Google AdMob Ads Mediation Adapter Development Kit Guide for iOS (Beta Release)

Last updated: 3/26/2012

Table of Contents

Background and Motivation

The AdMob Mediation product allows App publishers to make ad requests to multiple Ad

Networks, maximizing their earnings by increasing their fill rate. The integration point with the AdMob Mediation product is primarily in the iOS SDK. There is no server-side integration.

The client component of the AdMob Mediation product is included in the Google AdMob Ads SDK (**Mediation SDK** for short). The Mediation SDK makes use of the SDKs from individual Ad Networks (the **Ad Network SDK**) to make the actual ad request. Because the interfaces and workings of each Ad Network SDK are different, there is a need for intermediary code that sits between the Mediation SDK and the Ad Network's SDK. This way, the Mediation SDK can interact with a known protocol interface when it needs to communicate with the SDKs from different Ad Networks. The **Ad Network Adapter** is the bridge code that sits between the Mediation SDK and your Ad Network SDK.

If you would like to integrate your Ad Network with AdMob Mediation, we ask that you provide the Ad Network Adapter code for your Ad Network SDK. Writing and maintaining the Ad Network Adapter code allows you more control over how your ad is requested, and the ability to make changes should anything change in your Ad Network SDK.

We provide a customized Mediation Adapter Development Kit to you that you can use to develop your Ad Network Adapter. The Kit contains a skeleton Xcode project that you can use to start developing your Adapter. This document describes how our iOS Ad Network Adapter works, and provide details about the Development Kit.

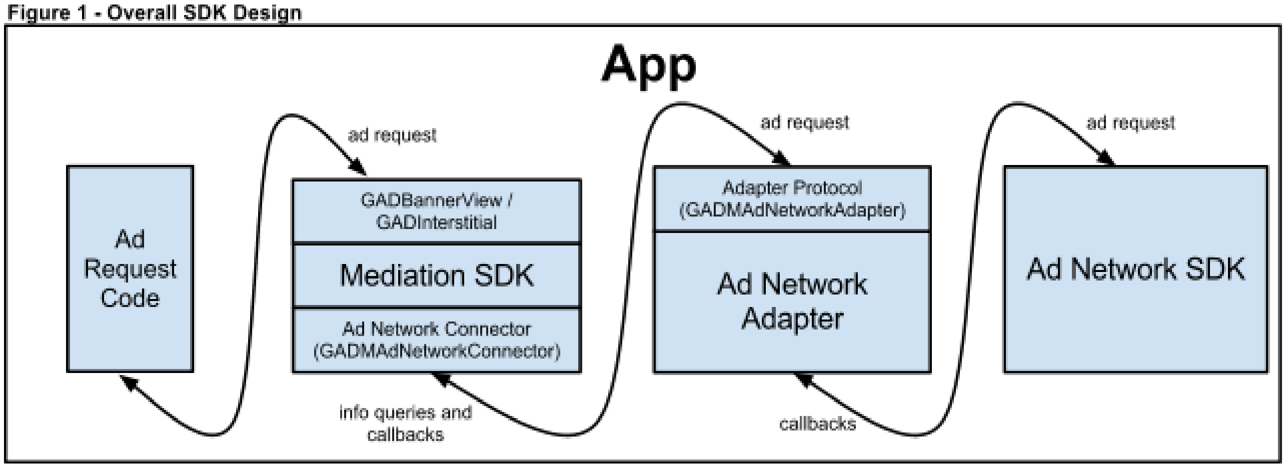
How It Works

To understand how Mediation works in the App, consider the components involved. The App interacts with the Mediation SDK, which relies on the Ad Network Adapters to make ad requests. Each Adapter then in turn interacts with the Ad Network SDK to do the actual ad request, and handle user interactions. Figure 1 is a high-level diagram of how the different pieces fit together. Details are discussed below.

Classes and Protocols

The publisher-facing objects of the Mediation SDK are GADBannerView and GADInterstitial. The Ad Network Adapter implements the GADMAdNetworkAdapter protocol. The Ad Network Adapter calls back into the Mediation SDK by interacting with an object that implements the GADMAdNetworkConnector protocol.

Figure 1 - Overall SDK Design


App
Ad Request Code
ad request
GADBannerView / GADInterstitial
Mediation SDK
Ad Network Connector (GADMAdNetworkConnector)
info queries and callbacks
ad request
Adapter Protocol (GADMAdNetworkAdapter)
Ad Network Adapter
callbacks
ad request
Ad Network SDK

Ad Types

Two modes of ads are supported by Mediation: the “**Banner**” ad and the “**Interstitial**” ad.

“Banner” ads are displayed as part of the App user interface. Users tap the ad to open a modal view that contains a web page or video, or initiate other actions, such as launching the App Store. “Banner” ads may come in any size. We have defined a few standard sizes in GADAdSize.h:

- **kGADAdSizeBanner**, here we refer to a specific format, typically the 320x50 ad, for use in iPhone UIs.
- **kGADAdSizeMediumRectangle**, typically 300x250.
- **kGADAdSizeFullBanner**, typically 468x60.
- **kGADAdSizeLeaderboard**, typically 728x90.
- **kGADAdSizeSkyscraper**, typically 120x600.

We also have special sizes:

- **kGADAdSizeSmartBannerPortrait**, full width of the device in portrait with a height based on the device size.
- **kGADAdSizeSmartBannerLandscape**, full width of the device in landscape with a height based on the device size.

To find the CGSize from these predefined constants, use CGSizeFromGADAdSize().

Interstitial ads take over the whole UI. They are usually shown on App load or during UI transitions.

Publisher Set Up

The publisher will need to supply a Mediation ID to request any ads, obtained at mediation.admob.com. If you would like access to mediation.admob.com, please contact us. We can provide access so you can use the Ad Mediation configuration UI for testing. We can also create an entry for your ad network, initially visible only to you, so you can perform end-to-end tests.

The publisher will also have to integrate the Mediation SDK into their App. Even though in this document it is called the Mediation SDK, it is in fact one and the same as the Google AdMob Ads SDK, with ad mediation capabilities built in. For more information on how publishers integrate the Google AdMob Ads SDK, refer to the Google AdMob Ads SDK Developer’s Guide.

The Guide covers both Banner and Interstitial Ads.

In addition to the steps outlined in the Developer's Guide, publishers will have to link the Adapters and SDK binaries, and set the "-ObjC" linker flag as described in the Static Libraries section.

Static Libraries (.a Files)

The GoogleAdMobAds.a static library contains the Mediation SDK. It is distributed in the Google AdMob Ads SDK package. The Ad Network Adapters and the Ad Network SDKs are distributed in separate static library files. Publishers will have to link several libraries and frameworks into their App binary. They include GoogleAdMobAds.a, the Adapters and Ad Network SDK libraries for all the Ad Networks they want to mediate, and all the frameworks required by the Google AdMob Ads SDK and the Ad Network SDKs. The Google AdMob Ads SDK requires the following frameworks:

- AudioToolbox
- MessageUI
- SystemConfiguration
- CoreGraphics

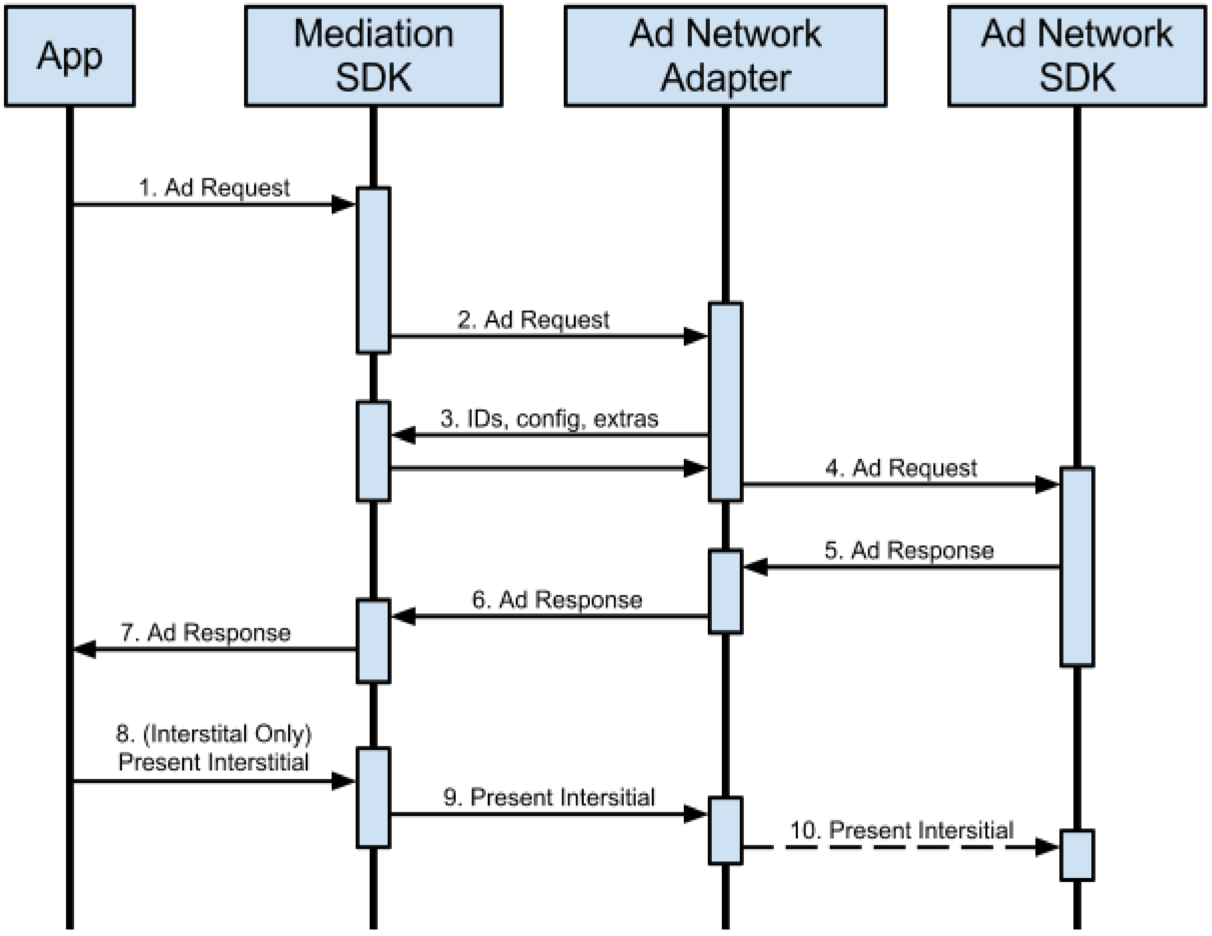
The Mediation SDK discovers the Ad Network Adapter class using the NSClassFromString() function. Because of how linking works for Objective-C, publishers will have to set the "-ObjC" flag in the "Other Linker Flags" of the Build Settings for their App.

You may distribute your Ad Network Adapter as a separate static library file from your Ad Network SDK library file. Or more preferably, as a single library file, where the Adapter and SDK are compiled into one .a file. This way, publishers who integrate your Ad Network into their App with Mediation will have one less .a file to juggle. The Mediation Adapter Development Kit offers support for both. Refer to Adapter Development Kit Contents for details.

Ad Request Flow

Ad requests are initiated from the publisher's App. Figure 2 is a sequence diagram of the ad request flow. The sequence is similar for Banner Ads and Interstitial Ads, and the difference is in the classes and methods used. Refer to the steps in the sequence diagram in the description below.

Figure 2 - Ad Request Flow



Banner Ad Request Flow

1. The App requests an ad through the Mediation SDK, by creating a GADBannerView object with the publisher's Mediation ID, and calling the -loadRequest: method with a GADRequest object.
2. The Mediation SDK contacts Google's servers to retrieve the publisher's settings (not shown in the diagram). The server responds with a list of Ad Network Adapter class names from which to request ads, and related configurations such as the ad type and publisher IDs. The Mediation SDK then creates the Ad Network Adapter using the -initWithGADMAdNetworkConnector: method. Once the Ad Network Adapter object is created, the Mediation SDK initiates an ad request by calling -getBannerWithSize: on the

Adapter, passing in a GADAdSize for a banner ad. This method should be called on the main run loop.
Note that the Ad Network Adapter is expected to keep the GADMAdNetworkConnector object in an instance variable for use later in the ad request process. However, the Ad Network Adapter should never retain the object. Doing so may create circular references, and may lead to memory leaks.

3. The Adapter then obtains additional information, such as publisher IDs and extra information, from the Mediation SDK, by calling the methods in the object that implements the GADMAdNetworkConnector protocol.
    a. Most ad networks have only one such parameter. In this case, call the -publisherId: method to retrieve this parameter.
    b. If there are multiple parameters, call the -credentials: method which returns a dictionary of key-value pairs. You should provide the key names when you register with Google AdMob as an ad network.
    c. Additional parameters set by the publisher are obtained by calling -networkExtras on the connector.
4. With the information obtained in the last step, the Ad Network Adapter then makes the ad request by calling into the Ad Network SDK. Ad requests are typically asynchronous, so the call is expected to return immediately, without waiting for the ad to come back from the network. The Adapter will register itself with the Ad Network SDK as a delegate, or a notification receiver.
5. When there is an ad, or when the ad request fails, the Ad Network Adapter should receive callbacks.
6. The Adapter should then pass the ad response callbacks back to the Mediation SDK, by calling the appropriate methods in GADMAdNetworkConnector, such as adapter:didReceiveAdView:.
7. Upon receiving the callbacks, the Mediation SDK will make the corresponding callbacks to the object that implements the GADBannerViewDelegate protocol, which is implemented by the App. An example is adViewDidReceiveAd:.

Interstitial Ad Request Flow

1. To create an Interstitial Ad, the App creates a GADInterstitial object, and calls the -loadRequest: or the -loadAndDisplayRequest:usingWindow:initialImage: method with a GADRequest object.
2. The Mediation SDK retrieves information from the server, and creates the Ad Network Adapter object similar to the Banner Ad Request Flow. The SDK then calls the getInterstitial: method of the Adapter.
3. The Adapter then obtains additional information, such as publisher IDs and additional parameters, from the Mediation SDK, by calling the methods in the object that implements the GADMAdNetworkConnector protocol.
    a. Most ad networks have only one such parameter. In this case, call the -publisherId: method to retrieve this parameter.
    b. If there are multiple parameters, call the -credentials: method which returns a dictionary of key-value pairs. You should provide the key names when you

register with Google AdMob as an ad network.
   c. Additional parameters set by the publisher are obtained by calling -networkExtras on the connector.
4. With the information obtained in step 3, the Ad Network Adapter then makes the ad request by calling into the Ad Network SDK. Ad requests are typically asynchronous, so the call is expected to return immediately, without waiting for the ad to come back from the network. The Adapter will register itself with the Ad Network SDK as a delegate, or a notification receiver.
5. When an interstitial ad is returned, or when the ad request fails, the Ad Network Adapter should receive callbacks.
6. The Adapter should then pass the ad response callbacks back to the mediation SDK, by calling the appropriate methods in GADMAdNetworkConnector, such as intersititalDidReceiveAd:.
7. Upon receiving the callbacks, the Mediation SDK will make the corresponding callbacks to the App, to the object that implements the GADInterstitialDelegate protocol. For a successful interstitial ad call, the interstitialDidReceiveAd: method will be called.
8. If the App calls the loadRequest: method, it needs control for when to show the interstitial. It does so by calling the presentFromRootViewController: method of the GADInterstitial object. Note that the App may call loadRequest: proactively during user interaction, such as when the user is playing a level of a game. The App may then keep the interstitial object around until the game is finished, and display the ad during a cut screen. Hence, you should never display the interstitial on behalf of the user.
9. The Mediation SDK then calls the -presentInterstitialFromRootViewController: method of the Ad Network Adapter. It does so whether the app calls the -loadRequest: method or -loadAndDisplayRequest:usingWindow:initialImage: method.
10. The Adapter should then arrange to show the interstitial using the supplied UIViewController. Note that you should not show your interstitial automatically. You should wait until -presentInterstitialFromRootViewController: is called and then present your interstitial from the view controller provided by the method.

Reporting

AdMob Mediation keeps track of ad impression and click information for publishers. It does so by relying on the Mediation SDK to report to the Mediation server whenever there is an impression and click. To detect impressions and clicks, the Mediation SDK relies on callbacks made by the Ad Network Adapter. To ensure publishers get accurate reporting information, you should make sure your Ad Network Adapter makes the proper callbacks at the right event.

For Banner Ads, the Mediation SDK reports the following events to the Mediation server:

- **Impression**: The Mediation SDK reports an impression when your Ad Network Adapter calls the -adapter:didReceiveAdView: method of GADMAdNetworkConnector .
- **Click**: Touches or taps are considered clicks. The Mediation SDK reports a click when your Ad Network Adapter calls the -adapter:clickDidOccurInBanner: method of GADMAdNetworkConnector. Note that only one click will be reported for any given ad.

For Interstitial Ads, only the concept of impressions makes sense. There is no “click” action that launches the interstitial. The Mediation SDK will report an impression when your Ad Network

Adapter calls the -adapterWillPresentInterstitial: method of GADMAdNetworkConnector.

User Interaction

When Ads are shown in the UI, the Ad Network SDK is responsible for handling touches to the Ad.

Modal View

For Banner Ads, the Ad Network SDK may want to show a modal view on touch. The way to present a modal view is by calling the -presentModalViewController:animated: method of the view controller of the current UI. In iOS 5 and later, the preferred method is -presentViewController:animated:completion:.

To obtain a UIViewController that the Ad Network SDK can use to present a modal view, the Ad Network Adapter can call the -viewControllerForPresentingModalView method of the GADMAdNetworkConnector.

Callbacks

The App may want to be notified when something takes over the UI, such as when a modal view is shown, or when the App is sent to the background. The Mediation SDK offers two sets of callbacks for Apps, one for Banners and the other for interstitials. They are defined in GADBannerViewDelegate and GADInterstitialDelegate respectively.

The Ad Network Adapter should make similar callbacks to the Mediation SDK, so they can be passed back to the App. The Ad Network SDK may not have all the callbacks that are defined. In this case, make sure the Adapter calls as many of the following GADMAdNetworkConnector methods as applicable:

Common Callbacks

`-adapterWillLeaveApplication:`
    Call this this method when user action will result in leaving the app.

Banner Ad Callbacks

`-adapterWillPresentFullScreenModal:`
    Call this method when a full-screen modal view appears (typically after a user taps on the banner).
`-adapterWillDismissFullScreenModal:`
    Call this method when the full-screen modal view is about to be dismissed.
`-adapterDidDismissFullScreenModal:`
    Call this method when the full-screen modal view is dismissed.

Interstitial Callbacks

`-adapterWillPresentInterstitial:`
    Call this method when the adapter is about to present the interstitial.
`-adapterWillDismissInterstitial:`
    Call this method when the adapter is about to dismiss the interstitial.
`-adapterDidDismissInterstitial:`

Call this method when the interstitial has been dismissed.

Ad Request Information

Publishers may decide to pass in information about the App user to the Ad Networks. This is so more relevant ads may get returned, thereby increasing the likelihood of clicks. They do so by setting the appropriate properties in GADRequest, such as the gender property. The GADRequest object will then be passed to the GADBannerView or GADInterstitial during ad requests.

You may access this information from the GADMAdNetworkConnector object. The methods prefixed with the word -user return information set by the app in GADRequest.

Ad Network Extras

If you want your publishers to pass more information than what is available in the Connector object, you may ask them to pass those in an object that implements the GADAdNetworkExtras protocol.

1. You should define an ad network extras class that implements the GADAdNetworkExtras protocol.
2. In the class, you may define any methods and properties that you want the publishers to set.
3. Return the class type in the +networkExtrasClass method of your ad network adapter.
4. Provide a header file with the class definition to your publishers.
5. Instruct the publishers to instantiate the class and set any appropriate information.
6. Ask them to pass the instance of the class to the -registerAdNetworkExtras: method in GADRequest.
7. Finally, you can get the ad network extras object by calling the -networkExtras method in the GADMAdNetworkConnector.

Location Information

Publishers may encounter issues with App Store approval if their App uses location information primarily for ads. When user's location is queried using CoreLocation, a dialog will appear to ask for the user's permission. If the App has no apparent reason to ask for location information, it may appear to be out of place for the user. Apple may reject the publisher's App due to this reason.

However, having location information may allow Ad Networks to return more relevant ads. To balance these competing goals, the Mediation SDK (and by extension, the Google AdMob Ads SDK) does not require the publisher to include CoreLocation.framework. Instead, publishers who have legitimate access to location information should make their location request during the normal course of their App. During ad request, they may then pass the previously requested location information to GADRequest, using the `setLocationWithLatitude:longitude:accuracy: method. The location information is then made available to Ad Network Adapters in GADMAdNetworkConnector. This way, Apps that do not need to access location information do not have to include CoreLocation.framework to use the Mediation SDK.`

You are encouraged to use the location information made available in GADMAdNetworkConnector, instead of using CoreLocation directly in the Ad Network SDK.

If your Ad Network SDK does use CoreLocation directly, and there is a BOOL flag in your Ad Network SDK interface that controls whether the location information is requested, you may provide a way for your publisher to control this flag using the Ad Network Extras mechanism described above.

Refresh

The Mediation SDK has a built-in ad refresh mechanism for Banner Ads. It refreshes in set intervals such as every 60 seconds. When the Mediation SDK refreshes, it fetches the Mediation settings from the server, and make ad requests from Adapters. That corresponds to step 2 and onwards in the sequence diagram above. Publishers can modify their refresh interval at mediation.admob.com .

If your Ad Network SDK can refresh itself, turn it off if possible. The refreshes between the Mediation SDK and the Ad Network SDK may result in unpleasant Ad transitions and short ad impressions. For example, let’s say the Mediation SDK will refresh 60 seconds from now, and the Ad Network SDK will refresh 56 seconds from now. 56 seconds later, the Ad Network SDK refreshes itself, but the new ad is only shown for 4 seconds before being rotated out by a new ad from the Mediation SDK.

Banner View Transition

When an ad refreshes, the Mediation SDK may apply transition animations between the old and the new ad. The enum GADMBannerAnimationType contains a list of possible Banner Ad animations. Which animation to use is sent from the Mediation server, and is configurable by the publisher.

If for some reason your ad view does not work properly with certain kinds of transition animation, you may opt out of transition animations between ads for some or all animation types. You do so by returning a BOOL from the isBannerAnimationOK: method of your Ad Network Adapter, given an animation type.

Adapter Development Kit Contents

The Adapter Development Kit contains the following:

- Skeleton code for an Ad Network Adapter.
- A Test App to exercise the Ad Network Adapter.
- A copy of the GoogleAdMobAdsSDK.

Xcode Project

The Xcode project contains build targets that you can use to build the Adapter and/or SDK library, to make it distributable to publishers. It initially contains a skeleton code for writing your adapters. There is also a Test App that you can use to exercise the Adapter. Finally, we provide a unit test framework for you to test the Adapter. You will find detailed descriptions of the Xcode project below.

You may opt to use this Xcode project to develop your adapter. Alternatively, you may import the adapter code and all the relevant headers into your SDK project, and make the adapter part of your SDK. The adapter is an additional class in your static library. In a lipo'd binary with three architectures, it typically adds less than 200Kb to your static library.

Ad Network SDK

This is an empty group initially. You can drag in the necessary headers and libraries into this group. Make sure your SDK's .a file are only included in the "Adapter SDK Lib" target and nothing else.

Mediation Protocols

These are Protocols required for Adapter development. You can find them under the Mediation group. They are:

`GADMAdNetworkAdapter in GADMAdNetworkAdapterProtocol.h:`

This is the protocol you have to implement for your Ad Network Adapter. Read through this file for detailed description for each method.

`GADMAdNetworkConnector in GADMAdNetworkConnectorProtocol.h:`

This is the protocol that your Adapter will have to interact with. The Mediation SDK implements this protocol. Your Adapter will be passed an object implementing this protocol in your constructor. Your Adapter should obtain necessary information for ad requests from this object. Some of the information are enums, and they are defined in GADMEnums.h. Your Adapter must also make callbacks to this object to report ad request status and user interaction with the ad.

`GADMEnums.h`

Contains a list of ad transition animation types.

The following protocols (found under GoogleAdMobAdsSDK group) are optional:

`GADAdNetworkExtras.h`

Implement to allow publishers to define client side settings for your ad network. For example,
define a class MyExtras that implements GADAdNetworkExtras. Distribute MyExtras.h to your publishers and return [MyExtras class] for -networkExtrasClass in your implementation of GADMAdNetworkAdapter. For details, see the Ad Network Extras section above.

Adapter

Under the Adapter group, you'll find a skeleton implementation of your Adapter. You should modify the file GADMAdapter<Name>.m where <Name> is your Ad Network's name. Go through each method and wire them up to your Ads SDK. Be sure to make your Adapter a delegate of your Ads SDK, and pass along all callbacks to the connector object. Remember to use the connector object to get any information associated with the request, such as the publisherId, credentials, demographic information, etc. Remove any boilerplate code that you don't need.

Build Targets

There are several build targets in this project:

- **Adapter SDK Lib**: This builds the adapter for a single architecture (i386 or arm) and bundles your Ads SDK code into one .a library.
- **Fat Adapter SDK Lib**: This runs Adapter SDK Lib multiple times with different architectures, and runs lipo to bundle the different .a files into a single library file. After this target is built, you can find the .a file under the Library/ directory. We recommend distributing this .a file to publishers, since this one library file contains all of your Ads SDK logic and the Adapter.
- **Adapter Lib**: This builds the adapter in a single architecture and produces a .a library. This library will not contain any of your Ads SDK code.
- **Fat Adapter Lib**: This runs the Adapter Lib target with several architectures and packs them into a fat .a library using lipo. This library will not contain any of your Ads SDK code. After this target is built, you can find the .a file under the Library/ directory.
- **Test App**: This builds the Fat Adapter SDK Lib, then links the resulting .a file and builds the Test App.
- **Automated Tests**: By building this target, you exercises your unit tests.

Test App

The Test App is an application that exercises your Adapter and, in turn, your Ad Network SDK. You use it to make ad requests via the adapter and display them. There are also visual cues on whether your callbacks are being received.

The AdViewScreenController object implements the GADMAdNetworkConnector protocol. You typically do not need to modify much of the TestApp, unless you have multiple values for credentials. You may modify the credentials method to return credentials that your Ads SDK requires. Otherwise, you can get Test App to use your publisher ID by setting the ADAPTER_PUBLISHER_ID environment variable. You can do this by editing the "Run" action of your Scheme. In Xcode, click on the "Product" menu, then select "Edit Scheme...". Select the "Test App" Scheme on top, and the Destination (Simulator/Phone and iOS version). Select the “Run” action on the left, and click on the “Arguments” tab. Under “Environment Variables”, click the “+” button to add an environment variable.

Automated Tests
`GADMAdapter<Name>Test.m is a Google Toolbox for Mac unit test file. Modify this file to add new unit tests for your Adapter. For more information about Google Toolbox for Mac, see` http://code.google.com/p/google-toolbox-for-mac/ .

GoogleAdMobAdsSDK
All of the headers and libGoogleAdMobAds.a are included in the Adapter Development Kit for your reference.