Flurry Advertising

iOS SDK Instructions

SDK version 4.3.2
Updated: 02/06/2014

Welcome to Flurry Advertising!

This README contains:

---

1. Introduction

Flurry AppSpot is a flexible ad serving solution that allows you to easily manage the complete monetization of your mobile applications. Through integration with Flurry Analytics and various mobile ad networks including Flurry AppCircle, publishers can easily generate the maximum value from ad inventory.

With Flurry Ads you will be able to:

1. Define your inventory
2. Traffic ad campaigns
3. Track & optimize your performance

The Flurry Ads SDK is modular and contains only the functionality related to serving advertisements. It is designed to be as easy as possible with a basic setup completed in under 10 minutes.

For ad space setup and more information on Flurry AppSpot, please visit
http://support.flurry.com/index.php?title=Guides/s/Publishers
Please note, it is **required** that you create ad spaces before retrieving ads. Adspaces can be created on the Flurry Developer Portal or in the application code. If Adspaces are created in the code, they will appear in the dev portal designated as “Determined by SDK” in the Adspace setup page.

These instructions assume that you have already integrated Flurry Analytics into your application. If you

have not done so, please refer to **Analytics-README** to get started.

---

2. Basic 10 Minute Integration

Follow these steps to quickly integrate Flurry Ads into your app:

1. **The Ad Support framework is required. Flurry Ads will throw a linking error without the framework, and ads will not display.**
2. In the finder, drag FlurryAds/ into project's file folder.
3. Now add it to your project: Project > Add to project > FlurryAds - Choose 'Recursively create groups for any added folders'
4. In your source code, import FlurryAds and initialize FlurryAds with the initialize call sometime after `startSession`.

The parameter is as follows:

- UIViewController rvc - The root view controller of your application window

The final integration will look like:

```
#import "Flurry.h"
#import "FlurryAds.h"
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

      [Flurry startSession:@"YOUR_API_KEY"];
      [FlurryAds initialize:window.rootViewController]; // pointer to your
rootViewController
      //your code
}

/**
       *    You can connect Ads in any existing placement in your app, but for
testing purposes we present integration within your ViewController
       */
#import "FlurryAdDelegate.h"

- (void)viewWillAppear:(BOOL)animated {
      [super viewWillAppear:animated];

      /**
       *    We will show banner and interstitial integrations here.
       *
       */
      // Register yourself as a delegate for ad callbacks
      [FlurryAds setAdDelegate:self];
```

```
        // 1. Fetch and display banner ads
        [FlurryAds fetchAndDisplayAdForSpace:@”BANNER_MAIN_VC” view:self.view
        size:BANNER_BOTTOM];

        // 2. Fetch fullscreen ads for later display
        [FlurryAds fetchAdForSpace:@”INTERSTITIAL_MAIN_VC” frame:self.view.frame
        size:FULLSCREEN];

}

-(void) viewWillDisappear:(BOOL)animated {

        [super viewWillDisappear:animated];

        // Remove Banner Ads and reset delegate
        [FlurryAds removeAdFromSpace:@”BANNER_MAIN_VC”];

        [FlurryAds setAdDelegate:nil];
}

// Most often there is a point in your app to invoke a takeover (e.g. - button
is pressed, level is completed, etc). Here we will be mocking this existence
of a button method that shows full screen ads
-(IBAction) showFullScreenAdClickedButton:(id)sender {

// Check if ad is ready. If so, display the ad
if ([FlurryAds adReadyForSpace: :@"INTERSTITAL_MAIN_VC"]) {
        [FlurryAds displayAdForSpace: :@"INTERSTITAL_MAIN_VC" onView:self.view];
        } else {
        // fetch an ad
        [FlurryAds fetchAdForSpace:@"INTERSTITAL_MAIN_VC" frame:self.view.frame
size:FULLSCREEN];
        }
}

/*
 *  It is recommended to pause app activities when an interstitial is shown.
 *  Listen to should display delegate.
 */
- (BOOL) spaceShouldDisplay:(NSString*)adSpace interstitial:(BOOL)
    interstitial {
   if (interstitial) {
     // Pause app state here
   }
   // Continue ad display
    return YES;
}
```

```
/*
 *  Resume app state when the interstitial is dismissed.
 */
- (void)spaceDidDismiss:(NSString *)adSpace interstitial:(BOOL)interstitial {
   if (interstitial) {
     // Resume app state here
   }
}
```

3. Requesting Ads

Call `[FlurryAds fetchAndDisplayAdForSpace:view:size]` after the session starts. Note: please see **Analytics-README** for details on [Flurry startSession:]

```
+ (void)fetchAndDisplayAdForSpace:(NSString*)space view:(UIView*)viewContainer
size:(FlurryAdSize)size;
```

The parameters are as follows:

- **NSString space** - Name of the adSpace that you defined on the Flurry website.
- **UIView viewContainer** - UIView that you want the ad to reside in.
- **FlurryAdSize size** - The default size for your adSpace. This is overwritten by any value set on the server.

Note that the same ad call is used for all ad formats and sizes including Banner, Takeover and Custom ad sizes. In order to specify the format and size, you should log into the Flurry Developer Portal at dev.flurry.com and select the Publishers tab to specify the configuration for the adSpace name you have chosen.

---

4. Additional Ad Controls (Optional)

Asynchronously fetching an ad

Flurry provides a method to fetch and store an ad asynchronously before it is displayed. This enables you to pre-load ads before they are actually displayed. Use the following call to fetch an ad

```
+ (void)fetchAdForSpace:(NSString*)space frame:(CGRect)frame
size:(FlurryAdSize)size;
```

The parameters are as follows:

- **NSString space** - Name of the adSpace that you defined on the Flurry website.
- **CGRect frame** - The frame of the UIView that you want the ad to reside in.
- **FlurryAdSize size** - The default size for your adSpace. This is overwritten by any value set on the server.

Once you have made the request for an ad to be fetched, there are two ways of proceeding to display the ad. The first is to check if the ad is ready at various times, and then display the ad once it is ready (see calls for `adReadyForSpace` and `displayAdForSpace` below). The other way is to implement the `<FlurryAdDelegate>` which will be notified with a call to `spaceDidReceiveAd` when the ad is ready. You can then call `displayAdForSpace` at your convenience. For details on `FlurryAdDelegate`, see section 7: Implementing Ad Delegate.

Checking if an ad is ready

After an ad is fetched, you can explicitly check if the ad is ready to be displayed.

```
+ (BOOL)adReadyForSpace:(NSString*)space
```

This call will check if an ad is ready. If an ad is ready to be displayed then this will return true, or false if the ad is not ready.

- **NSString space** - Name of the adSpace that you defined on the Flurry website

Displaying the fetched ad

Once the ad is fetched, it can be displayed with the following call. Make sure to use `adReadyForSpace` above to ensure that an ad can be displayed.

```
+ (void)displayAdForSpace:(NSString*)space view:(UIView*)viewContainer size
```

The parameters are as follows:

- **NSString space** - Name of the adSpace that you defined on the Flurry website.
- **UIView viewContainer** - UIView that you want the ad to reside in.

Alternatively, if the ad needs to be displayed modally on a view, it can be displayed with the following call, `+(void)displayAdForSpace:(NSString*)space modallyForViewController:(UIView*)viewContainer size`

this routine will show the ad as a fullscreen interstitial and needs the `adReadyForSpace` check as in the previous display routine to ensure that an ad can be displayed:

The parameters are as follows:

- **NSString space** - Name of the adSpace that you defined on the Flurry website.
- **UIView viewContainer** - The viewController to show the fullscreen ad modally.

Removing an ad

Flurry manages the lifecycle of the ads it displays, however, you can exercise finer control over display by choosing when to add and remove the ads from your app (e.g. - in `viewWillAppear` and `viewDidDissapear`). To remove an ad just call

```
+ (void)removeAdFromSpace:(NSString*)space;
```

The parameter is as follows:

- **NSString space** - Name of the adSpace that you defined on the Flurry website

---

5. Enabling Ad Network Mediation (Optional)

Once your Ad Spaces are configured, you have the option of selecting 3rd party ad networks to serve ads into your Ad Spaces. You can change which ad networks serve ads at any time on the Flurry website, but in order to enable them you need to add the ad network SDKs into your application and configure them. The following Ad Networks are currently supported:

- iAd
- Admob
- Millennial
- InMobi
- Greystripe
- Mobclix (*Mediation with Mobclix is supported through their SDK 3.2.0; newer SDKs are not be supported.*

Mediation with Jumptap has been deprecated.

To implement an Ad Network you must perform the following steps:

1. Include the Ad Network iOS SDK with your app and add it to the project. Follow the instructions from the Ad network on how to complete this step.
2. Implement the appropriate delegate methods in FlurryAdDelegate

Here is the example of implementing the Admob SDK into FlurryAds:

```
@interface MyDelegateClass : NSObject <FlurryAdDelegate> {

 // definitions
}

// MyDelegateClass.m

@implementation MyDelegateClass

      - (NSString *) appSpotAdMobPublisherID
      {
            // Your AdMob Publisher ID, found from here:
                  http://www.admob.com/my_sites/ (click manage settings)
            return @"<your id>";
      }

      - (BOOL)appSpotTestMode
      {
            // If testing, return yes
            return YES;
      }

@end
```

6. Request Configuration (Optional)

There are a number of configuration parameters that you can use to modify the behavior of your ad spaces.

Option 1. Enable Test Ads

Add a call to receive test ads from the flurry server to ensure proper implementation. Test ads do not generate revenue and therefore MUST be disabled before submitting to the AppStore:

```
[FlurryAds enableTestAds:(BOOL)enable];
```

Option 2. Set Location

Add a call to set the location (lat,long) that you want associated with the ad request, to be used with geographical targeting. Passing lat,long data enables better monetization of your ad inventory.

```
[Flurry setLatitude:(double)latitude longitude:(double)longitude
horizontalAccuracy:(float)horizontalAccuracy
verticalAccuracy:(float)verticalAccuracy];
```

Option 3. User Cookies

Add a call to identify any user specific information you want associated with the ad request:

```
[FlurryAds setUserCookies:(NSDictionary *) userCookies];
```

To remove any user cookies call:

```
[FlurryAds clearUserCookies];
```

Option 4. Keyword Targeting

Add a call to specify keywords to be used when targeting ads:

```
[FlurryAds setKeywordsForTargeting:(NSDictionary *) keywords];
```

To clear the set keywords call:

```
[FlurryAds clearKeywords];
```

---

7. Implementing Ad Delegate (Optional)

To be notified of certain events during the full lifecycle of the Ad, you can implement the FlurryAdDelegate and then call the [FlurryAds setDelegate:] method to attach your delegate to the FlurryAds. When you implement the FlurryAdDelegate you will implement the following callback methods:

- (void) spaceDidReceiveAd:(NSString*)adSpace;
    - This method is called when an ad has been received and is available for display on the ad space specified by adSpace.
- (void) spaceDidFailToReceiveAd:(NSString*) adSpace error:(NSError *)error;
    - This method informs the app that an ad has failed to be received for the given adSpace.
- (BOOL) spaceShouldDisplay:(NSString*)adSpace interstitial:(BOOL)interstitial;

    - This method informs the app that an ad is about to be displayed on the adSpace. The parameter interstitial will be YES/NO if the space to display will be an interstitial. You can decide at this point not to show this ad by simply returning NO. Returning YES will allow the ad to be shown.
- (void) spaceDidFailToRender:(NSString *)space error:(NSError *)error;
    - This method informs the user an ad was retrieved, however, was unsuccessful in displaying to the user (could be lost network connectivity for example).
- (void) spaceWillDismiss:(NSString *)adSpace;
    - This method will be called when the user dismisses the current Ad for the provided Ad Space name.
- (void)spaceDidDismiss:(NSString *)adSpace interstitial:(BOOL)interstitial;
    - This method informs the app that an ad has closed. You can use this to resume app states.
- (void) spaceWillLeaveApplication:(NSString *)adSpace
    - This method will be called when the user is leaving the application after following events associated with the current Ad in the provided Ad Space name.

Example usage:

```
@interface MyDelegateClass : NSObject <FlurryAdDelegate> {

// definitions
}

// MyDelegateClass.m

@implementation MyDelegateClass

-(void) init
{
      [super init];
      [FlurryAds setAdDelegate:self];          // Set the delegate
}

// Other code

- (void) spaceDidReceiveAd:(NSString*)adSpace
{
      // Show the ad if desired
      [FlurryAds displayAdForSpace:[self myAdSpace] onView:[self view]];
}

- (void) spaceDidFailToReceiveAd:(NSString*)adSpace error:(NSError *)error
{
      // Handle failure to receive ad
}

- (BOOL) spaceShouldDisplay:(NSString*)adSpace interstitial:(BOOL)interstitial
{
```

```
      // Decide if the Ad should be displayed
      return true;
}

- (void) spaceDidFailToRender:(NSString *)space error:(NSError *)error
{
      // Handle a failure to render the ad
}

- (void)spaceWillDismiss:(NSString *)adSpace
{
      // Handle the user dismissing the ad
}

- (void)spaceDidDismiss:(NSString *)adSpace
{
      // Handle the closing of the ad
}

- (void)spaceWillLeaveApplication:(NSString *)adSpace
{
      // Handle the user leaving the application
}

@end
```
