AdWhirl Open Source SDK Setup Instructions for iPhone

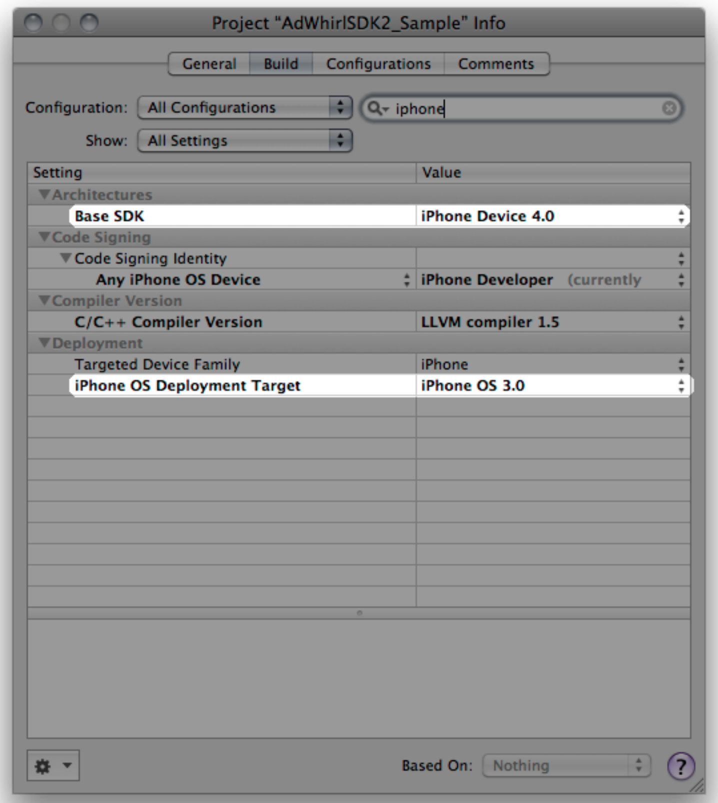
The AdWhirl Open Source SDK contains the code for your iPhone application to display ads from different ad networks. The instructions below assume that you are familiar with Xcode and understand how to change build settings and add frameworks to your Xcode projects.

Note: When adding files to your Xcode project, make sure "Copy items into destination group's folder (if needed)" is checked and use "Recursively create groups for any added folders".

Base SDK and Devices With Older OS Versions

To be consistent with Apple's published best practices, you should set the "Base SDK" setting of your project to the latest iPhone SDK when possible. To support users with older iPhone OSes, set the "iPhone OS Deployment Target" setting to an earlier version. For example, if the latest iPhone SDK is 4.0 and you need to support users with iPhone SDK 3.0, set the "Base SDK" to "iPhone OS 4.0" and "iPhone OS Deployment Target" to "iPhone OS 3.0"

Note that as of June 2010, some ad network libraries may not work well with projects built with iPhone OS 4.0 . If you want to use those ad networks, you must still keep the Base SDK of your project to iPhone OS 3.x . For a list of ad network SDKs that are tested to work with iPhone OS 4.0, see the wiki page [AdNetworkSDKCompatibilityMatrix]() in http://code.google.com/p/adwhirl/wiki .



Configuration: All Configurations

iphone

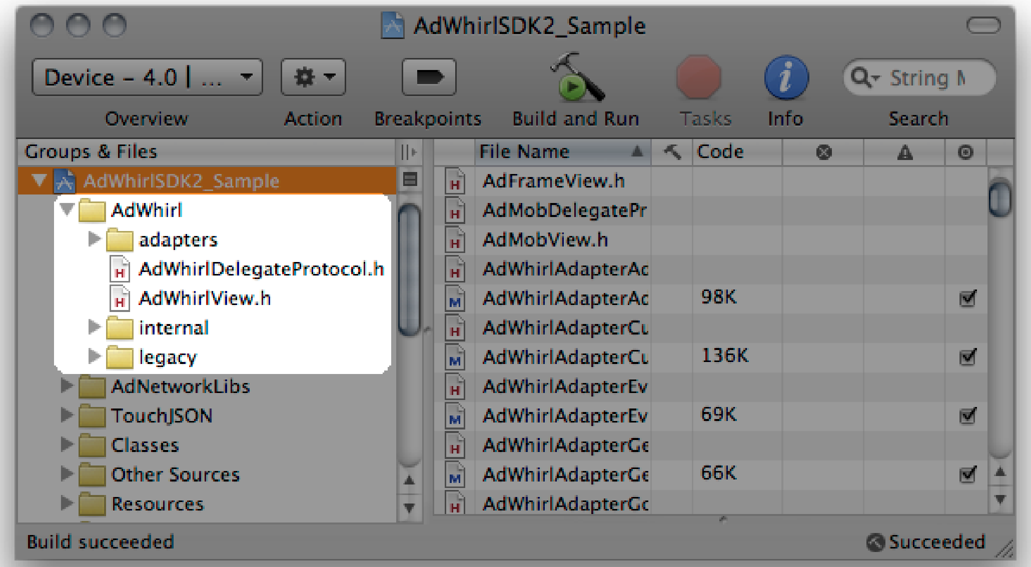
Show: All Settings

Setting	Value
Architectures	
Base SDK	iPhone Device 4.0
Code Signing	
Code Signing Identity	
Any iPhone OS Device	iPhone Developer (currently
Compiler Version	
C/C++ Compiler Version	LLVM compiler 1.5
Deployment	
Targeted Device Family	iPhone
iPhone OS Deployment Target	iPhone OS 3.0

Based On: Nothing

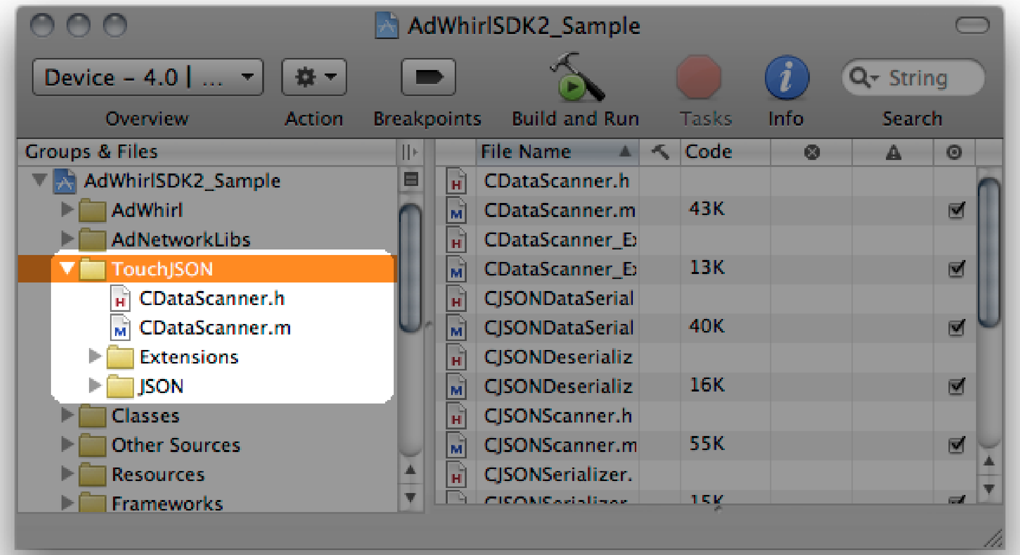
Setting up the AdWhirl SDK

1. Add the AdWhirl folder from the AdWhirl SDK into your project. In the AdWhirl folder, you'll find two files (`AdWhirlDelegateProtocol.h`, `AdWhirlView.h`) and three folders (`adapters/`, `internal/`, `legacy/`).



File Name	Code
AdFrameView.h	
AdMobDelegatePr	
AdMobView.h	
AdWhirlAdapterAc	
AdWhirlAdapterAc	98K
AdWhirlAdapterCu	
AdWhirlAdapterCu	136K
AdWhirlAdapterEv	
AdWhirlAdapterEv	69K
AdWhirlAdapterGe	
AdWhirlAdapterGe	66K
AdWhirlAdapterGo	

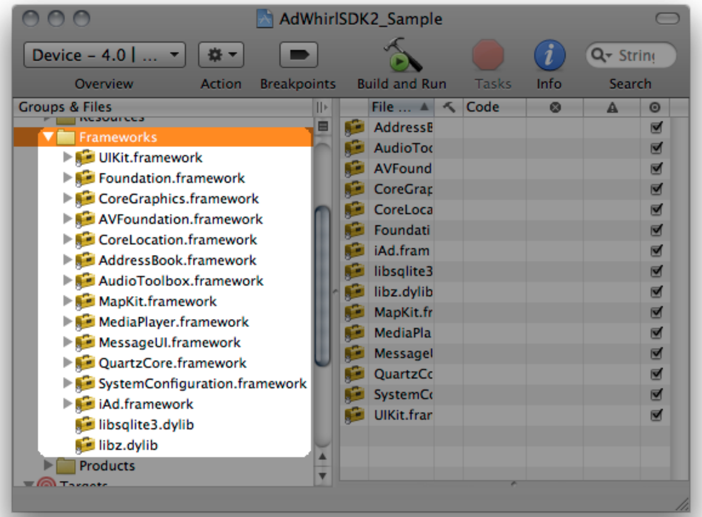
2. Add TouchJSON into your project.



File Name	Code
CDataScanner.h	
CDataScanner.m	43K
CDataScanner_Ex	
CDataScanner_Ex	13K
CJSONDataSerial	
CJSONDataSerial	40K
CJSONDeserializ	
CJSONDeserializ	16K
CJSONScanner.h	
CJSONScanner.m	55K
CJSONSerializer.	
CJSONSerializer	15K

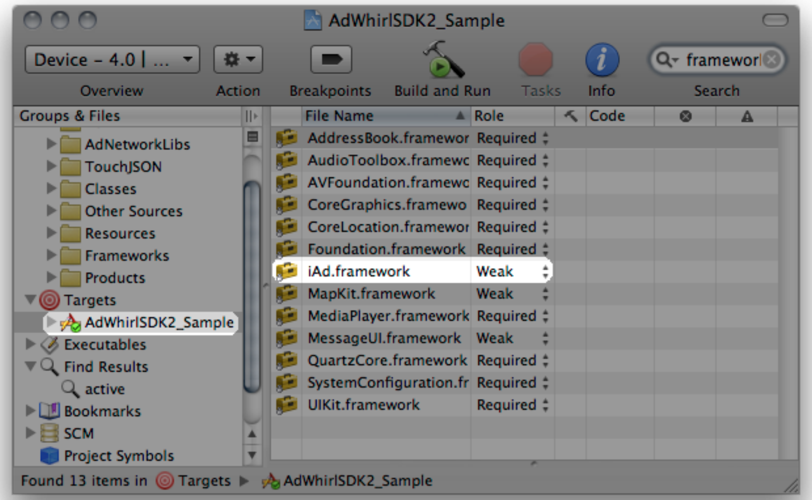
3. Add the supporting frameworks required by all of the supported ad networks. The frameworks will not affect the size of your executable unless they are used by ad network libraries that you integrate.

- AddressBook.framework
- AudioToolbox.framework
- AVFoundation.framework
- CoreLocation.framework
- MapKit.framework
  - weak-linked for OS 2.X support
- MediaPlayer.framework
- MessageUI.framework
  - weak-linked for OS 2.X support
- QuartzCore.framework
- SystemConfiguration.framework
- iAd.framework
  - required for iAd, weak-linked for OS 3.X support
- libsqlite3.dylib
- libz.dylib



UIKit.framework, Foundation.framework and CoreGraphics.framework should be included in your project by default.

Some frameworks may need to be weak-linked to support older iPhone OSes. You can weak-link a framework by selecting your application's target, on the left pane, change the Role of the framework from "Required" to "Weak".

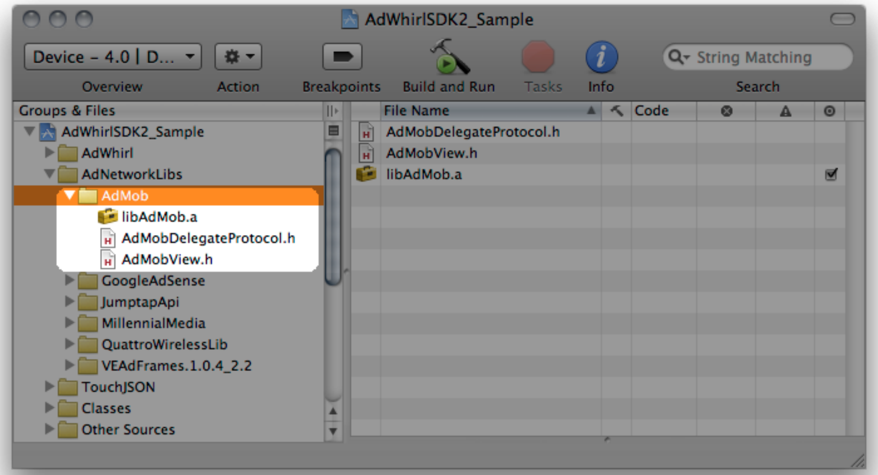


File Name	Role
AddressBook.framewor	Required
AudioToolbox.framewc	Required
AVFoundation.framewc	Required
CoreGraphics.framewo	Required
CoreLocation.framewor	Required
Foundation.framework	Required
iAd.framework	Weak
MapKit.framework	Weak
MediaPlayer.framework	Required
MessageUI.framework	Weak
QuartzCore.framework	Required
SystemConfiguration.fr	Required
UIKit.framework	Required

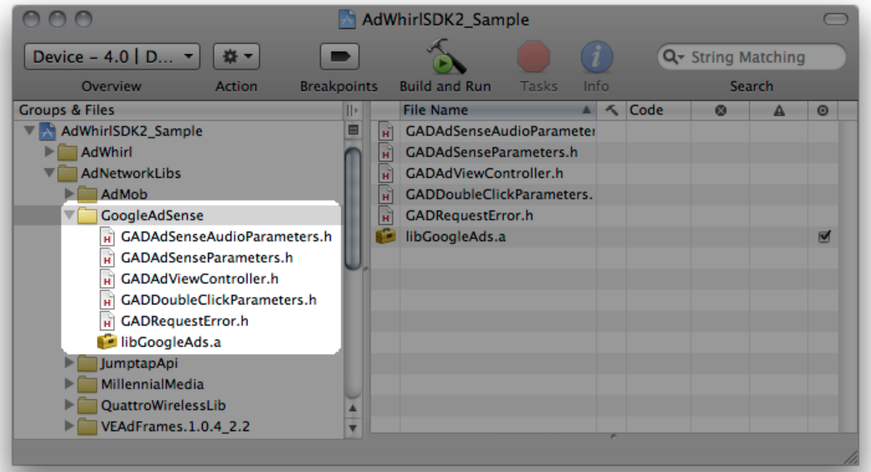
Found 13 items in Targets ▶ AdWhirlSDK2_Sample

4. Add the ad network libraries. All of the ad network libraries are optional. You do not have to integrate an ad network if you do not want to run their ads, or do not have access to their SDK.

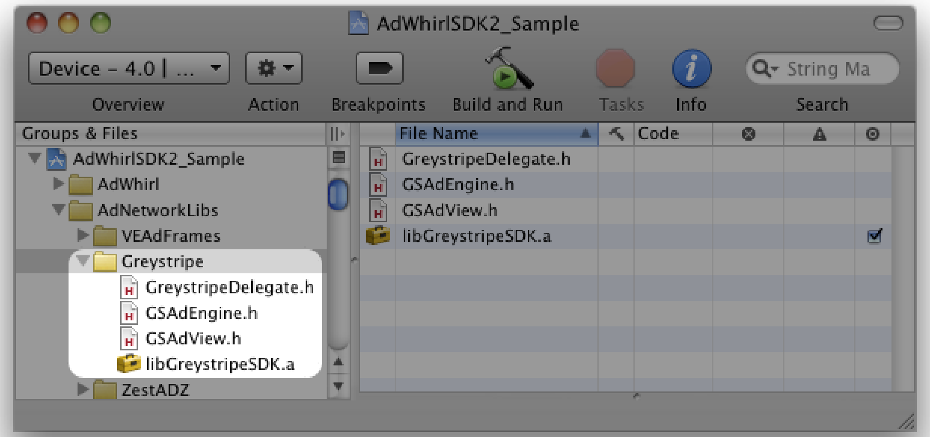
- **AdMob**
  1. Download the AdMob SDK from http://www.admob.com/ . AdMob provides a universal binary for all iPhone SDK versions.
  2. Drag the `AdMob/` folder into your Xcode project (containing `AdMobDelegateProtocol.h`, `AdMobView.h` and `libAdMob.a`)



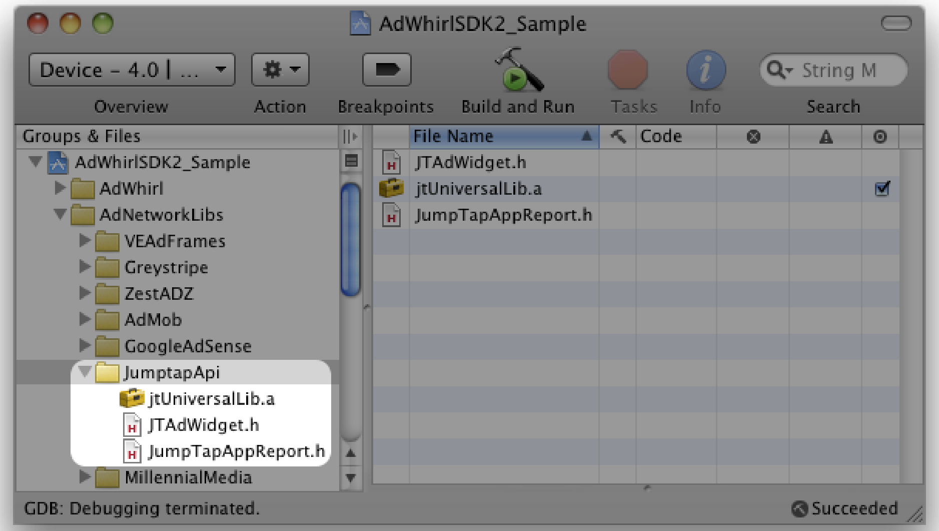
- **Google AdSense**
  1. Get your Google AdSense library from Google. There should be a universal binary for all iPhone SDK versions.
  2. Drag the `Library/` folder into your Xcode project (containing `GADAdSenseParameters.h`, `GADAdViewController.h`, `GADRequestError.h`, `libGoogleAds.a`)
  3. Make sure you implement the methods `googleAdSenseCompanyName`, `googleAdSenseAppName` and `googleAdSenseApplicationAppleID` in your AdWhirlDelegate.



- **Greystripe**
    i. Register with Greystripe and download the Greystripe SDK from http://developer.greystripe.com .
    ii. Register your application on the Applications tab and copy the Application Identifier to your AdWhirl Ad Network Settings.
    iii. Drag the `libGreystripeSDK.a`, `GSAdEngine.h`, `GSAdView.h`, and `GreystripeDelegate.h` files to your project.

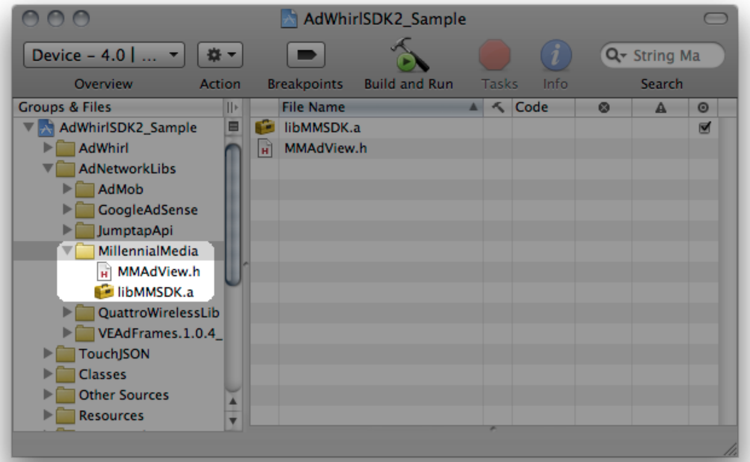


- **Jumptap**
    1. Download the JumpTap SDK from https://support.jumptap.com/index.php/IPhone_Library_Integration#Downloading_the_library .
    2. Drag the `JumptapApi/` folder into your Xcode project (containing the the files `jtUniversalLib.a`, `JTAdWidget.h` and `JumpTapAppReport.h`).
    3. If you've used Jumptap before AdWhirl iPhone SDK 2.5.0, read the following:
        a. Remove the old Jumptap Library Search Paths. Right click your primary target -> Get Info -> Build tab. Double click on the value for "Library Search Paths". Remove the older, JumptapApi that contains `$(PLATFORM_NAME)`
        b. Make sure you set SiteID and SpotID in adwhirl.com, in addition to PublisherID. SiteID and SpotID support in the AdWhirl server is added for AdWhirl iPhone SDK 2.5.0 or later.

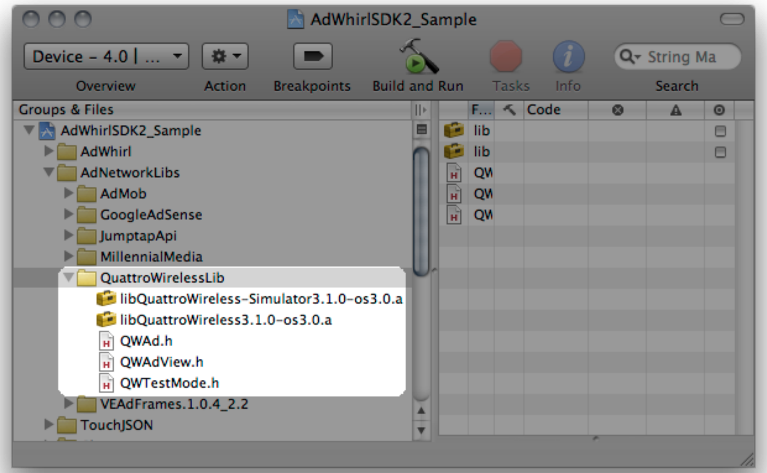


- **iAd**
  - Include the framework iAd.framework in your project. See step 3 above

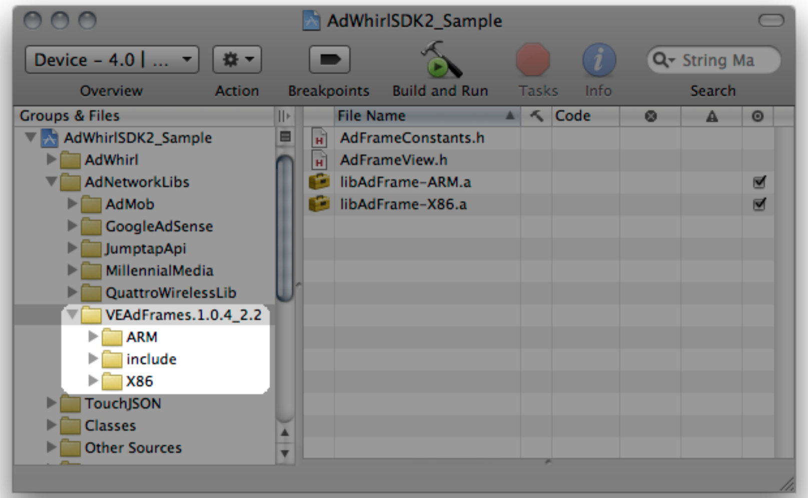
- **Millennial Media**
  i. Download the Millennial Media SDK from http://developer.millennialmedia.com/ . Millennial Media provides a universal binary for all iPhone SDK versions.
  ii. Drag the `Public/` folder into your Xcode project (containing the files `MMAdView.h` and `libMMSDK.a`).



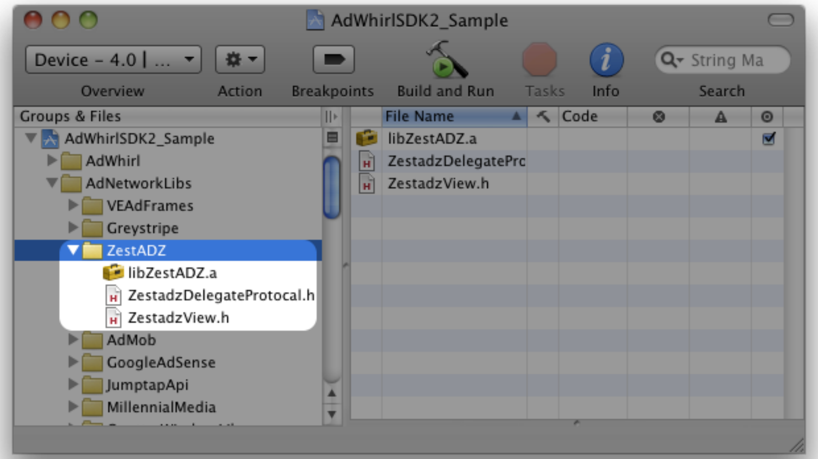
- **Quattro Wireless**
  i. Download the Quattro Wireless SDK from http://www.quattrowireless.com/ .
  ii. Drag the `QuattroWirelessLib/` folder into your Xcode project (containing the files `libQuattroWireless-Simulator3.1.0-os3.0.a`, `libQuattroWireless3.1.0-os3.0.a`, `QWTestMode.h`, `QWAd.h`, `QWAdView.h`. Exact names of .a files may be different depending on the version you downloaded).



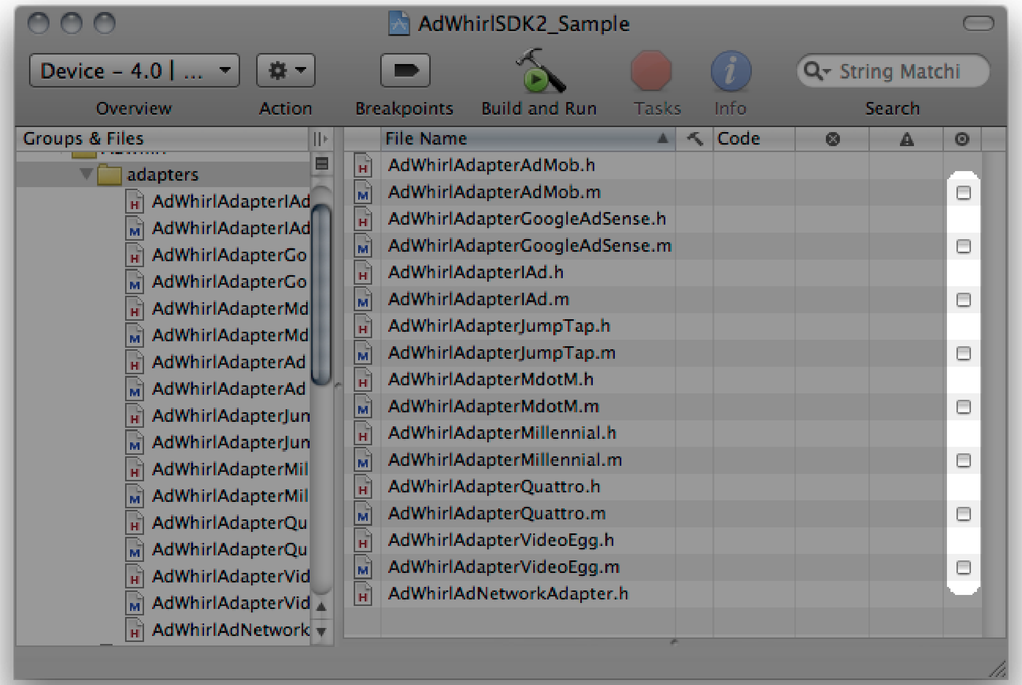
- **VideoEgg**
    i. Download the VideoEgg SDK from http://www.videoegg.com/ .
    ii. Drag the appropriate `VEAdFrames/` folder under the `lib/` folder into your project (containing the folders `ARM/`, `X86/`, and `include/`).
    iii. To avoid build errors, remove `AdFrameView.o` from under the `ARM/` and `X86/` folders. Rename `libAdFrame.a` in each directory to include the architecture in the file name, such as `libAdFrame-ARM.a` and `libAdFrame-X86.a` .



- **ZestADZ**
    iii. Download the ZestADZ SDK from http://zestadz.com/static/iphone_sdk . ZestADZ provides a universal binary for all iPhone SDK versions.
    iv. Drag the `ZestADZ/` folder into your Xcode project (containing `ZestadzDelegateProtocol.h`, `ZestadzView.h` and `libZestADZ.a`)



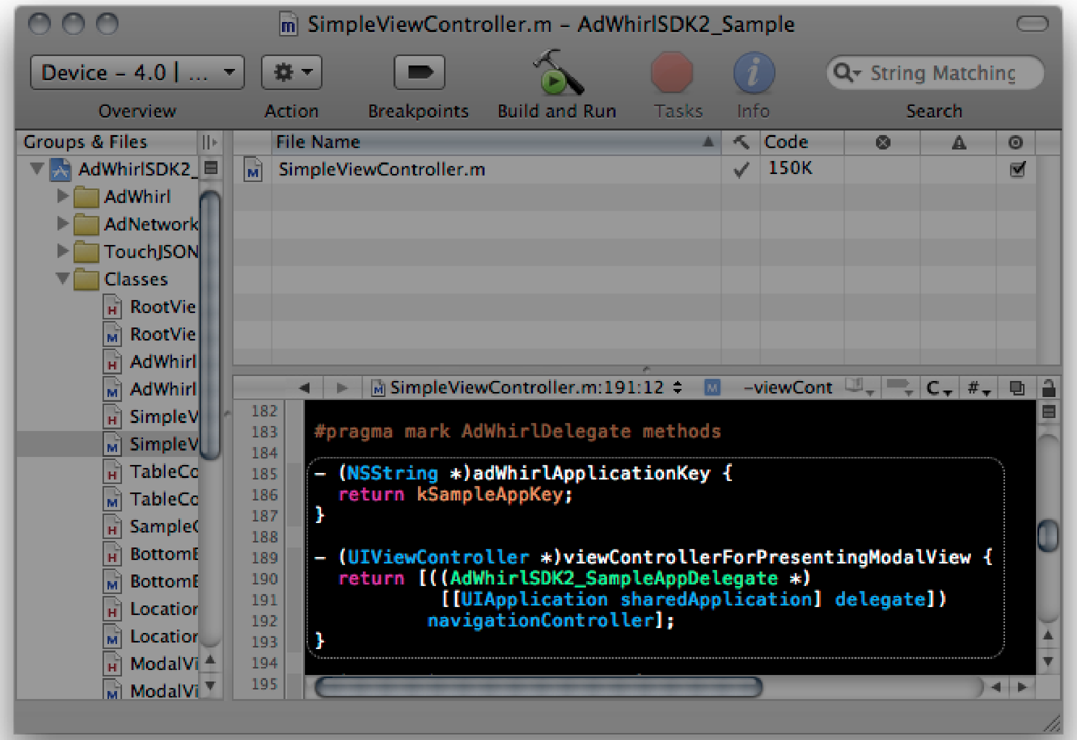
5. Remove the ad network adapters that you don't use. Under the `adapters/` group, you must uncheck the ad network adapters for ad networks that you did not integrate in step 4, or your project will not compile. In the extreme case, you can uncheck all `.m` files under `adapters/`. In which case your app can only run House Ads, Generic Notifications and Events.



AdWhirlAdapterAdMob.h
AdWhirlAdapterAdMob.m
AdWhirlAdapterGoogleAdSense.h
AdWhirlAdapterGoogleAdSense.m
AdWhirlAdapteriAd.h
AdWhirlAdapteriAd.m
AdWhirlAdapterJumpTap.h
AdWhirlAdapterJumpTap.m
AdWhirlAdapterMdotM.h
AdWhirlAdapterMdotM.m
AdWhirlAdapterMillennial.h
AdWhirlAdapterMillennial.m
AdWhirlAdapterQuattro.h
AdWhirlAdapterQuattro.m
AdWhirlAdapterVideoEgg.h
AdWhirlAdapterVideoEgg.m
AdWhirlAdNetworkAdapter.h

6. Implement the required AdWhirlDelegate methods in your code. You must implement the `adWhirlApplicationKey` method to return your AdWhirl application key. You must also implement the `viewControllerForPresentingModalView` method so click-through modal views are displayed properly.

See the comments in `AdWhirlDelegateProtocol.h` for optional methods to implement. You may need to implement extra delegate methods for some ad networks to operate properly.

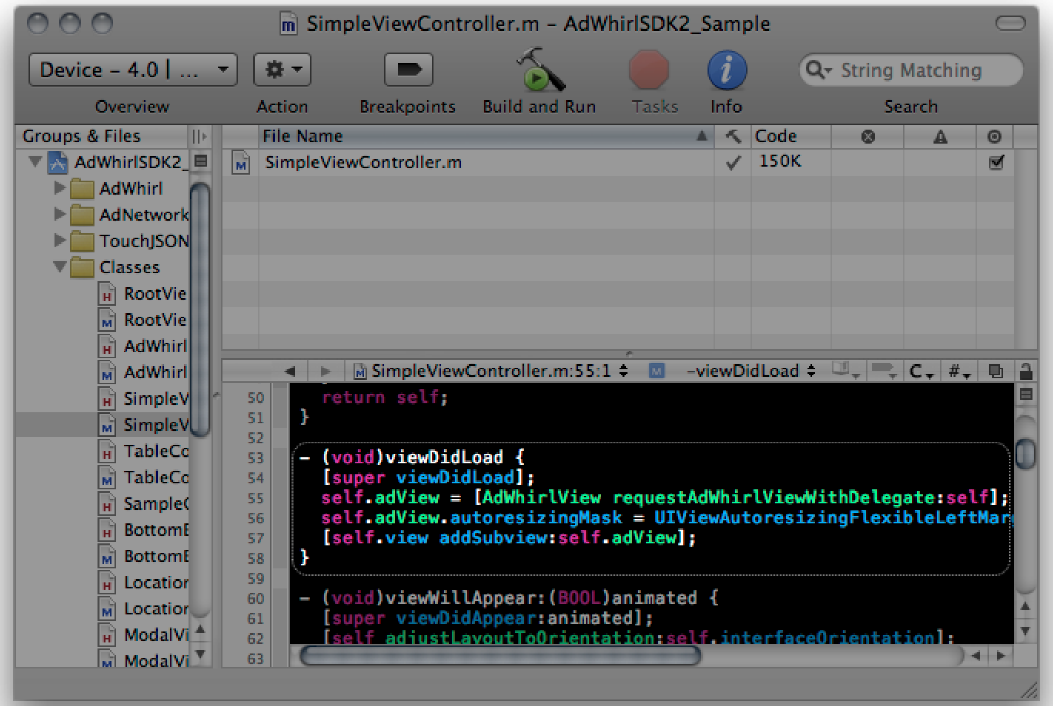


```
#pragma mark AdWhirlDelegate methods

- (NSString *)adWhirlApplicationKey {
  return kSampleAppKey;
}

- (UIViewController *)viewControllerForPresentingModalView {
  return [((AdWhirlSDK2_SampleAppDelegate *)
           [[UIApplication sharedApplication] delegate])
          navigationController];
}
```

If you have integrated with AdWhirl before the open source version (pre 2.0.0) and have implemented ARRollerDelegate, you will have to implement the extra required delegate method `viewControllerForPresentingModalView`. You can leave the rest of your implemented delegate methods intact as they have been defined to be the same as the equivalent AdWhirlDelegate methods. However we suggest that you replace your method names with the adWhirl-prefixed ones. Please see `ARRollerProtocol.h` under the `legacy/` directory for a list of method name mappings.

7. Request an AdWhirlView in your code and add it to your view hierarchy. For example:

```
AdWhirlView *awView = [AdWhirlView requestAdWhirlViewWithDelegate:self];
[self.view addSubview:awView];
```



```
    return self;
}

- (void)viewDidLoad {
    [super viewDidLoad];
    self.adView = [AdWhirlView requestAdWhirlViewWithDelegate:self];
    self.adView.autoresizingMask = UIViewAutoresizingFlexibleLeftMar
    [self.view addSubview:self.adView];
}

- (void)viewWillAppear:(BOOL)animated {
    [super viewDidAppear:animated];
    [self adjustLayoutToOrientation:self.interfaceOrientation];
```

Best Practices

Differing Ad Network Sizes

The ads from different ad networks may come in multiple sizes. For example, AdMob's banner is 320x48, and Millennial Media's banner is 320x53 . The default size of the AdWhirlView banner is 320x50 . To avoid unsightly borders or cutting off some part of the banner, you can adjust the sizes of AdWhirlView and surrounding views so the size of and spaces allocated for the AdWhirlView fits exactly the ad network's banner. You can obtain the actual size of the ad network by invoking the `actualAdSize` method of the AdWhirlView object. It is best to do so in the delegate method `adWhirlDidReceiveAd:` or `adWhirlDidAnimateToNewAdIn:` , like the following example:

```
- (void)adWhirlDidReceiveAd:(AdWhirlView *)adWhirlView {
  [UIView beginAnimations:@"AdResize" context:nil];
  [UIView setAnimationDuration:0.7];
  CGSize adSize = [adView actualAdSize];
  CGRect newFrame = adView.frame;
  newFrame.size.height = adSize.height; // fit the ad
  newFrame.size.width = adSize.width;
  newFrame.origin.x = (self.view.bounds.size.width - adSize.width)/2; // center
  adView.frame = newFrame;
  // ... adjust surrounding views here ...
  [UIView commitAnimations];
}
```

You can wrap the frame adjustments in an animation so the user experience is less jarring.

Handling Device Orientation

Some ad networks, notably iAd, may display a different sized banner depending on the device's orientation. If your app allows orientations other than portrait, you will have to:

- Invoke the `rotateToOrientation:` method of the AdWhirlView object whenever the orientation of the device changes, typically in your controller's `shouldAutorotateToInterfaceOrientation:` or `willRotateToInterfaceOrientation:duration:` method.
- Optionally implement the method `adWhirlCurrentOrientation` in your AdWhirlDelegate if your user interface's notion of orientation differs from `[UIDevice currentDevice].orientation.` This can happen when you rotate your views manually.
- Resize the adWhirlView and adjust your user interface after you receive an ad in `adWhirlDidReceiveAd:` and right after you invoke `rotateToOrientation:` when your view rotates.

```
- (void)adjustAdSize {
  // ... adjust the frame of the AdWhirlView and surrounding views ...
}

- (void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)newOrientation
                              duration:(NSTimeInterval)duration {
  [super willRotateToInterfaceOrientation:newOrientation
                                 duration:duration];
  [self.adWhirlView rotateToOrientation:newOrientation];
  [self adjustAdSize];
}

- (void)adWhirlDidReceiveAd:(AdWhirlView *)adWhirlView {
  [self adjustAdSize];
}
```

Build Warnings

To avoid build warnings (it is useful to have "Treat Warnings as Errors" turned on), you can uncheck static libraries from the target that are not the same architecture as those that you are building. This applies to cases where the ad network has separate binaries for the Simulator and the Device.