# iOS mOcean SDK

# Developer Getting Started Guide

For iOS SDK Version 4.0

136 Baxter St, New York, NY 10013

# Table of Contents

136 Baxter St, New York, NY 10013

# What's New in 4.0

Makes use of NSURLProtocol for cleaner MRAID JavaScript bridge injection.

More control over MRAID resize superview (if needed).

Stability and bug fixes.

Native Ads Support - Native ads allow a publisher to display ads using the same structure and formatting of their site/app without hindering the user's experience. Such ads grasp the attention of the visitors more than the traditional ad formats. Mocean and PubMatic now allow publishers to serve native ads on their inventory.

# Implementation Changes

SDK 4.0 is not compatible with the SDK 2.0.  Prior to installing SDK 4.0, remove the old SDK references.

# System Requirements

Intel based Mac

Xcode 5.1 or higher

iOS 6.0 or higher*

*iOS 5.0 and higher can be attempted with previous versions of Xcode and iOS SDK.

# Prerequisites

This guide does not cover iOS development techniques or instructions for using Xcode to develop applications for iOS.  iOS developer documentation is available from Apple at https://developer.apple.com/devcenter/ios/.

More thorough, complex examples and additional use cases in the sample application distributed with the SDK. Both the sample app and the SDK itself are available in source code form from our Google Code project site at http://code.google.com/p/mocean-sdk-ios/.

Additional documentation, information, and other supported platforms on the developer wiki at: http://developer.moceanmobile.com/Main_Page.

# Feature List

### Rich media

Support for MRAID 2 (includes MRAID 1).

### HTML/ JS ads

SDK supports displaying web ads using UIWebView component.

### Image/Text ads

SDK supports displaying image and text ads with non-UIWebView native components.

### Location auto detect

SDK can automatically detect user location.

### User-Agent auto detect

SDK automatically detects device User-Agent.

### Internal browser

SDK contains built-in browser for displaying ads in application.

### Ad visibility tracking

SDK automatically detects ads visibility for controlling updates.

### Logging

SDK supports logging through delegate callbacks and NSLog.

### Native ads support

# Installing the Ad SDK

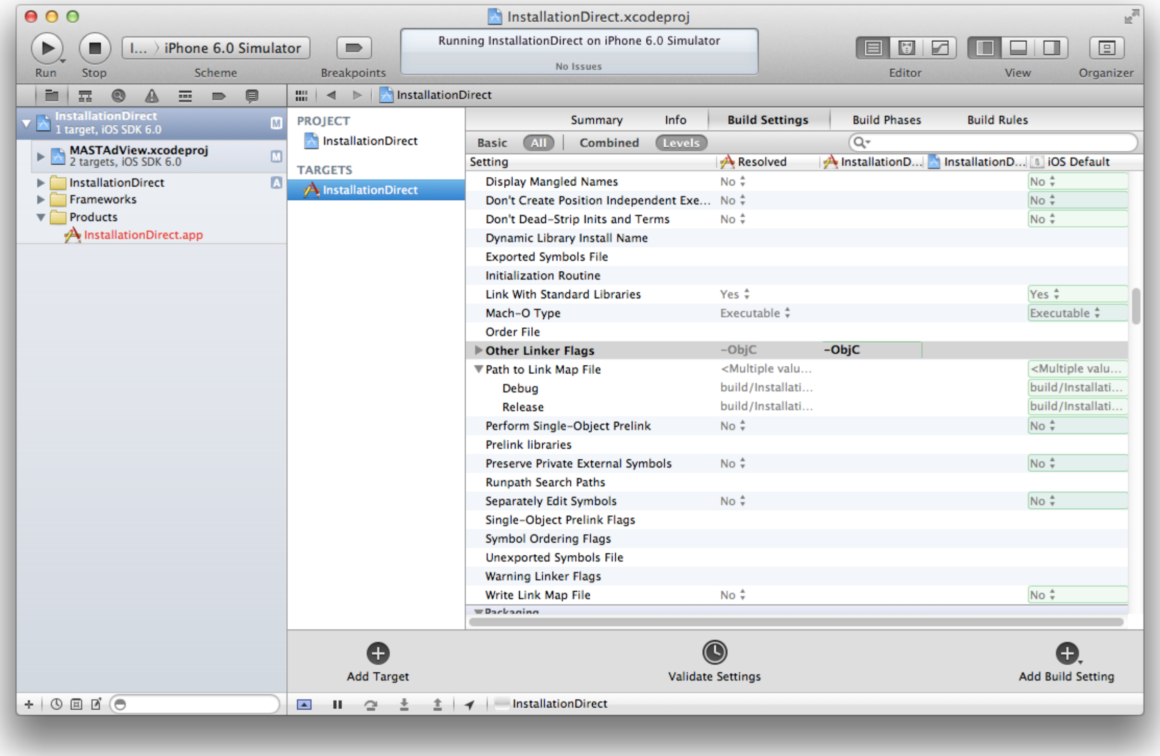
## SDK Integration Methods

The SDK can be integrated into applications in various ways.  The simplest is to reference the SDK Xcode project file in the application and its dependencies.  The SDK project can also build and package a framework bundle that can be prebuilt and referenced like other iOS bundles.

Developers can use either approach or other custom methods to include the SDK in an application.

## Objective-C Linking

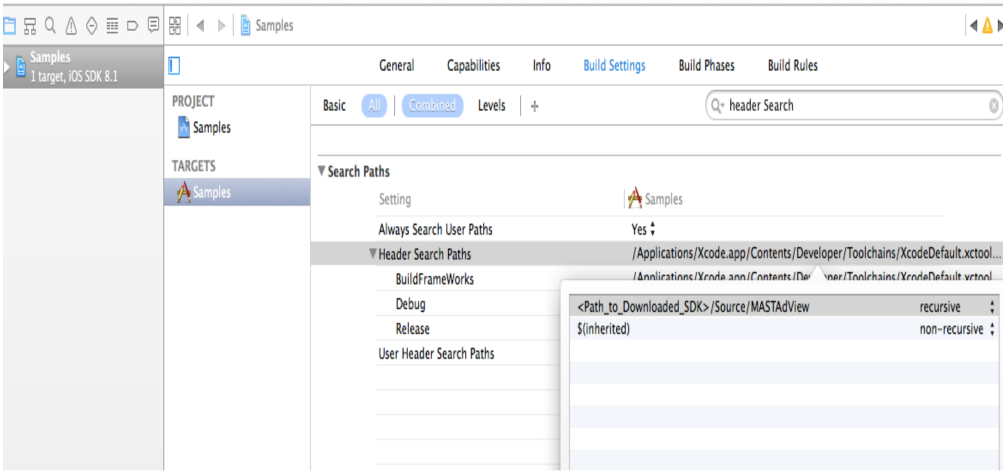
Regardless of the installation method the project must configure the linker to properly link Objective-C libraries. This is done by adding the **-ObjC** flag to the application's target **Other Linker Flags** setting.



| Setting | Resolved | InstallationD... | InstallationD... | iOS Default |
|---|---|---|---|---|
| Display Mangled Names | No | | | No |
| Don't Create Position Independent Exe... | No | | | No |
| Don't Dead-Strip Inits and Terms | No | | | No |
| Dynamic Library Install Name | | | | |
| Exported Symbols File | | | | |
| Initialization Routine | | | | |
| Link With Standard Libraries | Yes | | | Yes |
| Mach-O Type | Executable | | | Executable |
| Order File | | | | |
| **Other Linker Flags** | -ObjC | -ObjC | | |
| Path to Link Map File | <Multiple valu... | | | <Multiple valu... |
| Debug | build/Installati... | | | build/Installati... |
| Release | build/Installati... | | | build/Installati... |
| Perform Single-Object Prelink | No | | | No |
| Prelink libraries | | | | |
| Preserve Private External Symbols | No | | | No |
| Runpath Search Paths | | | | |
| Separately Edit Symbols | No | | | No |
| Single-Object Prelink Flags | | | | |
| Symbol Ordering Flags | | | | |
| Unexported Symbols File | | | | |
| Warning Linker Flags | | | | |
| Write Link Map File | No | | | No |

## Adding Header Search Paths

- Click the Project Name in Navigator and then select target.
- Go to the Build Settings Tabs and search "Header Search Paths" in the search box to view the "Header Search Paths" section
- Click "Header Search Paths" and add the header path entry, such as <Path_to_Downloaded_SDK>/Source/MASTAdView to the header search path section.
- Toggle the field in front of the path to recursive, to enable recursive search.



header Search

Search Paths

| Setting | Samples |
|---|---|
| Always Search User Paths | Yes |
| Header Search Paths | /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctool... |
| BuildFrameWorks | /Applications/Xcode.app/Contents/De per/Toolchains/XcodeDefault.xctool |
| Debug | |
| Release | |
| User Header Search Paths | |

| | |
|---|---|
| <Path_to_Downloaded_SDK>/Source/MASTAdView | recursive |
| $(inherited) | non-recursive |

## Required Frameworks

The SDK leverages various iOS frameworks.  Adding the SDK to an application target will require these frameworks to be referenced by the target.  The following frameworks are required:

- Foundation
- UIKit
- EventKit
- EventKitUI
- MessageUI
- CoreLocation
- CoreGraphics
- ImageIO
- CoreTelephony

## Option 1: Referencing MASTAdView.xcodeproj

This installation option references the SDK project directly from source. This is the easiest option since it doesn't require the SDK to be pre-built and SDK releases can be updated by updating the source. Each time the application target is built Xcode will build the SDK as needed. The following steps are used to reference the SDK project in an application target.

1. With the desired destination selected in the Project Navigator select "Add files to <project>…" from either the Project Navigator context menu or the Xcode File menu. Browse and select the MASTAdView.xcodeproj. This will add the SDK project reference to the application target.

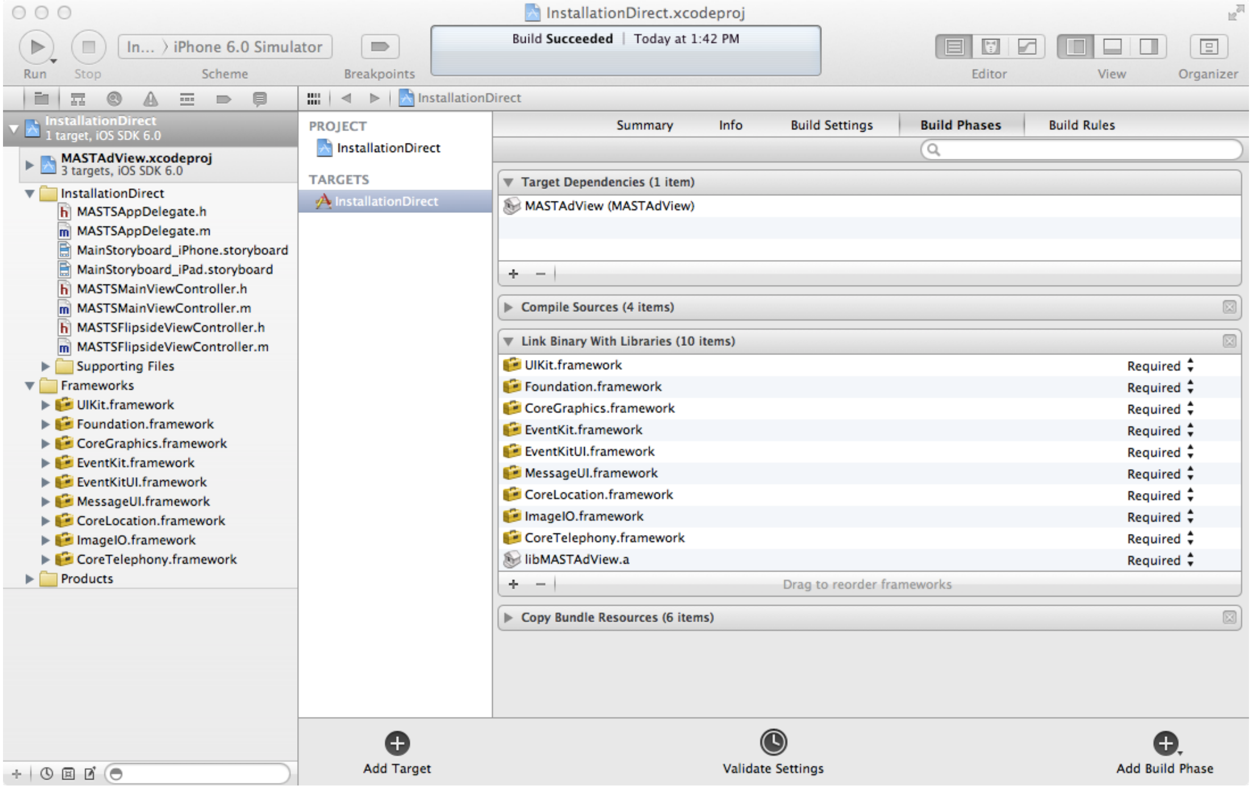
   NOTE: Xcode can only have one reference to any given project at a time. Be sure the MASTAdView.xcodeproj project isn't opened or referenced in another project in Xcode at the same time. For developers that will use the SDK in multiple projects it may be easier to have each project reference its own copy of the SDK.

2. Add the SDK to the Build Phase tab's Target Dependency list. This will ensure Xcode properly builds the SDK prior to building the application.

3. Add the SDK library to the Build Phase tab's Link Binary With Libraries list. After both the SDK and the application target are built this will cause Xcode to statically link the SDK library to the application executable.

4. Include the MASTAdView header in the source where it will be used with the following directive:
   **#import "MASTAdView.h"**

The following screen shot shows the InstallationDirect sample's configuration. The project is shown referenced in the Project Navigator on the left and the additions to the build phases are shown on the right.



Build Succeeded | Today at 1:42 PM

- InstallationDirect
  1 target, iOS SDK 6.0
  - MASTAdView.xcodeproj
    3 targets, iOS SDK 6.0
  - InstallationDirect
    - MASTSAppDelegate.h
    - MASTSAppDelegate.m
    - MainStoryboard_iPhone.storyboard
    - MainStoryboard_iPad.storyboard
    - MASTSMainViewController.h
    - MASTSMainViewController.m
    - MASTSFlipsideViewController.h
    - MASTSFlipsideViewController.m
    - Supporting Files
  - Frameworks
    - UIKit.framework
    - Foundation.framework
    - CoreGraphics.framework
    - EventKit.framework
    - EventKitUI.framework
    - MessageUI.framework
    - CoreLocation.framework
    - ImageIO.framework
    - CoreTelephony.framework
  - Products

PROJECT: InstallationDirect

TARGETS: InstallationDirect

Target Dependencies (1 item)

MASTAdView (MASTAdView)

Compile Sources (4 items)

Link Binary With Libraries (10 items)

| | |
|---|---|
| UIKit.framework | Required |
| Foundation.framework | Required |
| CoreGraphics.framework | Required |
| EventKit.framework | Required |
| EventKitUI.framework | Required |
| MessageUI.framework | Required |
| CoreLocation.framework | Required |
| ImageIO.framework | Required |
| CoreTelephony.framework | Required |
| libMASTAdView.a | Required |

Copy Bundle Resources (6 items)

## Option 2: Referencing MASTAdView.framework

This installation option references a pre-built SDK framework package. This allows developers to build and reuse the framework package without having to reference the SDK source. It also removes any limitation in Xcode with multiple projects open that include references to the same embedded project files (See note in Option 1 above). The following steps are used to reference the framework package.

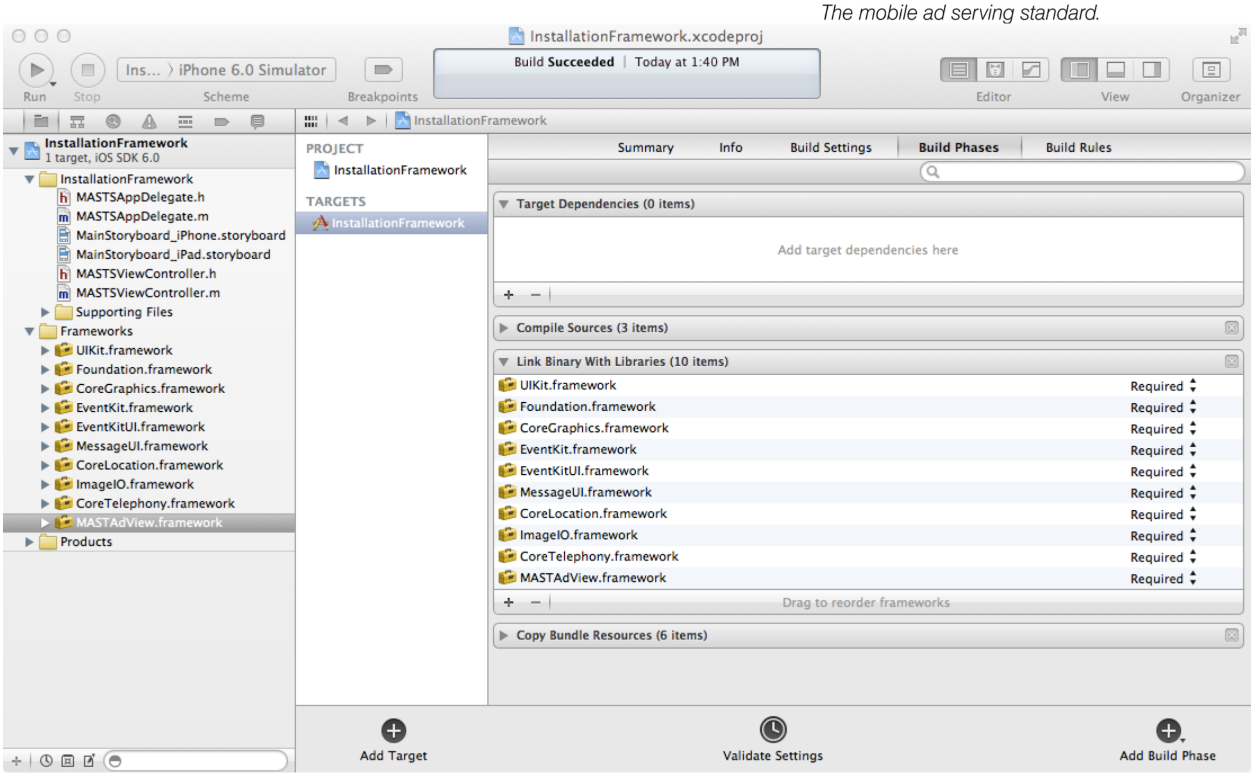
1. Open the SDK source and build the Framework target for an iOS Device destination. This will produce the MASTAdView.framework package (folder) in the Sources/MASTAdView/Products folder of the SDK source package.

2. Copy the MASTAdView.framework package to the desired destination (generally somewhere in the application source tree).

3. Add the MASTAdView.framework package to the Build Phase tab's Link Binary With Libraries list.

4. Include the MASTAdView header in the source where it will be used with the following directive:
   **#import <MASTAdView/MASTAdView.h>**

The following screen shot shows the InstallationFramework sample's configuration. Note the only change to the project is the addition of the framework. Unlike iOS frameworks the SDK framework will be statically linked and is not "installed" on the device.



- InstallationFramework
  1 target, iOS SDK 6.0
  - InstallationFramework
    - MASTSAppDelegate.h
    - MASTSAppDelegate.m
    - MainStoryboard_iPhone.storyboard
    - MainStoryboard_iPad.storyboard
    - MASTSViewController.h
    - MASTSViewController.m
    - Supporting Files
  - Frameworks
    - UIKit.framework
    - Foundation.framework
    - CoreGraphics.framework
    - EventKit.framework
    - EventKitUI.framework
    - MessageUI.framework
    - CoreLocation.framework
    - ImageIO.framework
    - CoreTelephony.framework
    - MASTAdView.framework
  - Products

PROJECT
InstallationFramework

TARGETS
InstallationFramework

Target Dependencies (0 items)

Compile Sources (3 items)

Link Binary With Libraries (10 items)

| | |
|---|---|
| UIKit.framework | Required |
| Foundation.framework | Required |
| CoreGraphics.framework | Required |
| EventKit.framework | Required |
| EventKitUI.framework | Required |
| MessageUI.framework | Required |
| CoreLocation.framework | Required |
| ImageIO.framework | Required |
| CoreTelephony.framework | Required |
| MASTAdView.framework | Required |

Copy Bundle Resources (6 items)

## Referencing Adapters' Projects (Mediation framework)

Incase you want to use Mediation framework, using one of the options mentioned above you can link adapter project to your application and use 3rd party SDK.

**Facebook Adapter**:

- Download FAN SDK from their official site.
- On downloading you will find "FBAudienceNetwork.framework" in it.
- Link this framework to the adapter project and your application

**MoPub Adapter**:

- Download MoPub Adapter from their official site
- On downloading you will find a folder "MoPubSDK" in it.
- Copy/Link this folder to your application.

**Note** - If you are using non-arc application and third party SDK is of kind ARC then set compiler flag for respective files with -fobjc-arc from build phases

# MASTAdView Use Case

1. A simple banner that is used for the embedding of the existing view and occupies a small area in it.

2. Interstitial is used for the full screen banner display. It's displayed only once and after closing it becomes inactive.

    1.1. Full screen banner is shown as a pop up on top of the main screen of the window and blocks all users' actions with the application.

    1.2. Full screen banner is shown as a pop up on top of the main screen of the window and doesn't block all user actions with the application. As an example, the user can switch tabs.

    1.3. If several displays are switched on, the banner will be displayed on the main screen only.

# User Interface / Layout (Design)

The first step is deciding where you want to incorporate ads in your application.

The simplest approach is to integrate a small horizontal banner ad into the user-interface (UI). A typical form factor is a 50-pixel tall (which would equal 100 pixels of the display for Retina devices), full width rectangle which does not crowd the existing UI elements or break the appearance and flow. As an example, consider the following Flickr image viewer before and after a banner ad has been inserted. We will show the steps to setup and display this ad below.

# Simple Ad Integration

Note: Be sure to follow the steps above for installation and import the MASTAdView.h header based on the installation type.

Implementation can vary depending on existing code style and layout as well as ARC (automatic reference counting) and non-ARC based projects. Because the MASTAdView can have a delegate it’s good practice to retain a reference and release it in dealloc and not rely on a superview to maintain its lifetime. Throughout this guide ARC and non-ARC semantics may be used (ex: strong vs. retain).

Basic code to extend an existing controller:

```
// In the interface directive add a property to hold the ad view
// reference(either the header or the private category in the implementation)

@interface MyViewController ()
@property (nonatomic, strong) MASTAdView* adView;
@end

// In the implementation directive, synthesize the property.
@implementation MyViewController
@synthesize adView;

- (void)dealloc
{
  // Whether or not the delegate is set, it’s good practice to clear it
  // to future proof the code.  Also, cancel any update timers and reset
  // adView state.
  [self.adView setDelegate:nil];
  [self.adView cancel];
  self.adView = nil;
}

- (void)viewDidLoad
{
  if (self.adView == nil)
  {
    // Create a frame for the ad view.  This is usually done to calculate
    // a position from some other views; here it’s just in a fixed banner
    // at the top of the controller’s view.
    CGRect frame = CGRectMake(0, 0, 320, 50);

    // Create the MASTAdView adView instance.
    self.adView = [[MASTAdView alloc] initWithFrame:frame];
    // Setup normal view properties like autorotation masks, background
    // colors, etc..

    // NOTE: Developers will need to get their own site and zone
    // configuration and should never use these test values in production
    // releases.
    self.adview.site = 19829;
    self.adView.zone = 88260;
  }

  // Add the adView to the view controller view.
  [self.view addSubview:self.adView];

  // Update the adView; this will request and display ad content.
  [self.adView update];
}

// If the application supports rotation and auto resizing the following
// can be used to refresh the adView after the view resizes due to a
// rotation event.
-
(void)didRotateFromInterfaceOrientation:(UIInterfaceOrientation)fromOrientatio
n toInterfaceOrientation:(UIInterfaceOrientation)toOrientation
{
  // If updating with a refresh time call that method here again instead.
  [self.adView update];
}

@end
```

MASTAdView reference creation, configuration, cleanup, rotation, presentation and rotation handling

This example shows a few properties of the ad view being set including:

- **Publisher site**: this is setup through the Mocean Mobile UI when you setup ad feeds to display content in your application. Typically a “site” will be used to identify one of your applications and distinguish it from another of your applications. The site is required in order to request an ad.

- **Ad zone**: this is used to identify one specific ad placement in your application. In this example we have created one placement so far, the banner ad to be displayed at the top of the screen. If we choose to display ads in another part of this application, a different placement will be used for that location. Zones are created through the Mocean Mobile UI and target content to ad placements in your application. A given zone falls under one site. The zone is required in order to request an ad.

- **Ad update interval**: This configured the time period (in seconds) after which the ad view will retrieve a new ad from the back-end.

See Also:

For more code samples examine the Samples application.

# Interstitial Ad Integration

Interstitial ads work much like inline/banner ads except that they are directly displayed and do not need to be added to a superview.  Normal interstitial ads are full screen and modal in appearance.  The MASTAdView can be initialized to display interstitial content directly to the screen without having to manage a separate modal view controller.  Note however that an interstitial instance can’t be used as a banner and vice versa.

Basic code to add an interstitial MASTAdView reference:

```
// In the interface directive add a property to hold the ad view
// reference(either the header or the private category in the implementation)

@interface MyViewController ()
@property (nonatomic, strong) MASTAdView* adViewInterstitial;
@end

// In the implementation directive, synthesize the property.
@implementation MyViewController
@synthesize adViewInterstitial;

- (void)dealloc
{
  // Whether or not the delegate is set, it’s good practice to clear it
  // to future proof the code.  Also, cancel any update timers and reset
  // adViewInterstitial state.
  [self.adViewInterstitial setDelegate:nil];
  [self.adViewInterstitial cancel];
  self.adViewInterstitial = nil;
}

- (void)viewDidAppear
{
  if (self.adViewInterstitial == nil)
  {
    // Create the MASTAdView adViewInterstitial instance.
    self.adViewInterstitial = [[MASTAdView alloc] initInterstitial];

    // NOTE: Developers will need to get their own site and zone
    // configuration and should never use these test values in production
    // releases.  There will usually be different site/zone combinations
    // for different ad placements and types.
    self.adViewInterstitial.site = 19829;
    self.adViewInterstitial.zone = 88260;
  }

  // Every time the view appears, display the interstitial.
  [self.adViewInterstitial update];
  [self.adViewInterstitial showInterstitial];

}

@end
```

MASTAdView interstitial reference creation, configuration, cleanup and presentation

Note that interstitial ad views still require update to be called and can be customized like inline/banner ads.

Since ads can be highly customized it is also to use the ad in inline mode to display partial screen interstitials. In this manner the ad view would be created as normal and the interstitial methods would not be used.

See Also:

For more code samples examine the Samples application.

To automatically close the interstitial after a specified amount of time use the showInterstitialWithDuration: method.

# Native Ads Integration

You can use the same SDK/framework to integrate native ads in the application.

You need to initialize **MastNativeAd** to use Native ads. The sample code is given below.

```
// In the interface directive add a property to hold the ad view
// reference(either the header or the private category in the
implementation)

@interface MyViewController ()
@property (nonatomic, strong) MASTNativeAd* nativeAd;
@end

@implementation MyViewController

// In the implementation directive, synthesize the property.
- (void)viewDidLoad

{

    [super viewDidLoad];

    self.nativeAd = [[MASTNativeAd alloc] init];

    // ZONE_ID received from Mocean Portal

    self.natitiveAd.zone = ZONE_ID;

    self.nativeAd.delegate = self;

    self.nativeAd.test = YES;

    [self.nativeAd setLocationDetectionEnabled:YES];

    self.nativeAd.nativeAdIconSize = self.iconImageView.frame.size;

    self.nativeAd.nativeAdCoverImageSize =
self.coverImage.frame.size;

    self.nativeAd.nativeAdDescriptionLength = 150;

    self.nativeAd.nativeAdTitleLength = 30;

    self.nativeAd.nativeContent = @"1,2,3,4,5";
    self.nativeAd.useAdapter = NO;

   [self.nativeAd update];

}

@end
```

## Deallocating MASTNativeAd

```
// For deallocating instance of native ad

- (void)dealloc {

    [_nativeAd destroy];

    [super dealloc];

}
```

## Receiving Notification from MASTNativeAd

```
#pragma mark MASTNativeAdDelegate methods

- (void)MASTAdViewDidRecieveAd:(MASTNativeAd *)nativeAd

{

     NSLog(@"Received Native Ad”);

     // Native Ad Rendering Logic

     self.titleLabelView.text = self.nativeAd.title;

    self.descriptionTextView.text = self.nativeAd.adDescription;
    [self.nativeAd loadInImageView:self.coverImage 
withURL:self.nativeAd.coverImageURL];
    [self.nativeAd loadInImageView:self.iconImageView 
withURL:self.nativeAd.iconImageURL];
    [self.ctaButton setTitle:self.nativeAd.callToAction 
forState:UIControlStateNormal];
}

- (void)MASTAdView:(MASTNativeAd*)aNativeAd 
didReceiveThirdPartyRequest:(NSDictionary*)properties 
withParams:(NSDictionary*)params

{

    NSLog(@"Third party ad received”);

   // Third party ad rendering logic

}

- (void)MASTAdView:(MASTNativeAd*)nativeAd 
didFailToReceiveAdWithError:(NSError*)error

{

    NSLog(@"Error encountered : %@”,[NSString stringWithFormat:@"Error :=>\n 
%@",error.localizedDescription]);

}

- (void)nativeAdDidClick:(MASTNativeAd *)nativeAd{

    NSLog(@"Native Ad Clicked");

}
```

## Using Mediation

If you want to use the mediation framework, follow the integration documents of **Facebook Audience Network** and **MoPub**, and set ***self.nativeAd.useAdapter=YES.*** Using this, the Mocean SDK will automatically handle the instantiation of the third-party SDKs and hence, fetching the required details from the third-party SDKs.

Facebook Audience Network (see the native section for more details):
https://developers.facebook.com/docs/audience-network/getting-started

MoPub (see the native section for more details): https://dev.twitter.com/mopub/ios/getting-started

## Downloading Adapters

The adapters for Facebook Audience Network and MoPub will be downloaded along with the Mocean SDK.

Please add these adapters only if you are using the Facebook Audience Network and MoPub as mediation SDKs.

## Facebook Audience Network-specific Change

```
/*Add your device id for Facebook Audience network to get test ads.

* This will be printed in the logs once you launch the application for

* the first time.*/

self.nativeAd.useAdapter = YES;
[self.nativeAd addTestDeviceId:DEVICE_ID forNetwork:kFaceBook]; //Please
change DEVICE_ID with actual ID
```

# MASTAdView Customization

## Customize view appearance

Ad links are opened in Safari by default. To enable the internal browser set the useInteralBrowser property to YES.

Default UIView customization such as animation, background color, orientation/sizing masks, etc. can be used on the MASTAdView.

The MASTAdView instance allows direct access to the ad content container views. These views can be customized but should not have properties adjusted that would affect their behavior in the MASTAdView view.

## Customize ad network properties

By default the Mocean ad network is used. To use a different network change the adServerURL property to the URL of the desired network. The network is expected to follow the same interface and implementation as the Mocean ad network.

To supply additional parameters or override SDK defaults set ad network parameters using the adRequestParameters property. All parameter keys and values must be NSString objects. The ad request parameters can be found here: http://developer.moceanmobile.com/Mocean_Ad_Request_API

## Location detection

The SDK can automatically determine the user's location using the iOS Core Location framework. This feature is disabled by default and can be enabled with the setLocationDetectionEnabled: methods. Note that if the application has no other location detection support from Core Location iOS will prompt the user to allow the application access to the devices location.

Developers that wish to reuse existing application location information can do so by setting location parameters for the ad network. See the section above for setting custom ad request parameters.

## Custom close button

The SDK includes a default close button used for expanded and interstitial ads. Developers can use the MASTAdView delegate message to override the default button and provide a custom, application themed button.

# Content Updates

MASTAdView updates content only by the following methods:

1. Calling the update method.  Use this after initializing and during display of the owning view controller.

2. Calling the updateWithTimeInterval: method.  This method will cause the SDK to update every interval seconds.  Interacting with the current ad suspends the timer.  This can be due to a user expanding the ad, clicking and viewing publisher content with the internal web browser or if the user's action leaves the application.

Call the cancel method to stop ad loading and cancel any timers.

# Detecting Updates and Failures

Sometimes a developer might want to take a special action if no ad is available that satisfies the current constraints sent to the mobile ad server. This might occur if a particular ad type or minimum size was requested, and no matching ad is available. This could also happen if all ads scheduled for the requested zone have reach the maximum daily or monthly cap. Developers can also take advantage of a successful ad update to redisplay a hidden banner or to show interstitials after the ad is downloaded.

The SDK includes an optional *MASTAdViewDelegate* protocol which applications can implement to receive notifications when download related ad events occur. This protocol includes the following methods that relate to ad download status in addition to others not described in this section:

- *MASTAdViewDidReceiveAd:* which is invoked after the ad content has been downloaded successfully.
- *MASTAdView:didFailToReceiveAdWithError:*  which is invoked if downloading ad content fails for any reason.

# Troubleshooting

## Ad content loading issues

1. Verify the specified content zone has ad content.

2. Implement the ad view's delegate and debug any ad download failure errors.

3. Enable simple test banners by setting the testMode property to YES.

# Next Steps

More thorough, complex examples and additional use cases in the sample application distributed with the SDK. Both the sample app and the SDK itself are available in source code form from: http://code.google.com/p/mocean-sdk-ios/.

Additional documentation, information, and other supported platforms on our developer wiki at: http://developer.moceanmobile.com/Main_Page.