

## Guide to configuring Xcode to use the CuteiOS Platform Plugin

This guide will describe the steps required to configure an Xcode project, created by the Qt Project's Qt SDK for iOS, to use the 'CuteiOS' Platform Plugin for iOS.

The Qt SDK for iOS is assumed to be installed in the default location,

~/Qt5.2.x/5.2.x/ios (where **x** is the Qt5.2 revision number) or

~/Qt5.3.x/5.3.x/ios (where **x** is the Qt5.3 revision number). The paths specified in this guide will need to be modified if another install location is used.

### What is included?

The CuteiOS Qt Platform Plugin for iOS contains various plugins to enable the Qt framework to be used when building applications for Apple iOS devices and the Apple iOS simulator.

The Qt5.2/5.3 'CuteiOS' plugin supports many features that are currently not supported by the (default) 'iOS' plugin: switchable raster/OpenGL painting, full support for high-DPI (aka. 'Retina') displays, enhanced OpenGL support, enhanced keyboard support, amongst others.

The evaluation version of the 'CuteiOS' plugin is fully functional, but displays a 'watermark' on the screen. The 'watermark' will prevent the application from being accepted for distribution on Apple's App Store. To distribute applications on the App Store, either the 'CuteiOS' plugin must be licensed (from Mediator Software - [licensing@mediator-software.com](mailto:licensing@mediator-software.com)), or the 'iOS' plugin must be used. The demo/evaluation version of the 'CuteiOS' plugin is not licensed for distribution.

The current CuteiOS Qt Platform Plugin for iOS is built for ARMv7 compatible iOS devices with OpenGL/ES 2.x compatible graphics accelerators, running iOS 4.0 or later. Currently this includes all iPad devices, all 3<sup>rd</sup> generation and later iPod devices and all iPhone 3GS or later iPhone devices. In addition, the CuteiOS Qt Platform Plugin for iOS is distributed as 'Universal' libraries that contain the necessary binaries for running Qt applications on the iOS simulator.

### 1) Install the 'CuteiOS' platform plugin

The 'CuteiOS' platform plugin for Qt 5.2/5.3 is shipped as standalone static library (.a) files. To install the plugin, open 'Terminal' and type the following:

```
unzip -x libqcuteios-5.2.x.zip -d ~/Qt5.2.x/5.2.x/ios/plugins/platforms
```

or

```
unzip -x libqcuteios-5.3.x.zip -d ~/Qt5.3.x/5.3.x/ios/plugins/platforms
```

This will install the 'CuteiOS' platform plugin in the Qt SDK for iOS platform plugin folder.

## 2) Use the Qt SDK build tools to generate an Xcode project

The Qt SDK for iOS build system generates Xcode project files from **.PRO** project files by default when building for iOS. The Qt SDK build tools will automatically configure the Xcode project to use the 'iOS' platform plugin, and so users of the 'CuteiOS' platform plugin will need to edit the Xcode project each time it is regenerated.

The first step is to use **qmake** to generate the makefiles for the project. In order to do this, the version of **qmake** that ships with the Qt for iOS SDK must be used. To do this (for example: using the **animatedtiles.pro** file) open 'Terminal' and type the following:

```
~/Qt5.2.x/5.2.x/ios/bin/qmake animatedtiles.pro
```

or

```
~/Qt5.3.x/5.3.x/ios/bin/qmake animatedtiles.pro
```

NOTE: to avoid having to type the full path to the **qmake** executable every time, the directory containing **qmake** above can be added to the **PATH** environment variable.

Once **qmake** has generated the project's makefiles, the application project can be opened in Xcode. To do this (using the above example) open 'Terminal' and type the following:

```
open animatedtiles.xcodeproj
```

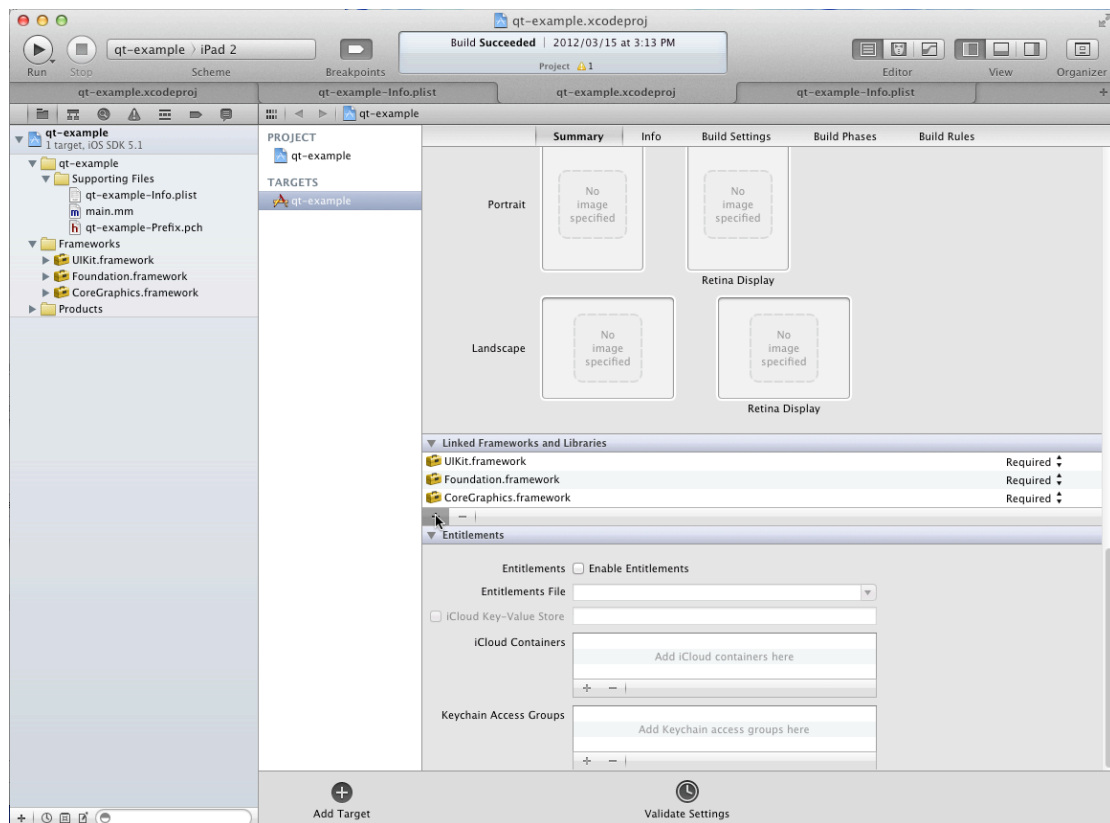
## 3) Configure the Xcode project

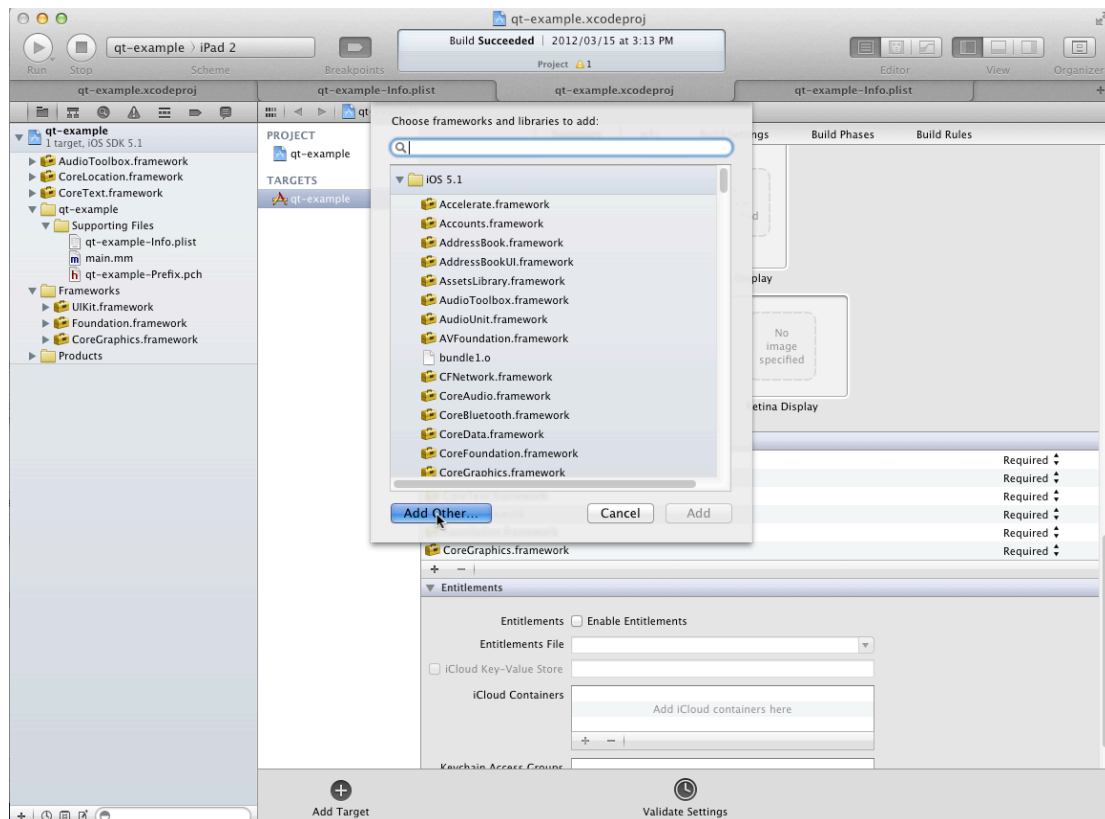
To configure the project to use the 'CuteiOS' platform plugin instead of the (default) 'iOS' platform plugin, the **qios** file must first be removed from the 'Frameworks' section of the project. Then the project can be configured as described below:

### Add 'CuteiOS' platform plugin

A Qt application needs to be linked against the Qt libraries, the iOS platform plugin for Qt, and the Apple libraries and frameworks that the platform plugin depends on. Which Qt and Apple libraries are required is determined by the **qmake** utility and added to the Xcode project when it is generated. There are 2 different versions of the 'CuteiOS' platform plugin: **libqcuteios.a** - has dependencies on **QtWidgets (libQt5Widgets.a)** and so should be used for widget-based applications, and **libqcuteios-qml2.a** - has dependencies on **QtDeclarative (libQt5Quick.a)** and so should be used for QML2-based applications.

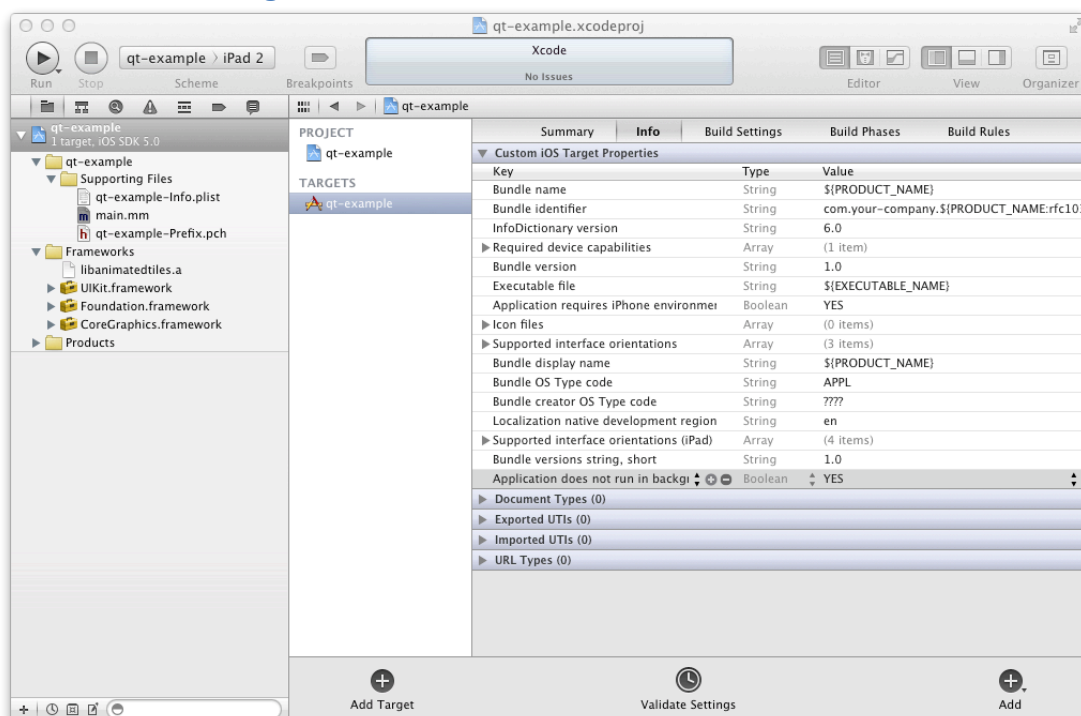
The appropriate 'CuteiOS' platform plugin for the type of application being developed must be added as follows:





The location of the platform plugin is the folder where it was installed in the previous section (where ~ refers to the 'Home' directory in OSX).

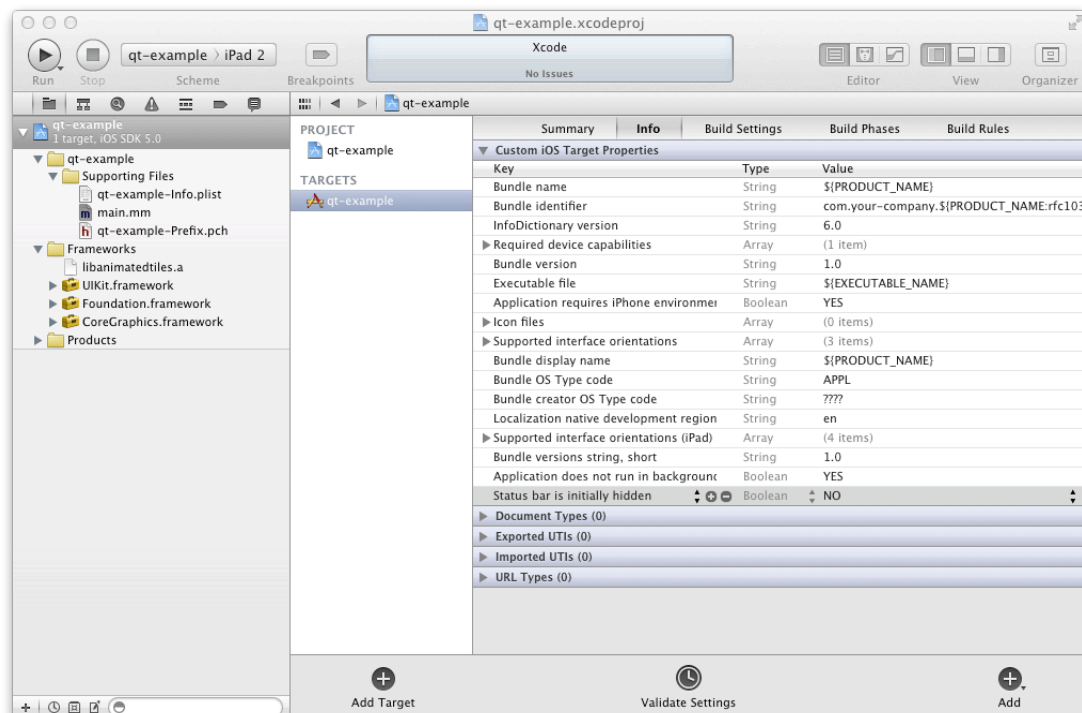
## Enable/disable background execution



By default, applications built for iOS with Xcode will continue to run in the background when the 'Home' button is pressed on the iOS device. If this behavior is not desirable, the 'Application does not run in background' key must be added

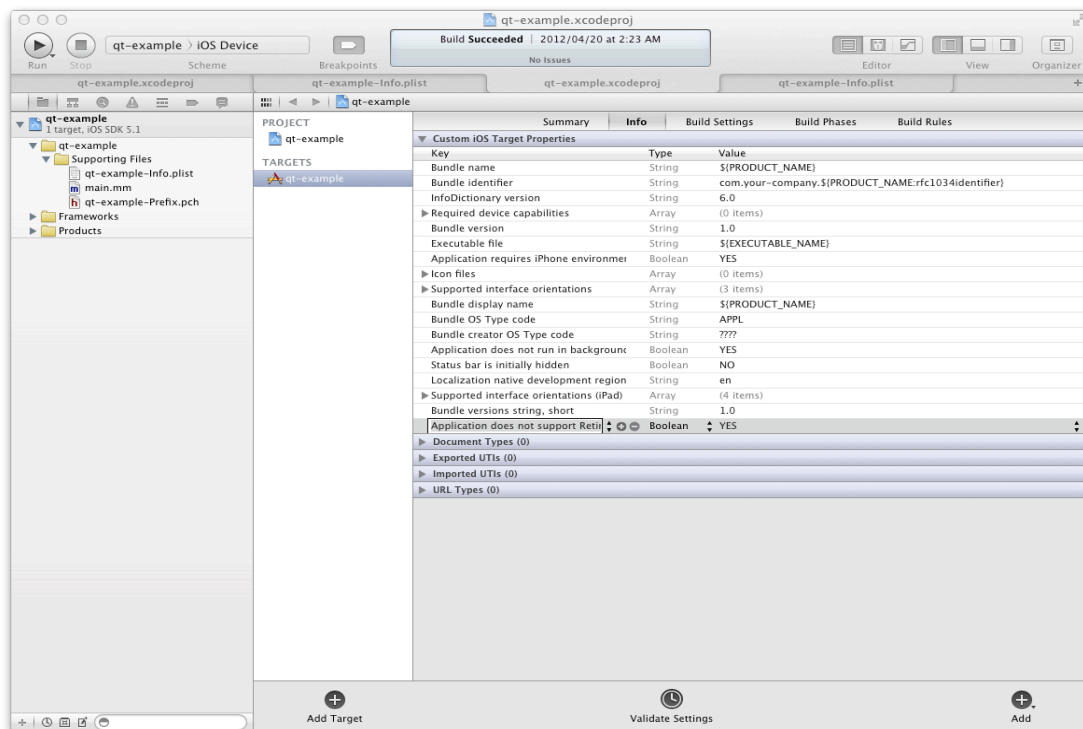
to the 'Custom iOS Target Properties' in Xcode, as shown above.

### Enable/disable iOS status bar



By default, applications built for iOS with Xcode will run fullscreen and will not show the iOS status bar. If this behavior is not desirable, the 'Status bar is initially hidden' key must be added to the 'Custom iOS Target Properties' in Xcode, as shown above.

## Configuring the 'CuteiOS' platform plugin



### Enable/disable 'Retina' display support

By default, applications built for iOS with Xcode will run at the full resolution of 'Retina' displays on iPhone/iPod/iPad devices. If this behavior is not desirable, a custom key 'Application does not support Retina display' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'.

### Enable/disable 'High DPI' rendering

By default, applications built for iOS with Xcode will use 'High DPI' rendering when 'Retina' display support is enabled. This allows high resolution pixmaps and text to be rendered without changing the apparent resolution of the display (from a non-'Retina' display). If this behavior is not desirable, and the full resolution of the display is required, a custom key 'Renders with high DPI' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'.

### Enable/disable OpenGL rendering

By default, applications built for iOS with Xcode will use an OpenGL painter for drawing. The OpenGL painter is hardware-accelerated and so provides a good mix of performance and accuracy. In some cases, it may be preferable to use a raster painter for drawing (either for improved performance in some scenarios, or improved rendering accuracy). To switch between OpenGL and raster painting, a custom key 'Renders with OpenGL' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'.

### Enable/disable 'retained' backing stores

By default, applications built for iOS with Xcode will use a 'retained' backing

store when using an OpenGL painter for drawing. This means that the platform plugin will 'retain' the contents of the display in between display updates. This can improve performance (at the cost of using more memory), but may cause rendering artifacts when using transparent windows. If this behavior is not desirable, and the contents of the display should be redrawn on every display update, a custom key 'Renders with retained backing' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'.

#### ***Enable/disable multi-threaded event processing***

By default, the 'CuteiOS' platform plugin runs the Qt and UIKit event loops in the same thread (using QPA event loop integration). It may be necessary to use the platform plugin in single-threaded mode in an application that has static QObject classes, as these would not be initialised in the main Qt thread when using the platform plugin in multi-threaded mode. In many other circumstances, it may be preferable for performance or compatibility reasons to run the event loops in separate threads. To switch between single- and multi-threaded modes, a custom key 'Application does not support multiple event dispatchers' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'.

#### ***Enable/disable multi-touch***

By default the 'CuteiOS' platform plugin reports all touch events to Qt. In some instances, for compatibility, it may be preferable to simulate a mouse-type pointing device and only allow a single touch event (which is reported to Qt as a mouse event). To enable/disable multi-touch, a custom key 'Application does not support multiple touch' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'.

#### ***Enable/disable transparent status bar [iOS7.0+ only]***

By default the 'CuteiOS' platform plugin emulates the status bar of iOS6 and earlier when running on an iOS7.0+ device for maximum compatibility. To enable/disable the default transparent status bar on iOS7.0+ devices, a custom key 'Status bar is transparent' must be created and added to the 'Custom iOS Target Properties' in Xcode, as shown above. The key type must be set to 'Boolean'. When using a transparent status bar, it may be necessary to specify a status bar colouring that matches the application content. This can be done by adding a 'Status bar style' key to the 'Custom iOS Target Properties' in Xcode, and setting it to a suitable value.

The Xcode project can now be used to build, package and sign the Qt application for deployment to iOS devices.

### **Where to find the CuteiOS Qt Platform Plugin for iOS**

The latest CuteiOS Qt Platform Plugin for iOS SDK releases and the latest Qt for iOS SDKs for older Qt versions (4.8 - 5.1) can be downloaded from:

<http://www.mediator-software.com>

For further information about licensing the 'CuteiOS' platform plugin, please contact: [licensing@mediator-software.com](mailto:licensing@mediator-software.com)