Xcode Plug-in

The Xcode plug-in manages all aspects of code generation, including automatically recompiling Slice files that have changed, removing obsolete generated classes, and tracking dependencies. The Xcode plugin is provided with the Ice Touch installer.

On this page:

Adding Slice Files to an Xcode Project

To add an existing Slice file, select a folder in the project, select *File*, and choose *Add -> Existing Files...*

To create a new Slice file, select a folder in the project, select *File*, and choose *New -> New Files...* Select the *Other* category and choose *Empty File* as the file type. Save the file with a `.ice` extension.

Configuring Xcode Project Settings

The Xcode plug-in is configured using the per-target info build settings, just as you would configure the compiler settings. Select a target, and then select the build settings tab, and enter `Slice` in the *Search in Build Settings* field.

- **Header Directory**
  Directory to use as the header include directory in source files.
- **Header Search Paths**
  The list of directories to search for included Slice files (`-I` option). Note that the Ice Slice files are automatically in the header search path.
- **Ice Home**
  If you are building with the Ice Touch Xcode SDK, this should be left unset; otherwise, if using the command-line toolkit or regular Ice, set this to the location of the installation (`/Library/Developer/Ice-3.5.0` or `/Library/Developer/IceTouch-1.3.0`).
- **Output Directory**
  Directory to place files generated by the Slice compiler.
- **Permit `Ice' prefix**
  Pass `--ice` to the Slice compiler.
- **Streaming support**
  Pass `--stream` to the Slice compiler (C++ only).
- **Checksum support**
  Pass `--checksum` to the Slice compiler (C++ only).
- **Preprocessor Macros**
  Set the list of preprocessor macros to define (`-D` option).
- **Translate C++ code**
  Leave this option unset for an SDK build. For non-SDK builds, setting this option means `slice2cpp` is used to compile the Slice files; otherwise, `slice2objc` is used.
- **Underscore support**
  Permit underscores in Slice identifiers.
- **Link with Ice services client libraries**
  Link with Glacier2, IceStorm and IceGrid client libraries.

You can also define Slice compiler options for individual Slice source files. Select your project in the Project Navigator, select the relevant target (you may have only one), then select the *Build Phases* tab, expand the *Compile Sources* phase and the *Compiler Flags* column lets you set each file's options for that target.

Xcode Project Settings for Cocoa and iPhone Applications

For Cocoa and iPhone applications, which use the Xcode SDK, you must add the appropriate directory to *Additional SDKs*:

Objective-C SDK	`/Library/Developer/IceTouch-1.3/SDKs/ObjC/$(PLATFORM_NAME).sdk`
C++ SDK	`/Library/Developer/IceTouch-1.3/SDKs/Cpp/$(PLATFORM_NAME).sdk`

In addition, when creating a new iPhone Xcode project, you must set the Code Signing Resource Rules Path to:

```
$(SDKROOT)/ResourceRules.plist
```

You must also add the following to the `Frameworks` folder:

```
CFNetwork.framework
Security.framework
Foundation.framework
```

When using the Objective-C SDK you must also add the following to the `Frameworks` folder:

```
ExternalAccessory.framework
```

Configuring Non-SDK Builds

For non-SDK builds, you must configure the location of the Ice or Ice Touch header and library directories.

Under the *Search Paths* section in the project build settings:

- Add `/Library/Developer/Ice-3.5.0/include` for Ice or `/Library/Developer/IceTouch-1.3.0/include` to the *Header Search Paths* setting.
- Add `/Library/Developer/Ice-3.5.0/lib` for Ice or `/Library/Developer/IceTouch-1.3.0/lib` to the *Library Search Paths* sett ing.

Generating Code using Xcode

The plug-in compiles a Slice file whenever you build the project. The extension tracks dependencies among Slice files in the project and recompiles only those files that require it after a change.

See Also

- slice2objc Command-Line Options