iOS Amazon Mobile Ads SDK

Date: 2012-04-17
Version:1.6.1

Table of Contents

Welcome

This guide contains information for you to add Amazon Mobile Ad Network support to your iOS apps.

Important
This guide assumes that you are an iOS developer.

What is Amazon Mobile Ad Network?

Amazon Mobile Ad Network connects Amazon's advertisers with qualified publishers and developers. Advertisers can now target Amazon customers on 3rd party applications as well as Amazon websites and mobile applications. Publishers and developers can join the Amazon Mobile Ad Network to display highly targeted ads within their mobile applications and mobile websites.

The Amazon Mobile Ad Network allows you to participate in highly targeted campaigns with premium advertisers. In addition to traditional targeting based on device type, operating system, and location, our network allows advertisers to target based on intent (over 400 "In Market Segments"), interest (over 50 "Lifestyle Segments"), and demographics. This rich targeting capability allows you to earn higher eCPM vs. traditional less targeted ad networks.

Available Tools, Libraries, and Interfaces

The Amazon Mobile Ad network provides a Java SDK for Android 1.6 and newer, and Objective C SDK for iOS 4.3 and newer.

Amazon Mobile Ad Network Concepts

Application ID

The Application ID is a 32 character globally unique alphanumeric string, which is used to identify your app. The same Application ID can be used across platforms, but each distinct app should have a unique Application ID. For example the IMDb app on iOS and Android use the same Application ID, but the IMDb Trivia app has a different Application ID.

Ad Sizes

The Amazon Mobile Ad Network supports the following standard static image and expandable banner sizes for:

Phone Apps

- 300x50
- 320x50
- 300x250

Tablets

- 300x250
- 728x90
- 1024x50

App Rotation

The expandable rich media ads target landscape and portrait mode separately. If your app supports device rotation events, your app should reload the ad when rotating between portrait and landscape mode. The SDK passes the device orientation back to the ad server to pick an appropriate ad.

Above the Fold

All ad placements must be above the fold, which means fully visible without scrolling. If an ad placement is embedded with content in a screen, it can scroll with that content, and scroll off of the screen as long as it started above the fold.

If you are interested in discussing support for below the fold placements, please contact your account manager.

Quickstart Guide

The Quickstart Guide provides you step-by-step instructions for incorporating Amazon Mobile Ads into your app. The Amazon Mobile Ads SDK currently supports both static image banners and expandable rich media banners with videos.

Before You Use the Amazon Mobile Ads SDK

The Amazon Mobile Ads SDK for iOS requires iOS 4.3 or later, assumes you have Xcode 4.5 installed, and are familiar with iOS development.

Account Registration

As part of the developer onboarding process your account manager should have provided you with a unique 32-character Application ID for each of your apps. The same Application ID can be used across platforms viz. iOS, Android, but each distinct app should have a unique Application ID. For example the IMDb app on iOS and Android use the same Application ID, but the IMDb Trivia app has a different Application ID.

Enabling Ads in Apps

This section of the quick start guide steps you through adding ads to your iOS app:

1. Incorporate the SDK into your project
2. Register your App
3. Loading an Ad using Amazon Ad View

Step 1 – Incorporate the SDK into your project

Add the AmazonAd.framework to your project

In Xcode:

1. Open your app in XCode.
2. Click on your project file and Go to “Build Phases” -> “Link Binary With Libraries”.
3. Add the AmazonAd.framework from the [AMAZON AD SDK FOLDER]/AmazonAd.framework
4. Add the AdSupport.framework (if not already linked)
5. Add the CoreLocation.framework (if not already linked)

Step 2 – Set Your Application ID

You must set your app’s Application ID in order to receive ads. This allows Amazon to track your impressions and clicks, and associate them with your account. You should have your app call the `AmazonAdRegistration setApplicationId` class method on every app start using the Application ID from the Amazon AppStore. You could add the setApplicationId call in your `AppDelegate’s didFinishLaunchingWithOptions` method.

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
    // Override point for customization after application launch.
   [[AmazonAdRegistration sharedRegistration] setApplicationId:@"0123456789"];
}
```

```
Note: Include the AmazonAdRegistration.h header file in your implementation file.
         #include <AmazonAd/AmazonAdRegistration.h>
```

Step 3 – Loading an Ad using AmazonAdView

To retrieve and display the ad you'll use an `AmazonAdView` to display the view. You can also set a number of optional parameters, including whether this is a test request. Test requests always return an ad and don't count as part of your metrics. After setting the options on your request an asynchronous task requests an ad from the Amazon Mobile Ad Network.

Here's a simplified loadAd call placed in one of the `ViewController's viewDidLoad` method for an app.

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

    // Load an ad

    // Create an AmazonAdView instance with size 320x50.
    AmazonAdView *amazonAdView = [AmazonAdView amazonAdViewWithAdSize:AmazonAdSize_320x50];

    // Set the adOptions.
    AmazonAdOptions *options = [AmazonAdOptions options];
    adOptions.isTestRequest = YES;

    // Register the ViewController with the delegate to receive callbacks.
    amazonAdView.delegate = self;

    // Call loadAd
    [amazonAdView loadAd:options];

    // Ad the AmazonAdView to your viewController's view.
    [self.view addSubView:amazonAdView];
}
```

Here's a how you would implement the `AmazonAdViewDelegate` protocol within your ViewController. The `viewControllerForPresentingModalView` protocol method should be implemented so that the `AmazonAdView` is aware off the ViewController that will be used for presenting/dismissing modal views, such as the browser view when a user clicks on an Ad. You may also choose to lazily add the `AmazonAdView` to your parent view in the `adViewDidLoad` optional callback method.

```
// #pragma mark AmazonAdViewDelegate
// @required
- (UIViewController *)viewControllerForPresentingModalView {
    return self;
}

// @optional
- (void)adViewWillExpand:(AmazonAdView *)view {
    NSLog(@"Will present modal view for an ad. Its time to pause other activities.");
}

// @optional
- (void)adViewDidCollapse:(AmazonAdView *)view {
    NSLog(@"Modal view has been dismissed, its time to resume the paused activities.");
}

// @optional
- (void)adViewDidLoad:(AmazonAdView *)view {
```

```
        NSLog(@"Successfully loaded an ad");
    }

    // @optional
    - (void)adViewDidFailToLoad:(AmazonAdView *)view withError:(AmazonAdError *)error {
        NSLog(@"Failed to load an AmazonAd %d %@", error.code, error.description);
    }
```

```
Note: Include the following header files in your implementation file.
       #include <AmazonAd/AmazonAdError.h>
       #include <AmazonAd/AmazonAdOptions.h>
       #include <AmazonAd/AmazonAdView.h>
```

Where do I go from here?

Please read the reference sections regarding:

- Ad Options on page 6

- Callbacks on page 7
- Errors on page 9
- Ad Sizes on page 9
- on page 10
- Appendix A – Developer Launch Checklist on page 12

Reference

Ad Options

Besides ad unit size, there are a number of options that you can include in the request that's sent to the Amazon Mobile Ad Network. These include gathering latitude and longitude from the device, flagging a request as a test request, and sending the user's gender and age.

To set options you construct an Amazon Ad Options object and pass it to the load ad function call.

Test Requests

During integration and testing of the Amazon Mobile Ads SDK, you should set the testRequest option to true. This indicates that the request is for testing. Test requests will not show up in your metrics.

Important
During development you should always set this flag to true. Test traffic that doesn't include this flag can result in blocked requests, fraud investigation, and potential account termination.

Example:

```
AmazonAdOptions *options = [AmazonAdOptions options];
options.isTestRequest = YES;
```

Including Latitude and Longitude

If your app or the device is enabled to provide the latitude and longitude coordinates, you can configure the Amazon Mobile Ads to provide these values to the Amazon Mobile Ad Network. Requests that contain geographic location may earn higher CPMs.

Example:

```
AmazonAdOptions *options = [AmazonAdOptions options];
options.usesGeoLocation = YES;
```

Specifying Gender

If your app collects gender information from the user, then you can pass that information to the Amazon Mobile Ad Network. Requests that contain gender information may earn higher CPMs.

Example:

```
AmazonAdOptions *options = [AmazonAdOptions options];
options.GENDER = FEMALE;
```

Using Geolocation

If your app already uses the Core Location service, then you can enable the SDK to send location information to the Amazon Mobile Ad Network. Requests that contain location information my earn higher CPMs. The SDK defaults to not send location information.

Example:

```
AmazonAdOptions *options = [AmazonAdOptions options];
options.useGeolocation = YES;
```

Advanced Option

Advanced options can be set as String key value pair through the set advanced option function. The advanced options currently include “ec” for floor price.

Example:

```
AmazonAdOptions *options = [AmazonAdOptions options];
[options setAdvancedOption:@”value” forKey:@"key"];
```

Optional Floor Price

When making `loadAd` requests you can specify a floor price in microdollars. For example if you wanted to earn a minimum of $0.85 per thousand ads returned, then you would specify “850000” microdollars. The floor price is set through the advanced options using the key “ec”.

Important

The Amazon Mobile Ad Network is designed to maximize your revenue opportunity. Setting a floor price may limit your revenue potential as it might prevent several paid campaigns from running on your placement. Amazon recommends not setting this value. In addition, please double check your floor price value as setting a high floor price, even unintentionally, may severely limit your fill rate and revenue, requiring a new app version to fix.

The following example will only returns ads with a CPM of $0.85 or greater.

Example:

```
AmazonAdOptions *options = [AmazonAdOptions options];
[options setAdvancedOption:@”850000” forKey:@"ec"];
```

Callbacks

The SDK includes an AmazonAdViewDelegate protocol that defines methods that a delegate of the AmazonAdView object can implement to infer ad engagements. Ad loaded, ad will expand, ad collapsed and ad failed to load are example of such ad engagement protocol methods. They allow your app to take action based on the current state of the Amazon ad view.

ViewController for Modal Views

The application's ViewController is required to implement this protocol method. The `AmazonAdView` relies on this method to determine which view controller is used to present/dismiss modal views, such as the browser view presented when a user clicks on an ad.

```
- (UIViewController *)viewControllerForPresentingModalView;
```

Ad Loaded

Each time an ad is successfully loaded this callback is called. You can use this to log metrics on ad views and assist with initial integration.

Callback:

```
- (void)adViewDidLoad:(AmazonAdView *)view;
```

Ad Will Expand

After a user clicks on a rich media ad, but prior to expanding, this callback is called. This callback can be used to do things like pause your app or suspend audio prior to expanding the ad.

Callback:

```
-   (void)adViewWillExpand:(AmazonAdView *)view;
```

Important
Do not release the last reference on the AmazonAdView while the ad is expanded or the app will crash upon closing the expanded ad.

Ad Collapsed

After a user clicks on the close ad button on an expanded rich media ad, this callback is called immediately after collapsing the ad. This callback can be used to do things like resume your app or restart audio.

Callback:

```
- (void)adViewDidCollapse:(AmazonAdView *)view;
```

Ad Failed To Load

Whenever an ad fails to be retrieved the Ad Failed To Load callback is called with an error code. An ad can fail for a number of reasons, which can be grouped into two categories: transient and non-transient errors. When an error is transient, you should make another call to load the ad based on a standard retry algorithm, or when conditions change, e.g. network available. When the failure is non-transient, you should log the error and investigate why your app is unable to successfully retrieve an ad.

Callback:

```
- (void)adViewDidFailToLoad:(AmazonAdView *)view withError:(AmazonAdError *)error;
```

Errors

When the ad fails to load the Ad Failed To Load callback returns an error code and message. The Mobile Ads API does not contain any retry logic, so when you encounter a transient error you may want to call `loadAd` again. For example the Network Connection error can be caused by a dropped network connection, so you could successfully make another `loadAd` call when the network is available.

The error codes are:

Error Code	Error Description	Retry-able?
InvalidRequestErrorCode	"Invalid applicationId"	No
OsVersionTooLowErrorCode	"OS Version is too low. Example: < 4.x"	No
InternalServerErrorCode	"Ad not found"	Yes
NetworkConnectionErrorCode	"Network connection error"	Yes

Ad Sizes

The Amazon Mobile Ad Network supports a number of standard sizes for phones and tablets. Your app should request a device appropriate size, which will be scaled to your ad view. The recommended sizes by device are:

The AdSizes are specified during the construction of the AmazonAdView object.

```
// Standard Amazon Ad Sizes for phones.
extern const CGSize AmazonAdSize_320x50;
extern const CGSize AmazonAdSize_300x50;
extern const CGSize AmazonAdSize_300x250;

// Standard Amazon Ad Sizes for tablets.
extern const CGSize AmazonAdSize_728x90;
extern const CGSize AmazonAdSize_1024x50;
```

The recommended sizes by device family are:

Ad Size	iPhone/iPod	iPad
300x50	No	No
320x50	Yes	No
300x250	Yes	No
728x90	No	Yes
1024x50	No	Yes

Testing and Debugging

When integrating with the Amazon Mobile Ads SDK, you can enable logging Debug messages to the Xcode console. To enable them, call

```
[[AmazonAdRegistration sharedRegistration] setLogging:YES]];
```

Important
Logging must be disabled for production releases.

Application Rotation

If your app supports device rotation events, your app should reload the ad when rotating between portrait and landscape mode. However when a rich media ad has been expanded, you should not rotate the app. Failure to prevent the app from rotating will result in multiple ads being displayed on top of each other.

```
- (BOOL) shouldAutorotate {
    if (self.amazonAdView.isAdExpanded) {
        return NO;
    }
    return YES;
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Backwards compatibility with iOS 4.3 and 5.x
    if (self.amazonAdView.isAdExpanded) {
        return NO;
    }
    return YES;
}
```

Ad expansion is presented as modal view. Upon dismissal of the modal view, you should additionally implement dismissModalViewControllerAnimated callback for any necessary custom behaviors such as but not limited to reloading, resizing, and repositioning the contents on your app view.

Universal Apps

We recommend requesting the following ad sizes for iOS devices:

- iPhone: 320 x 50
- iPad: 728 x 90 in portrait mode
- iPad: 1024 x 50 in landscape mode

For universal apps, you should load ads of different sizes based on the current device and supported orientations. The following is a simplified code snippet to do so:

```
@interface ViewController ()<AmazonAdViewDelegate>
{
    AmazonAdView *_amazonAdView;
}

- (void)loadAmazonAdWithUserInterfaceIdiom:(UIUserInterfaceIdiom)userInterfaceIdiom
userInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation;

@property (nonatomic, retain) AmazonAdView *amazonAdView;

- (void)loadAmazonAdWithUserInterfaceIdiom:(UIUserInterfaceIdiom)userInterfaceIdiom
userInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Set the ad options and load the ad
    AmazonAdOptions *options = [AmazonAdOptions options];
    // Test Requests disable metric tracking and typically have a 100% fill rate
    options.isTestRequest = YES;

    // IMPORTANT: Create the AmazonAd view for requesting an ad of appropriate size based on the
current device and orientation if necessary
    if (userInterfaceIdiom == UIUserInterfaceIdiomPhone) {
        if (self.amazonAdView == nil) {
            // Create the Amazon Ad view of size 320x50 for iPhone if we have not already
            self.amazonAdView = [[[AmazonAdView alloc] initWithAdSize:AmazonAdSize_320x50]
autorelease];
        }
    } else {
        // Remove the Amazon Ad view since we may request an ad of different size
        [self.amazonAdView removeFromSuperview];

        if (UIInterfaceOrientationIsPortrait(interfaceOrientation)) {

            // Create the Amazon Ad view of size 728x90 for iPad while in portrait mode
            self.amazonAdView = [[[AmazonAdView alloc] initWithAdSize:AmazonAdSize_728x90]
autorelease];

            // Reposition and resize the Amazon Ad view to center at top
            [self.amazonAdView setFrame:CGRectMake((self.view.bounds.size.width - 728.0) / 2.0,
0.0, 728.0, 90.0)];
        } else {

            // Create the Amazon Ad view of size 1024x50 for iPad while in landscape mode
            self.amazonAdView = [[[AmazonAdView alloc] initWithAdSize:AmazonAdSize_1024x50]
autorelease];

            // Reposition and resize the Amazon Ad view to center at top
            [self.amazonAdView setFrame:CGRectMake((self.view.bounds.size.width - 1024.0) / 2.0,
0.0, 1024.0, 50.0)];
        }

        // Add the newly created Amazon Ad view to our view.
        [self.view addSubview:self.amazonAdView];
    }

    // Load an amazon ad with the given options
    [self.amazonAdView loadAd:options];
}
```

Appendix A – Developer Launch Checklist

Prior to submitting your app to Apple iTunes, your app should pass the following criteria:

Ad Functionality

- [ ] Does the sample Amazon Kindle Ad expand?
- [ ] Does the sample Amazon Kindle Ad open the SDK Browser to Mobile Optimized Amazon Website?
- [ ] Does the sample Amazon Kindle Ad play video?
- [ ] Does the Amazon Cloud Player Ad correctly open the iTunes store?

User Experience

- [ ] Are the sample ads legible? E.g. minimal stretching/skewing
- [ ] When an ad takes a long time to load, is the app's user experience impacted?
- [ ] When an ad fails to load, is the app's user experience impacted?
- [ ] Is the ad banner away from device and app controls?
- [ ] If the ad banner contains a close button, does the button interfere with ad content? (Recommendation: Upper right hand corner)
- [ ] Do automatic ad reloads take at least 60 seconds?

App Configuration

- [ ] Are requests being sent without the test request flag? E.g. App is either not calling `isTestRequest` method or is passing false as the argument
- [ ] Are requests being sent without the debug logging flag enabled? E.g. App is not calling `setLogging` method or is passing false as the argument
- [ ] Are requests being sent with the correct applicationId?

Important
Apps that fail to pass these criteria may be deemed unsuitable and blocked from the Amazon Ad Network.

Appendix B – App Download Conversion Tracking

The Ads API can be used to track app downloads from mobile ad campaigns. When your app starts you make a register call, which sends hashed device ids to our ad network. These ids are joined with the ids of the devices that displayed ads for your download ad campaign. Your download ad campaign metrics will include impressions, clicks, and downloaded app starts.

To add app download conversion tracking to your app, add the `setApplicationId] and registerApp` call whenever your application become active for use. Here's an example

```
- (void)applicationDidBecomeActive:(UIApplication *)application
{
      [[AmazonAdRegistration sharedRegistration] setApplicationId:@"0123456789"];
      [[AmazonAdRegistration sharedRegistration] registerApp];
}
```

Note
If you are already displaying ads from the Amazon Mobile Ad Network, then the `registerApp` call is automatically made when your app calls the AmazonAdView loadAd method.

The `registerApp` method fires off a background request that sends your applicationId and hashed deviceIDs to the Amazon Ad Network. To verify that the `registerApp` call succeeded you should review the Console log. If the call fails, that will be reported as an error. To view additional debug messages in your log you should call the `[AdRegistration sharedRegistration] setLogging:YES];` before calling `registerApp`. For example:

```
[[AmazonAdRegistration sharedRegistration] setApplicationId:@"0123456789"];
[[AmazonAdRegistration sharedRegistration] setLogging:YES]];
[[AmazonAdRegistration sharedRegistration] registerApp];
```

Should my app continue to call register?

Yes, we recommend that you call `registerApp` on every app start. The `registerApp` call checks to see if your device has already been registered. If so, it then checks if the timestamp is recent i.e. less than 24 hours old. If the timestamp is recent, then the registerApp call does nothing. If the timestamp is older than 24 hours, the `registerApp` method makes a couple of network calls in the background to refresh the device information. These calls allow us to create app segments for future download advertising campaigns. For example you could target devices that don't have your app installed. Or when releasing a sequel you could target devices that both have the prequel app installed and have made a register/refresh call in the last 30 days.

If your app doesn't make these refresh calls then we won't be able to create the app segments for your download advertising campaigns.

Appendix C – Using Amazon Mobile Ads API with Other SDKs

The Amazon Mobile Ads API can be used in conjunction with other Ad SDKs. Amazon recommends providing Amazon Mobile Ad Network with the first look. You should implement the optional callbacks for success and failure from the AmazonAdViewDelegate. If no ads were available from Amazon Mobile Ad Network, then the `adViewDidFailToLoad` callback function can be used to call another Ad SDK.

Below is a simple example of a program that first tries to load an ad from the Amazon Ad Network. On failure it makes a second ad request via the Other SDK:

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

    // Load an ad

    // Create an AmazonAdView instance with size 320x50.
    self.amazonAdView = [[[AmazonAdView alloc] initWithAdSize:AmazonAdSize_320x50]
autorelease];

    // Set the adOptions.
    AmazonAdOptions *options = [AmazonAdOptions options];
    adOptions.isTestRequest = YES;

    // Register the ViewController with the delegate to receive callbacks.
    amazonAdView.delegate = self;

    // Call loadAd
    [amazonAdView loadAd:options];

    // Ad the AmazonAdView to your viewController's view.
}

#pragma mark AmazonAdViewDelegate

- (UIViewController *)viewControllerForPresentingModalView
{
    return self;
}

- (void) adViewDidLoad:(AmazonAdView *)view
{
    NSLog(@"Ad loaded");
    if (!self.hasAmazonAdView) {
            [self.otherSDKAdView removeFromSuperview];
    }
    [self.view addSubView:view];
    self.hasAmazonAdView = YES;
}

- (void)adViewDidFailToLoad:(AmazonAdView *)view withError:(AmazonAdError *)error
{
    NSLog(@"Ad Failed to load. Error code %d: %@", error.errorCode, error.errorDescription);
    if (self.hasAmazonAdView) {
            [self.amazonAdView removeFromSuperView];
    }
    // Make an adRequest for the view of the other SDK and add it to this view.
    [self.view addSubView:otherSDKAdView];
}
```