ADITION iOS Ad SDK Integration Guide for App Developers

SDK Version 15 as of 2013·07·26

Table of Contents

1 Ad SDK Requirements

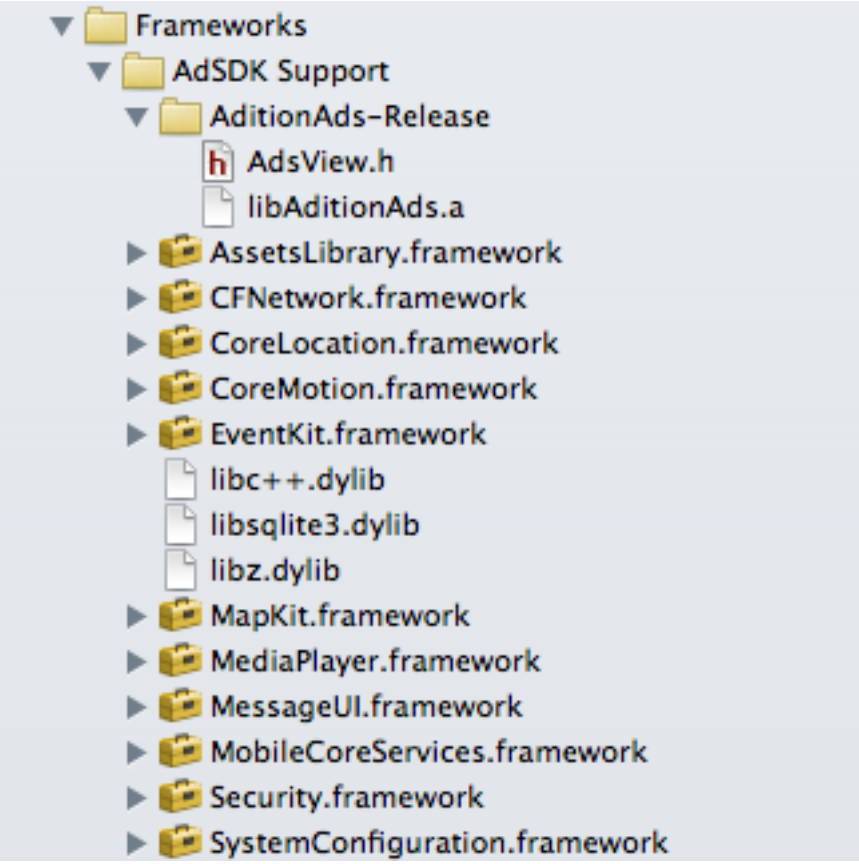
There is only one hard requirement for using the Ad SDK: Apps that make use of the SDK must be built for iOS version 4.2 or later. Beyond that, there's only the recommendation that you use Xcode 4.2 or later as your development environment.

The functionality implemented in the SDK is based on the MRAID standard, as defined in the IAB document https://www.iab.net/media/file/IAB_MRAID_v2_FINAL.pdf. Although MRAID is aimed at SDK implemetors and rich media authors, knowing the contents of this document is certainly helpful for an app developer, albeit not a strict requirement.

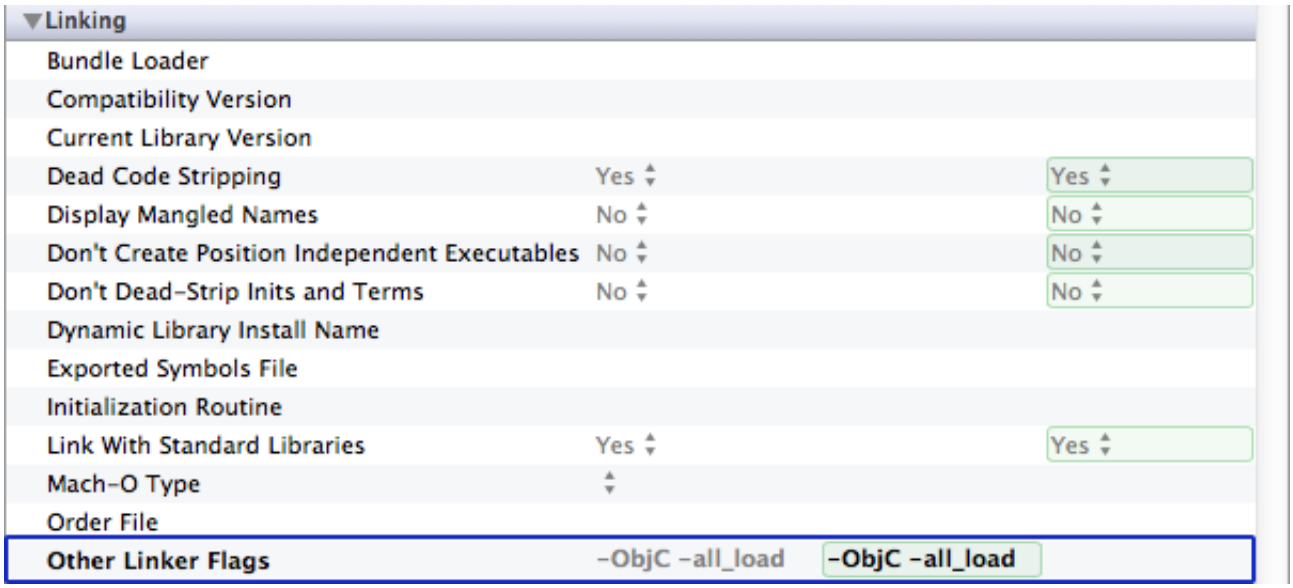
2 Configuring Xcode

In order to use the Ad SDK, you need to add several files to your Xcode project, as shown in the screenshot below:

- The static library containing the SDK (`libAditionAds.a`), and the header file that comes with it (`AdsView.h`).
- Cocoa frameworks and libraries, supplying the classes and APIs that the SDK uses internally.



Additionally, you need to set the `-all_load` and `-ObjC` linker options in the Other Linker Flags setting of Xcode's Build Settings, as shown in the following screenshot:



3 Compatibility Notes

The SDK comes wrapped as a static library with binary code for multiple CPU architectures (“universal binary”). One of the supported architectures is `i386` so you can run an app that links to the library in the simulator. The other supported architectures are currently `armv6` (iPhone 3G and earlier) and `armv7` (iPhone 3GS and later). The newer `armv7s` architecture (for iPhone 5 and later) is not included yet since that would mean dropping `armv6` support.

4 Displaying an Ad

To display an ad, you create an instance of the `AdsView` class and make that view a part of your app's layout. As a minimum, you need to provide the following information and infrastructure, in the sequence shown:

- You initialize the `AdsView` (via InterfaceBuilder or programmatically) with a size (width × height in points) that matches one of the available ad formats.
- You assign a delegate to your `AdsView` that implements the mandatory methods of the `AdsViewDelegate` protocol (see the `AdsView.h` header file). The view controller for the `AdsView`'s parent view would be a natural choice for an `AdsViewDelegate`.
- If necessary, you indicate interstitial placement for the ad by passing a `YES` argument to the `-[AdsView placeAsInterstitial:]` method. An interstitial is a modal full-screen ad that blocks the workflow of your app until the ad is closed. Because of this, interstitals always need the active participation of the host app for preparing the ad's display and resuming the app workflow after the ad is closed. Only ads that are typed as interstitial in the ad server are eligible for interstitial placement.
- You pass an ad server network ID and a content unit ID to the `-[AdsView loadCreativeFromNetwork:withContentUnitID:]` method to start loading an ad.
- You track the actual visibility of the `AdsView` in your app layout and pass the new status to the `-[AdsView considerAdOnscreen:]` method when visibility changes. In the simplest of cases, you can just pass `YES` from inside your implementation of `-[AdsViewDelegate adDidShow:]`.

Your publisher can provide you with relevant network IDs, content unit IDs and available ad size formats.

Note that an `AdsView` must be part of a view hierarchy that is rooted in a visible window before you can start loading an ad. So if you want an `AdsView` to be invisible until it's loaded, you should do so by setting its origin to an off-screen value (but not by leaving it out of a view hierarchy altogether).

Optionally, before loading an ad, you can add targeting information to an existing `AdsView` using its `targeting` component. The `targeting` object has properties you can set to further specify an ad's desired characteristics:

- `humanLanguage`

  If your app offers its own language selection, you assign a 2- or 5-character locale code (like “`en`”, “`de`”, or “`en_GB`”) for the user's language choice. The ad server will prefer ads that are marked with a matching language code. If you don't assign a value, ad selection will use the system language the user has designated in the Settings app.

- `maximumSize`

  In order to allow ads to use the MRAID `resize()` function, you need to assign the maximum allowed size, in points, to this property. If you assign a size, the creative will be able to resize up to the given maximum width and/or height. If you don't assign a size, the creative will not be allowed to resize.

- `trackingGroupName`

  The usage tracking group name acts like a namespace for anonymously tracking app usage on an individual device. If you assign the same name in several apps, requesting an ad in one of those apps will generate the same tracking ID on a specific device (but different IDs on different devices). The default value is the app's bundle identifier, which will produce different tracking IDs for each app on the same device. Since the tracking group name must be unique across all app developers, it is recommended that you use a reverse DNS naming scheme similar to your bundle identifier (à la "`com.yourdomain.groupname`"). This is about as close as you can get to anonymous user tracking with scalable granularity across multiple apps. No personally identifiable information is ever transmitted.

- `keyword`

  An ASCII-encoded text with a maximum length of 255 bytes that narrows down which ads can be served. Consult the ADITION user manual to learn more about keyword targeting.

5 Mandatory Delegate Methods

There is only one mandatory method your `AdsViewDelegate` must implement.

- `-adsViewMayUseLocationServices`

  You return a Boolean that states whether your app makes meaningful use of Location Services from the perspective of the end user. Only if your app presents a sensible application of Location Services to the user is the SDK allowed to make use of location data for advertising purposes.

6 Demo Xcode Project

For demonstrating the minimum effort needed to display an ad, we have added a demo Xcode project, `SimpleAdDemo`. You open the project in Xcode 4.2 or later, select an iPhone simulator scheme and choose Run from the Product menu to see it in action. Note that you need an internet connection for this to work.

The project's `ViewController` class contains the relevant demo code. In about one screenful of Objective-C, all necessary steps for displaying a simple ad are laid down.

The source code also demonstrates how you can make Safari open an external web site after the user taps an HTTP link in an ad.

7 Symbol Names Used By The SDK

The following shows the class and category names the SDK uses. To avoid “duplicate symbol” errors when linking your app, you should not use any of these names in your app project.

7.1 Category Names

```
(AdSDK)
(Adition)
(FMDatabaseAdditions)
(JSONKitDeserializing)
(JSONKitSerializing)
(JSONKitSerializingBlockAdditions)
(Private)
```

7.2 Class Names

```
ASIAuthenticationDialog
ASIAutorotatingViewController
ASIDownloadCache
ASIFormDataRequest
ASIHTTPRequest
ASIInputStream
ASINetworkQueue
AditionAdsLibTests
AdsAdminDatabase
AdsAdminRecord
AdsAssetRecord
AdsDownloader
AdsFileCache
AdsHeadlessBrowser
AdsJavaScriptBridge
AdsMediaPlayer
AdsSecondaryView
AdsTargeting
AdsView
AdsWebBrowser
AlertAgent
AlertAgentInternalDelegate
FMDatabase
FMResultSet
FMStatement
JKArray
JKDictionary
JKDictionaryEnumerator
JKSerializer
JSONDecoder
MRAIDReachability
```

7.3 Third-Party Source Packages

Some of the symbol names listed above stem from the use of the following third-party source packages in the SDK. See the LICENCE file for the respective licenses.

```
ASI-HTTP-Request
FMDB
JSONKit
```

8 Change Log

Changes fom SDK version 14 to version 15:

- The SDK now supports attaching `UIGestureRecognizers` to `AdsViews`. Using gesture recognizers, the host app can intercept any special gestures that started on the `AdsView` but are meant to be handled by the app itself. Gesture recognizers that were attached while the `AdsView` is in the “default” state are active in the “resized” and “expanded” state as well.
- Removed the following `AdsViewDelegate` callback methods:
  - `adCannotComposeMail:`
  - `makeCallToPhoneNumber:`
  - `openAppStoreURL:`

  The app developer no longer has to take care of these cases.

Changes fom SDK version 12 to version 14:

- Removed the `-[AdsViewDelegate adsViewOwningViewController]` callback method. The host app no longer needs to supply this information.

Changes fom SDK version 9 to version 12:

- Added the `-[AdsView placeAsInterstitial:]` method.

Changes fom SDK version 8 to version 9:

- The `-adsViewMayUseLocationServices:` delegate method was revamped to no longer take an argument (using Location Services or not doesn't depend on an individual `AdsView`).

