AppCircle

iPhone SDK Instructions

SDK version 2.8.4
Updated: 02/04/2011

Welcome to Flurry AppCircle!

This README contains:

---

1. Introduction

The Flurry SDK contains all the existing Flurry Analyitcs functionality as well as the new AppCircle functionality. It is designed to be as easy as possible with a basic setup complete in under 5 minutes. This SDK can be used with universal (iPhone/iPad), iPhone only, and iPad only apps. The SDK is compiled with Base SDK 4.2.

The archive should contain these files for use with Flurry AppCircle:

- **AppCircle-README.txt** : This text file contains instructions to use Flurry AppCircle.
- **FlurryLib/AppCircle/FlurryAdDelegate.h** : The header file containing methods for Flurry AppCircle delegate.
- **FlurryLib/AppCircle/FlurryAPI+AppCircle.h** : The header file containing methods for Flurry AppCircle.
- **FlurryLib/AppCircle/FlurryOffer.h** : The header file containing methods for FlurryOffer.

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. AppCircle Integration

The following instruction is to integrate AppCircle inside an app that uses Flurry Analytics. Please see the Flurry Analytics README for more information on how to integrate Flurry Analytics.

To integrate Flurry AppCircle into your iPhone application, you must first enable AppCircle before you start the session:

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
[FlurryAPI setAppCircleEnabled:YES];
[FlurryAPI startSession:@"YOUR_API_KEY"];
//your code
}
```

When you start the session, AppCircle related data will be downloaded to the device. You may need to wait a couple seconds on initial application launch before all the AppCircle data is downloaded. Once AppCircle data is downloaded, it is cached for use on subsequent application launches.

There are 4 types of AppCircle ad integrations:

- Banners
- Full screen canvas page
- Full screen takeover page
- Custom ad formats

2.1 Banner Integration

The most basic integration of AppCircle involves placing a banner on an UIView. Clicking on the banner will display a full screen canvas that gives more information about the banner. A user can cancel out of the canvas and go back to the previous application view.

The following code will create and place the banner on the parent view. This is optimized for both iPhone and iPad resolution:

```
UIView *banner = [FlurryAPI getHook:@"USE_AN_UNIQUE_HOOK_NAME"
                              xLoc:0
                              yLoc:0
                              view:self.view];
```

Each banner should use an unique hook name. The hook name allows Flurry to track and configure the banners individually.

If you require more control on the banner itself, use the longer **`getHook`** method with optional parameters:

- **`attachToView`** controls whether the banner is automatically placed on the parent view. The default setting is YES.
- **`orientation`** controls the length of the banner. The values are @"portrait" and @"landscape". @"portrait" sets the banner dimension at 728x90 in iPad and 320x48 in iPhone. @"landscape" sets the banner dimension at 960x90 in iPad and 480x48 in iPhone. The default setting is @"portrait".
- **`canvasOrientation`** controls the canvas orientation. The values are @"portrait", @"portraitUpsideDown", @"landscapeRight", and @"landscapeLeft". The default setting is @"portrait".
- **`autoRefresh`** controls whether the banner will automatically update itself with different ads cached on the device. The current refresh interval is 20 seconds. The default setting is NO.
- **`canvasAnimated`** controls if the canvas is animated when opening and closing. The default is YES.
- **`rewardMessage`** controls the reward message for apps participating in AppCircle Rewards. The default setting is nil.
- **`userCookies`** controls the reward amount, currency, and other incentive information for apps participating in AppCircle Rewards. The default setting is nil.

After requesting the banner with **`getHook`** method, subsequent calls with the same parent and hook will result in the same banner instance being returned. Only one banner will be created per hook and parent view. We recommend updating existing banners with new ads instead of creating new banners.

To update an existing banner with new ad:

```
[FlurryAPI updateHook:banner];
```

To remove an existing banner from its parent view and hook in order to create additional banners on the same hook and parent view:

```
[FlurryAPI removeHook:banner];
[banner removeFromSuperview];
```

2.2 Catalog Integration

Each canvas should use an unique hook name. The hook name allows Flurry to track and configure the canvases individually.

If you want to display a full screen canvas page without displaying any banner, call this method to bring up a canvas page anytime. This is optimized for iPhone and iPad resolution.

```
[FlurryAPI openCatalog:@"USE_AN_UNIQUE_HOOK_NAME"
           canvasOrientation:@"portrait"
           canvasAnimated:YES];
```

More information about the parameters:

- **openCatalog** uses an unique hook name just like the banners.
- **canvasOrientation** controls the canvas orientation. The values are @"portrait", @"portraitUpsideDown", @"landscapeRight", and @"landscapeLeft". The default setting is @"portrait".
- **canvasAnimated** controls if the canvas is animated when opening and closing. The default is YES.

2.3 Takeover Integration

Each takeover should use an unique hook name. The hook name allows Flurry to track and configure the takeovers individually.

If you want to display a full screen takeover page, call this method to bring up a takeover page anytime. This is optimized for iPhone and iPad resolution.

```
[FlurryAPI openTakeover:@"USE_AN_UNIQUE_HOOK_NAME"
           orientation:@"portrait"
           rewardImage:nil
           rewardMessage:nil
           userCookies:nil];
```

More information about the parameters:

- **openTakeover** uses an unique hook name just like the banners.
- **orientation** controls the takeover orientation. The values are @"portrait", @"portraitUpsideDown", @"landscapeRight", and @"landscapeLeft". The default setting is @"portrait".
- **rewardImage** is a transparency enabled png image with dimension of 137x108 used for the user reward icon image. The default setting is nil.
- **rewardMessage** controls the reward message for apps participating in AppCircle Rewards. The default setting is nil.
- **userCookies** controls the reward amount, currency, and other incentive information for apps participating in AppCircle Rewards. The default setting is nil.

2.4 Custom Ad Formats Integration

You can create custom ad formats using the cached Flurry offers. The offer data returns the app name, the app icon, the app price, some app description, and the referral URL. Flurry will not render any ads when the offer is called. It is up to the developer to use the offer data to render their own ads.

Each offer corresponding to a unique ad instance should use an unique hook name. The hook name allows Flurry to track and configure the offers individually.

To access an offer:

```
FlurryOffer *flurryOffer = [[FlurryOffer alloc] init];
BOOL validOffer = [FlurryAPI getOffer:@"USE_AN_UNIQUE_HOOK_NAME"
                           withFlurryOfferContainer:flurryOffer];
if (validOffer) {
      NSString *appName = flurryOffer.appName;
      UIImage *appIcon = flurryOffer.appIcon;
      NSNumber *appPrice = flurryOffer.appPrice;
      NSString *appDescription = flurryOffer.appDescription;
      NSString *referralUrl = flurryOffer.referralUrl;
```

```
      // render ad format here
    ...
}
```

To figure out how many distinct offers exist for a hook:

```
int offerCount = [FlurryAPI getOfferCount:@"USE_AN_UNIQUE_HOOK_NAME"];
```

---

3. AppCircle Rewards

Normally, when a user downloads and launches an AppCircle promoted app, the user is not rewarded for doing so. For apps that reward users with virtual currency or other incentives, the app can reward users when they download and launch an AppCircle promoted app. Once Flurry receives the data from the promoted app, Flurry will make a callback to the developer's server with the user's UDID, reward currency, reward amount, and any other data that should be returned in the callback. The developer can use the Flurry callback to reward the user accordingly.

To integrate AppCircle Reward, the developer will need to do the following:

- Setup a server to receive the reward callback from Flurry
- Contact Flurry to setup the reward callback URL in the Flurry system
- Use offers, banners, or takeovers to render ads with custom reward messages
- Set user cookies with reward information that should be returned in the callback
- Reward user when Flurry callback is received

The app can notify the users of the rewards for downloading AppCircle promotions in a couple of ways:

Using offers, you can render your own custom ad type. You can change the reward currency and reward amount in the callback by using user cookies. Flurry will send you the correct reward currency and reward amount in the callback. An example:

```
// reward user with 10 gold for downloading and launching the following offer
// your callback will contain rewardCurrency=gold&rewardAmount=10
NSDictionary *dictionary =
[NSDictionary dictionaryWithObjectsAndKeys:@"gold",
                                           @"rewardCurrency",
                                           @"10",
                                           @"rewardAmount",
                                           nil];
BOOL validOffer = [FlurryAPI getOffer:@"USE_AN_UNIQUE_HOOK_NAME"
                           withFlurryOfferContainer:flurryOffer
                           userCookies:dictionary];
if (validOffer) {
      // render ad format here with reward message
    ...
}
```

In the banner, the rewardMessage parameter can be set to replace the default message with a reward message. An example:

```
[FlurryAPI getHook:@"USE_AN_UNIQUE_HOOK_NAME"
           xLoc:0
           yLoc:200
           view:self.view
           attachToView:YES
           orientation:@"portrait"
           canvasOrientation:@"portrait"
           autoRefresh:YES
           canvasAnimated:YES
           rewardMessage:@"DOWNLOAD and LAUNCH this app to earn 10 gold"
           userCookies:dictionary];
```

In the takeover, the rewardMessage parameter can be set to replace the default message with a reward message. You can also set a rewardImage to show a reward icon. An example:

```
UIImage *rewardImage = ....
[FlurryAPI openTakeover:@"USE_AN_UNIQUE_HOOK_NAME"
           orientation:@"portrait"
           rewardImage:rewardImage
           rewardMessage:@"DOWNLOAD and LAUNCH this app to earn 10 gold"
           userCookies:dictionary];
```

---

4. Optional / Advanced Features

You can pass in a delegate to FlurryAPI in order to receive callbacks from FlurryAPI. This is an optional feature that allows more control over the how the app interact with the canvas and takeover pages.

The following example shows how it can be used with the app delegate. By setting up this way, the app delegate will received the callback from FlurryAPI. You can pass in any object as the delegate.

In AppDelegate.h:

```
// add FlurryAdDelegate.h file to the project if you are using FlurryAdDelegate
#import "FlurryAdDelegate.h"

// add FlurryAdDelegate as a protocol
@interface AppDelegate : NSObject <UIApplicationDelegate, FlurryAdDelegate> {
}
```

In AppDelegate.m:

```
// implement to do something when the data is available
// currently just output debug message
- (void)dataAvailable {
    NSLog(@"Flurry data is available");
}

// implement to do something when the data is unavailable
// currently just output debug message
- (void)dataUnavailable {
    NSLog(@"Flurry data is unavailable");
}

// implement to do something when the canvas will open
// currently just output debug message
- (void)canvasWillDisplay:(NSString *)hook {
    NSLog(@"Flurry canvas will display:%@", hook);
}

// implement to do something when the canvas will close
// currently just output debug message
- (void)canvasWillClose {
    NSLog(@"Flurry canvas will close");
}

// implement to do something when the takeover will open
// currently just output debug message
- (void)takeoverWillDisplay:(NSString *)hook {
    NSLog(@"Flurry takeover will display:%@", hook);
}

// implement to do something when the takeover will close
```

```
// currently just output debug message
- (void)takeoverWillClose {
    NSLog(@"Flurry takeover will close");
}

// set FlurryAdDelegate before the session starts
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    [FlurryAPI setAppCircleDelegate:self];
      [FlurryAPI startSession:@"YOUR_API_KEY"];
}
```

---

5. Recommendations

For universal apps that are optimized for both iPad and iPhone, please create an iPhone platform project in the Flurry system. For non-universal apps, please use the correct platform that your app is optimized for.

See Flurry Analytics README for more information.

---

6. FAQ

How much data does the Agent send each session?

All data uploaded by the Flurry Agent is sent in a compact binary format. The total amount of data can vary but in most cases it is around 2Kb per session.

All data downloaded by the Flurry Agent is also sent in a compact binary format. Since AppCircle involves the displaying of images the amount of data downloaded is significantly larger than the amount uploaded with the typical session involving about 10Kb of download.

How many recommendations can I display?

By default, 7 recommendations are cached locally on the device for fast loading. These recommendations will rotate through at random whenever a recommendation is displayed.

If you need to display more than 7 recommendations in your application please contact support.

Why does a downloaded app continue to display as an ad?

There are a few possibilities on why an ad continue to display even after you downloaded the app. iOS pauses the app when the app closes so the same view is kept in memory on next launch. The banners on the same view is also kept the same for the next launch. If you want the banner to display new ads, you can update the banner or enable auto refresh.

Another possibility is that the app is using cached ads. On the app launch, new ads are fetched and cached. If ads are requested before new ads are not completely fetched, the previous set of cached ads are used to fulfill the ad requests. To clear the cache, remove and reinstall the app.

Please let us know if you have any questions. If you need any help, just email iphonesupport@flurry.com!

Cheers,
The Flurry Team
http://www.flurry.com

iphonesupport@flurry.com