify custom values for enumerators:

Slice

```
enum Color
{
    red = 10,
    green,    // Automatically assigned value 11
    blue = 15
};
```

C++11 lambda functions

With a supported compiler, Ice 3.5 allows you to use C++11 lambda functions for AMI and dispatcher callbacks. This feature is enabled by default for Visual Studio 2010 SP1 and Visual Studio 2012 in the Ice installer for Windows and for the C++11 libraries in the Ice installer for OS X. For all other platforms and compilers, Ice must be recompiled with C++11 support enabled.

Remote Update of Server Properties

You can now use the Ice.Admin Properties facet to update a server's Ice properties remotely. IceGrid leverages this new feature by allowing you to update a server's properties *without restart*: the server gets the new properties from IceGrid and you decide if or when to restart your server.

IPv6 now enabled by default

The support for IPv6 is now enabled by default. A server listening on the wildcard address will accept both IPv4 and IPv6 TCP/IP connections by default (except for Windows XP and Windows 2003 which do not support a dual stack IP implementation). When resolving hostnames, for backward compatibility, Ice will still prefer using the IPv4 address when the hostname resolves to both an IPv4 and an IPv6 address. This behavior can be changed with the new Ice.PreferIPv6Address property.

Windows 8 and Windows RT

Windows 8 and Windows RT are now fully supported platforms.

In addition, Ice 3.5 supports the WinRT API to enable building Windows Store applications.

Ice 3.5 compiled to target the WinRT API has some limitations compared to the regular Windows version. The following features are not available:

- Protocol compression
- Per-thread implicit contexts
- Server-side SSL
- Client SSL certificates
- IceBox server
- IcePatch2 client library
- `Ice::Service`, `Ice::Application` and `Glacier2::Application` classes
- Accessing properties from Windows registry

- `ICE_CONFIG` environment variable

The Ice installer for Windows includes a Visual Studio SDK for WinRT (with static libraries for the x86, x86_64 and ARM architectures). The Ice Visual Studio Add-in automatically references this SDK for WinRT projects.

Ice 3.5.1 adds support for Windows 8.1.

Silverlight

Ice 3.5 includes support for Silverlight, which was previously offered as a ZeroC Labs project. Limitations in the Silverlight API mean that some features of Ice for .NET are unavailable when using Ice for Silverlight.

Python 3

The Ice extension for Python adds support for Python 3.x while still maintaining compatibility with 2.x. The minimum supported version is now Python 2.6.

The Ice extension included in the Ice installer for Windows requires Python 3.3.

The binary distributions for Linux and Mac include a Python extension that was built against (and requires) the default Python version for that platform. To use the Ice extension with a different Python version, you will need to obtain the Ice source code and compile a new version of the extension.

If you are migrating an Ice application from Python 2.x to Python 3.x, please refer to the upgrade instructions for more information.

Ruby 1.9

The Ice extension for Ruby adds support for Ruby 1.9 while still maintaining source compatibility with 1.8.x.

For compatibility with the Ruby Installer for Windows, the Ice extension included in the Ice installer for Windows is compiled with MinGW and requires Ruby 1.9.x.

The binary distributions for Linux and Mac include a Ruby extension that was built against (and requires) the default Ruby version for that platform. To use the Ice extension with a different Ruby version, you will need to download the Ice source code and compile a new version of the extension.