Release Notes

Ice Touch offers C++ and Objective-C SDKs for building iOS and Cocoa applications. Ice Touch also includes an Objective-C run time for use in Mac OS X applications.

The Ice Touch distribution does not include any Ice services, but its support for the complete Ice protocol means that your Ice Touch applications can work seamlessly with existing Ice servers as well as Ice services such as IceGrid, Glacier2, and IceStorm.

On this page:

Back to Top ^

New features in Ice Touch 1.2

This section outlines changes and improvements in this release 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 CHANGES file included in your Ice Touch distribution.

Features added in Ice Touch 1.2.2

Support for Xcode 4.5 and iOS 6.0 was added in Ice Touch 1.2.2. Ice Touch no longer supports iOS 4.2.x.

Features added in Ice Touch 1.2.1

Support for Xcode 4.3 was added in Ice Touch 1.2.1. Ice Touch is no longer installed in the `/Developer` directory since Xcode 4.3 is now installed as an application bundle in `/Applications`. Ice Touch is now installed in `/Library/Developer/IceTouch-1.2.1`. The installer also creates the symbolic link `/Library/Developer/IceTouch-1.2` to point to the latest Ice Touch 1.2 patch release.

Features added in Ice Touch 1.2.0

C++ SDKs

Ice Touch includes C++ SDKs for Xcode that are intended for use in graphical C++ applications targeting iOS and Cocoa. Developers of command-line applications for Mac OS should continue to use our standard Ice for C++ distribution.

Apple LLVM Support

The Ice Touch binary distribution is compiled with Apple LLVM (clang).

Changes and fixes in Ice Touch 1.2.1

- Ice Touch now requires Xcode 4.3.

- A memory leak in the Objective-C language mapping where Slice generated structs or classes wouldn't correctly release data members was fixed.

Changes and fixes in Ice Touch 1.2.0

- Ice Touch now requires Xcode 4.2.
- Updated Ice Touch to support iOS 5.
- Added a C++ implementation of `Glacier2::SessionHelper`, which simplifies the use of Glacier2 in graphical applications.

Back to Top ^

Corresponding Ice release

The Slice definitions included in Ice Touch 1.2 are the same as the Slice definitions included in Ice 3.4.2. In particular, the Glacier 2 client libraries included in this Ice Touch release (`libGlacier2ObjC.a`, `libGlacier2ObjC.11.dylib`, `libGlacier2Cpp.a`) use the Ice 3.4.2 Glacier2 definitions. If the Glacier2 service in a future Ice release adds new APIs (such as a new operation, or a new interface), you will need to rebuild these libraries using the newer Glacier2 Slice definitions in order to use these APIs.

Back to Top ^

Upgrading your application from Ice Touch 1.2.0

Xcode project settings

For Xcode iOS and Cocoa applications, you need to update the project property `ADDITIONAL_SDKS` to match the location of the new Ice Touch SDK installation. You should use:

- `/Library/Developer/IceTouch-1.2/SDKs/ObjC/$(PLATFORM_NAME).sdk` instead of `/Developer/SDKs/IceTouch-1.2/$(PLATFORM_NAME).sdk` for the Objective-C SDK.
- `/Library/Developer/IceTouch-1.2/SDKs/Cpp/$(PLATFORM_NAME).sdk` instead of `/Developer/SDKs/IceTouchCpp-1.2/$(PLATFORM_NAME).sdk` for the C++ SDK.

Command line SDK settings

The command line SDK is no longer installed in `/opt`. It is now installed in `/Library/Developer/IceTouch-1.2.1` and the install name of the Ice Touch shared libraries is prefixed with the path `/Library/Developer/IceTouch-1.2`. You will need to change your build system accordingly.

Source installation

If you are building Ice Touch 1.2 from sources, you need to manually remove the Xcode plug-in installed by Ice Touch 1.0 prior to the Ice Touch 1.2.0 installation. To do so, remove the directory containing this plug-in, which can be either `/Developer/Library/Xcode/Plug-ins/slice2objcplugin.pbplugin` or `~/Library/Application Support/Developer/Shared/Xcode/Plug-ins/slice2objcplugin.pbplugin`.

Back to Top ^

Upgrading your application from Ice Touch 1.1

ZeroC does not guarantee binary compatibility between Ice Touch 1.1 and Ice Touch 1.2, therefore you must recompile your Slice files and rebuild your application.

If you are building Ice Touch 1.2 from sources, you need to manually remove the Xcode plug-in installed by Ice Touch 1.1 prior to the Ice Touch 1.2.0 installation. To do so, remove the directory containing this plug-in, which can be either `/Developer/Library/Xcode/Plug-ins/slice2objcplugin.pbplugin` or `~/Library/Application Support/Developer/Shared/Xcode/Plug-ins/slice2objcplugin.pbplugin`.

Back to Top ^

Upgrading your application from Ice Touch 1.0

Xcode project settings

For Xcode iOS and Cocoa applications, you need to update the project property `ADDITIONAL_SDKS` to match the location of the new Ice Touch SDK installation. If you installed the Ice Touch 1.2.0 SDK in the default location, you would use `/Developer/SDKs/IceTouch-1.2/iphoneos.sdk` instead of `/Developer/SDKs/IceTouch-1.0/iphoneos.sdk`.

Source installation

If you are building Ice Touch 1.2 from sources, you need to manually remove the Xcode plug-in installed by Ice Touch 1.0 prior to the Ice Touch 1.2.0 installation. To do so, remove the directory containing this plug-in, which can be either `/Developer/Library/Xcode/Plug-ins/slice2objcplugin.pbplugin` or `~/Library/Application Support/Developer/Shared/Xcode/Plug-ins/slice2objcplugin.pbplugin`.

Back to Top ^

Ice Touch feature set

Ice Touch supports the following features:

- Objective-C mapping
- C++ mapping

Ice Touch currently lacks support for the following Ice features:

- Protocol plug-ins
- Local interfaces
- `Ice::Application` and `Ice::Service` helper classes
- `Ice::Stats` interface

The Objective-C mapping currently lacks support for the following Ice features:

- Asynchronous method dispatch (AMD)
- Collocation optimization
- Servant locators
- Implicit contexts

Ice Touch has limited support for:

- SSL
  See below for more information.

- UDP
  On iPhone, UDP requests do not transparently establish a 3G/Edge connection.

Back to Top ^

SSL support

Ice Touch for Mac OS X and Cocoa uses the Ice for C++ SSL protocol plug-in.

For iOS devices, Ice Touch SSL provides only a subset of this functionality. Due to limitations in iOS SSL support, the following restrictions apply:

- Ice Touch servers cannot authenticate SSL clients.

- iOS 5 clients reject server certificates that use MD5 hashes.

Furthermore, the semantics of some IceSSL configuration properties have changed, and new properties have been added. The IceSSL property reference provides complete details.

Back to Top ^

Logger notes

Objective-C applications must install custom loggers via `ICEInitializationData`. You cannot use any of the Ice for C++ logger properties, such as `Ice.UseSyslog`, and you cannot install a custom logger with a plug-in (`Ice.Plugin.*`).

Back to Top ^

Garbage collection requirements

For Mac OS X and Cocoa, the Ice Touch run time is built with garbage collection support.

For iOS devices and the simulator, you must use explicit `retain` and `release` because garbage collection is not supported. Consequently, the Ice Touch run time for iOS cannot garbage collect graphs of objects containing cycles. Consider this Slice definition:

Slice

```
class Foo
{
Foo next;
};
```

Suppose we use these definitions as follows:

Objective-C

```
Foo a = [[Foo alloc] init];
Foo b = [[Foo alloc] init];
a.next = b;
b.next = a;
```

If you send this graph over the wire, your application will leak memory unless you somehow retain the graph and manually break the cycle.

Back to Top ^

Auto release pool

The Ice run time creates an `NSAutoReleasePool` object before each server-side dispatched invocation and client-side AMI callback. The pool is released once the dispatch is complete.

Back to Top ^

Accessory transport

The accessory transport enables Ice Touch clients running on an iOS device to communicate with accessories connected either via Bluetooth or USB.

This transport is built on top of the iPhone OS External Accessory framework. In order to use it, proxies must use the following endpoint syntax:

`accessory [-p` *`PROTOCOL`*`] [-n` *`NAME`*`] [-m` *`MANUFACTURER`*`] [-o` *`MODELNUMBER`*`]`

For example, to invoke on a proxy for the `hello` object running on an accessory that implements the `com.zeroc.helloWorld` protocol, use the following stringified proxy:

```
hello:accessory -p com.zeroc.helloWorld
```

If the `-p` option is omitted, the transport will look up an accessory that supports the `com.zeroc.ice` protocol by default. This protocol string is application-defined and must match the protocol string that the accessory advertises.

In order to enable the accessory transport, Ice Touch applications must add properties to the Ice communicator initialization property set using the `ICEConfigureAccessoryTransport` method.

For example:

Objective-C

```
ICEInitializationData* initData = [ICEInitializationData initializationData];
initData.properties = [ICEUtil createProperties];
ICEConfigureAccessoryTransport(initData.properties);
communicator = [[ICEUtil createCommunicator:initData] retain];
```

This call will add the required properties and ensure that the accessory transport is linked in with your application. The project for your application will need to include the `ExternalAccessory` framework as well.

To specify that your application supports a given accessory protocol, you need to set `UISupportedExternalAccessoryProtocols` in your project `Info.plist` file as demonstrated below:

XML

```
<key>UISupportedExternalAccessoryProtocols</key>
<array>
<string>com.zeroc.helloWorld</string>
</array>
```

The iPhone hello world demo from the `demo/iPhone/hello` directory demonstrates how to configure the accessory transport.

In order to use the Ice Touch accessory transport, your accessory must also support running an Ice server that is capable of receiving Ice requests from Ice Touch over USB or Bluetooth. We can assist you with the implementation of the Ice server-side transport for your accessory. For more information on this, please contact us at info@zeroc.com.

Back to Top ^

Targeting iOS >= 4.3

Xcode 4.5 targets iOS 6 by default. To build your application for an iOS >= 4.3 target, you will need to modify your Xcode project as follows:

- Set the Xcode `Deployment Target` property to the desired iOS version.