Mobclix SDK Quick Start Guide

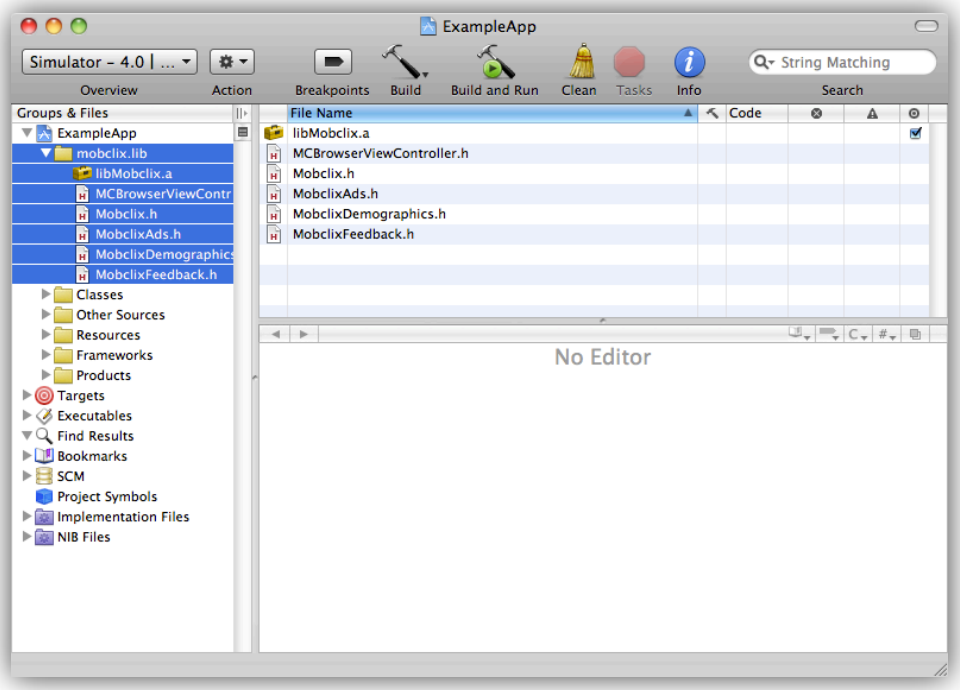
Version 4.2.2

Rev. 12/3/10

Contents

1. Setting up the Mobclix SDK

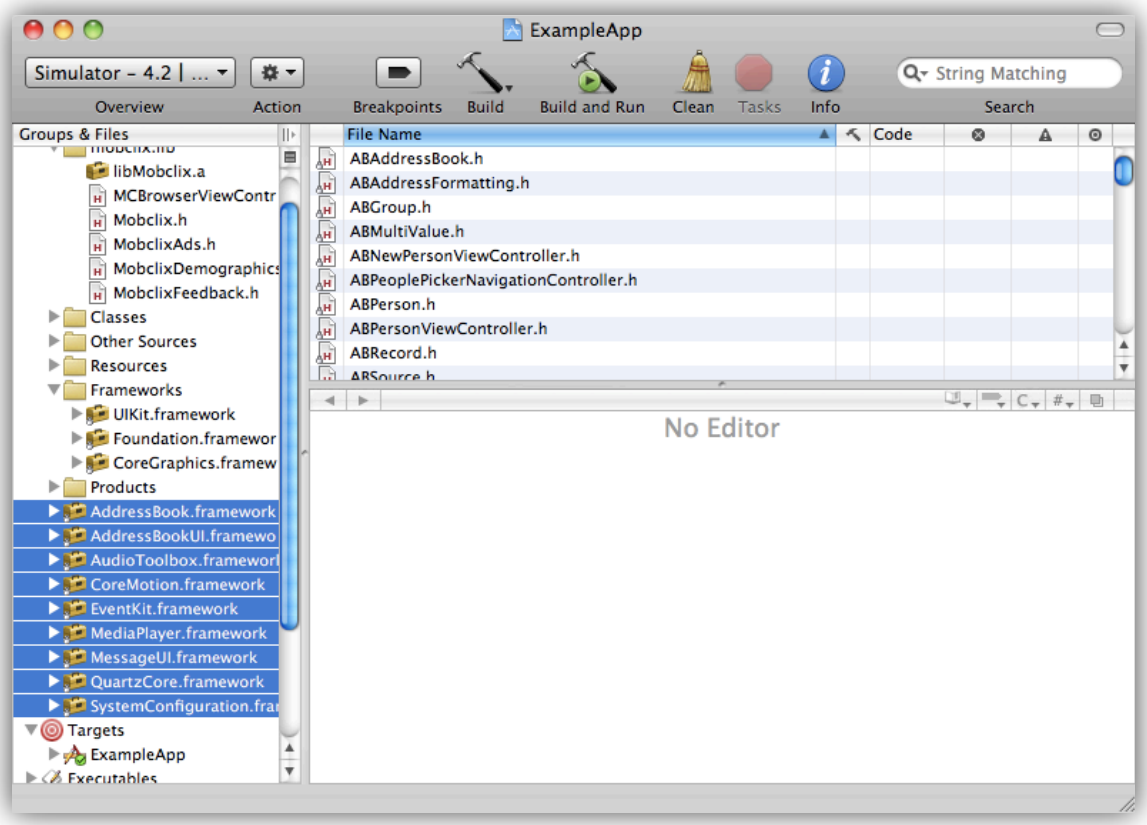
1.1 Adding the Mobclix SDK files to your Project



To get started, simply drag the "mobclix.lib" folder you downloaded into your project. The static library, **libMobclix.a**, should be added to your Application's target so it's correctly bundled.

Note: While **libMobclix.a** is roughly 8MB in size, the actual added size to your application will only be about 300KB per architecture.

1.2 Linking to the Dependency Frameworks



There are a few framework dependencies that the Mobclix library requires. These will not add any additional size to your application, since they're already on the device, but you'll need to link against them.

1. Right click on your application under "Targets" and select "Get Info"
2. Under the "General" tab, select the plus icon in the bottom left corner
3. Select the following frameworks, then press "Add"

a.	AddressBook	d.	AVFoundation	g.	MediaPlayer	j.	SystemConfiguration
b.	AddressBookUI	e.	CoreMotion	h.	MessageUI		
c.	AudioToolbox	f.	EventKit	i.	QuartzCore		

- *The standard UIKit, Foundation, and CoreGraphics frameworks are also required. If you've removed any them from your project, you'll need to re-add them.*
- *When targeting 3.x devices, the AVFoundation, CoreMotion and EventKit frameworks should be weak linked. To do this, switch the "required" option next to each framework to "weak".*

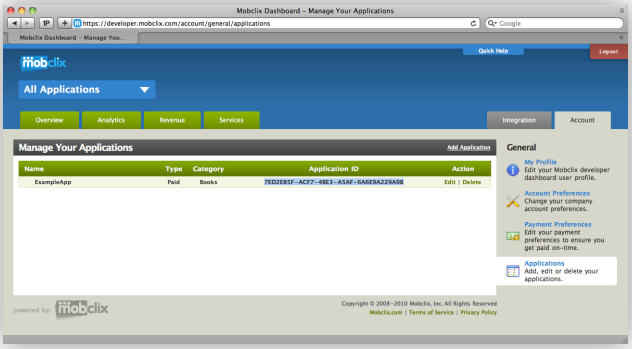
2. Integrating with Your Code

2.1 Setting up your App Delegate

You will need to "start" Mobclix in your App Delegate. In most cases, this file is called ***AppName*AppDelegate.m**. At the top of this file you should add `#import "Mobclix.h"`. The only other line you'll need to add to this file is `[Mobclix startWithApplicationId:]`. Depending on your application, this should go at the bottom of the `applicationDidFinishLaunching:` method, or towards the bottom of the `application:didFinishLaunchingWithOptions:` method. You can see an example of this below:

```
#import "ExampleAppAppDelegate.h"
#import "RootViewController.h"

// Make sure to import the Mobclix header file
#import "Mobclix.h"

@implementation ExampleAppAppDelegate

- (void)applicationDidFinishLaunching:(UIApplication*)application {
    // Add this line to start up Mobclix
    [Mobclix startWithApplicationId:@"insert-your-application-id"];

    // Add this line to start up Mobclix
    [window addSubview:[navigationController view]];
    [window makeKeyAndVisible];
}
```

You should set the value of this ID to your Mobclix Application ID, found in the Mobclix Dashboard by clicking the "Account" tab and then clicking into the "Applications" section. If you haven't added your application to Mobclix, now would be a great time to do so. You can follow the steps found here, and you'll receive your Application ID when you're finished adding it.



Next you'll need to integrate the ads into your application. You can do this with either Interface Builder, or by initializing the views yourself. We've outlined both methods in this document.

2.2 Integrating with Interface Builder

1. Setup your Interface file (*YourViewController*.h) with "adView" as an outlet, as shown below:

```
#import <UIKit/UIKit.h>
#import "MobclixAds.h"

@interface RootViewController : UIViewController {
@private
     MobclixAdView* adView;
}

@property(nonatomic,retain) IBOutlet MobclixAdView* adView;
@end
```

2. Next, you should synthesize the adView property that was setup above, in your Implementation file (*YourViewController*.m).

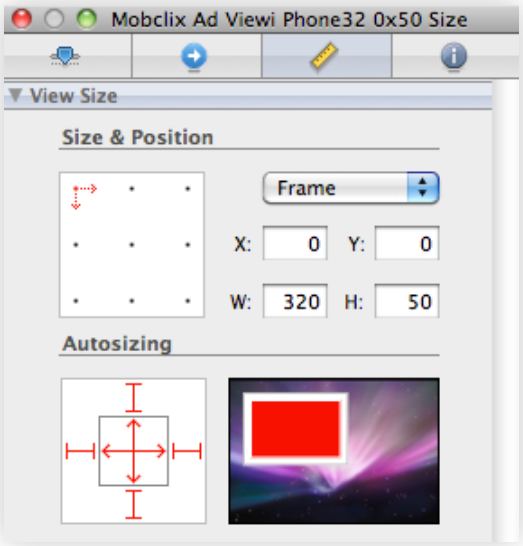
```
#import "RootViewController.h"


@implementation RootViewController
@synthesize adView;
```

3. Setup your Interface Builder file (usually *YourViewController*.xib) with an ad view.

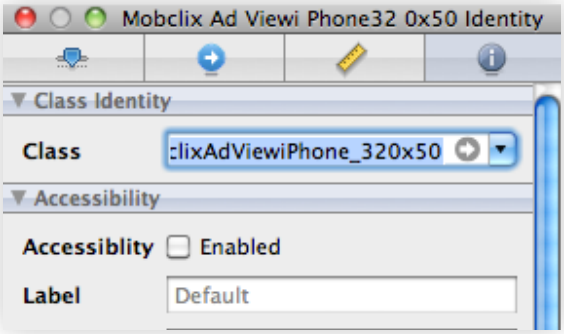
    3.1. Double click "View" to bring your main view into focus.

    3.2. Drag a new View from your library window onto the main view.

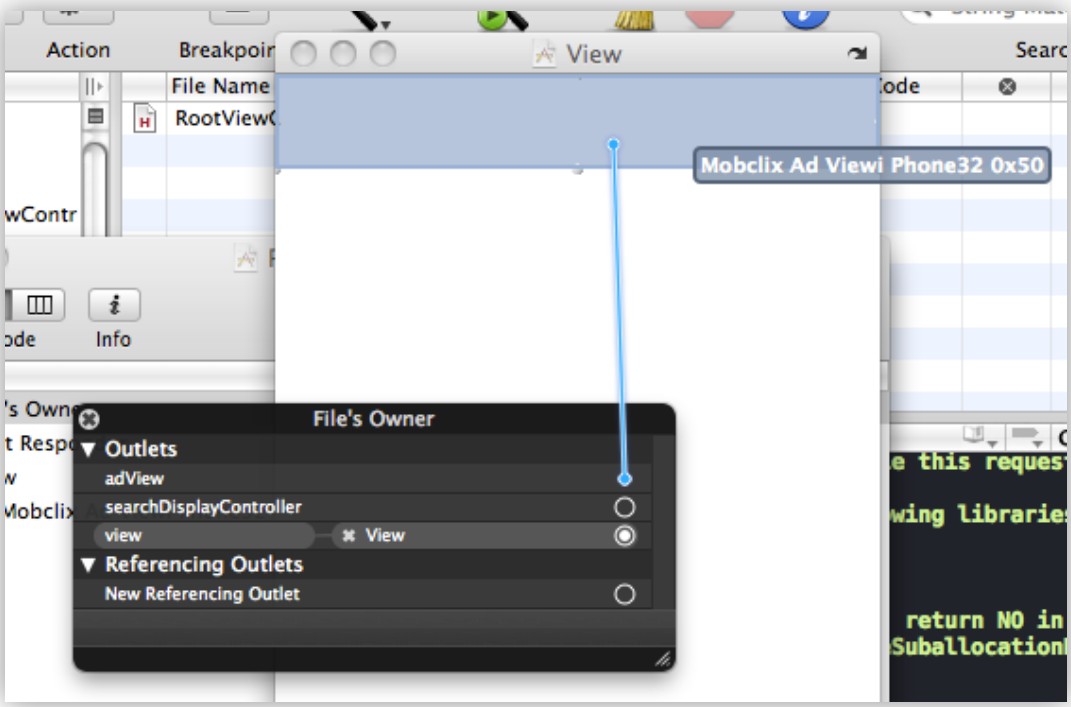
    3.3. Make sure the new created view is selected, and open the "Inspector" window. Select the Ruler tab and change the width to **320** and the height to **50**. You should then drag the view to where ever you'd like for it to appear on the screen. You should also update the autoresizing masks and disable flexible width, and flexible height. You can do this by clicking each solid line, so the only solid line left is the top one. When you're done your window should look like the screenshot shown to the right.



3.4. In the Inspect window, select the tab with the "i" icon. Change the class to "MobclixAdViewiPhone_320x50" as shown below:



3.5. In the Document window, right click your view controller, and drag the circle to the right of "adView" to the ad view you created in your main view, as shown below:



3.6. Next you'll need to add the "-ObjC" linker flag to your target.

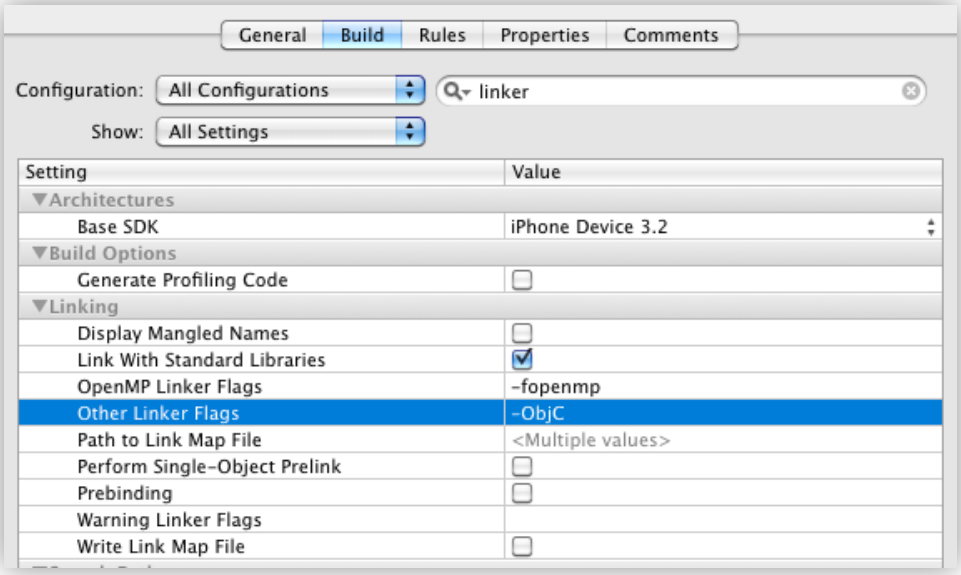
3.6.1. To do this, expand the "Targets" section of your project.

3.6.2. Right click on your project's target, and select "Get Info"

3.6.3. Select the "Build" tab at the top and scroll down to "Other Linker Flags" in the "Linking" section

3.6.4. Double click "Other Linker Flags" and add "-ObjC" (no quotes).

3.6.5. **Note:** Make sure "All Configurations" is selected for the "Configuration" drop down, otherwise you'll only set it for one configuration.



3.7. You can now save and close your Interface Builder file. Jump to the Section 3 called **"Requesting and Managing Ads"**.

2.3 Integrating without Interface Builder

For more advanced implementations of ads, you'll want to create and place the views in your code. We're going to show you a basic implementation using viewDidLoad.

Integrating this way is actually quite simple and only requires three steps:

1. Setup your Interface file (*YourViewController*.h) with "adView", as shown below:

```
#import <UIKit/UIKit.h>
#import "MobclixAds.h"

@interface RootViewController : UIViewController {
@private
     MobclixAdView* adView;
}

@property(nonatomic,retain) MobclixAdView* adView;
@end
```

2. Next, you should synthesize the adView property that was setup above, in your Implementation file (*YourViewController*.m).

```
#import "RootViewController.h"


@implementation RootViewController
@synthesize adView;
```

3. Now all you need to do is create the view, and add it to your view hierarchy, as shown below.

```
- (void)viewDidLoad {
    [super viewDidLoad];

     self.adView = [[[MobclixAdViewiPhone_320x50 alloc]
initWithFrame:CGRectMake(0.0f, 0.0f, 320.0f, 50.0f)]
autorelease];
     [self.view addSubview:self.adView];
}
```

3. Requesting and Managing Ads

It's very important to request ads properly. If you don't follow the instructions below, you will have lower click through rates, which will lead to lower CPM's and lower revenue for you.

3.1 Automatically request new ads

In your view controller's `viewDidAppear:` method, you should start the ad refresh timer, by calling `resumeAdAutoRefresh`. If this is the first time you call this method, it will automatically call `getAd`. Subsequent calls to this method, will reschedule the auto refresh timers.

```
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];
    [self.adView resumeAdAutoRefresh];
}
```

3.2 Pause ad requests when your view disappears

When your view goes off screen, you should pause the ad refresh timers. You do this by calling `pauseAdAutoRefresh`. If you don't pause ads when your view goes off screen, you will not only lower your CPMs and CTRs, you will impact your applications overall performance.

```
- (void)viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];
    [self.adView pauseAdAutoRefresh];
}
```

3.3 Properly canceling and releasing your ad view

The number one cause of crashes when using ads is developers not properly canceling and releasing the ad view when their view controller unloads or terminates.

Upon your view unloading or your view controller deallocates, three calls should be made:

1. Call `cancelAd` to stop any current ads from loading, and cancel the ad refresh timers.
2. Set your ad view's delegate to nil. This is important, as your ad view might not deallocate immediately and could send a delegate message to a released object if you don't set it to nil.
3. Release your ad view.

A correctly implemented example of this can be found below:

```
@implementation RootViewController

/* ... */

- (void)viewDidUnload {
     [self.adView cancelAd];
     self.adView.delegate = nil;
     self.adView = nil;
}


- (void)dealloc {
     [self.adView cancelAd];
     self.adView.delegate = nil;
     self.adView = nil;

    [super dealloc];
}


@end
```

5. Additional Support

If you have any additional questions, there are a few more resources available to you:

- The Mobclix Example Application, found here, shows the Mobclix SDK fully integrated and running.
- You can view the Mobclix SDK Documentation online for a complete overview of the SDK and all classes and methods available.
- The header files provided with the Mobclix SDK contain documentation above each method.
- The Mobclix iPhone & iPad SDK Google Group is a great way to interact with other developers using the Mobclix SDK as well as asking a question directly to the engineers at Mobclix.
- If you have any other questions, send us an email at support@mobclix.com and we'll be happy to help!