AppCircle Reengagement

iOS SDK Instructions

SDK version 3.x
Updated: 02/06/2012

Welcome to Flurry AppCircle Reengagement!

This README contains:

---

1. Welcome

Thank you for your interest in integration with Flurry Reengagement. Based on Flurry analytics, we know the mobile audience is ideal for Reengagement advertisements. We project Flurry Reengagement will provide a sustainable alternative stream of revenue for publishers and through your integration with Reengagement your company will be at the forefront of this emerging market.

2. Introduction

The FlurryReengage SDK is modular and contains only Reengagement functionality. The FlurryReengage library is dependent on the Analytics library. If you have not read instructions for Analytics, please refer to **Analytics-README** to get started.

Reengagement 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 5.0, but supports devices running iOS 3.0 and above.

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

- **Reengagement-README.pdf** : This pdf contains instructions on how to use Flurry Reengagement.
- **FlurryReengage/FlurryAdDelegate.h** : The header file containing methods for Flurry Reengagement delegate.
- **FlurryReengage/FlurryReengage.h** : The header file containing methods for Flurry Reengagement.
- **FlurryReengage/libFlurryReengage.a** : The required library containing Flurry Reengagement ad serving code.

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.

---

3. AppCircle Reengagement Integration

The following instruction is to integrate AppCircle Reengagement 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 Reengagement into your iOS application, use the following steps:

1. In the finder, drag FlurryReengage/ into project's file folder.

2. Now add it to your project:
File > Add Files to “Your Project” … > FlurryReengage
- Destination: select Copy items into destination group’s folder (if needed)
- Folders: Choose 'Create groups for any added folders'
- Add to targets: select all targets that the lib will be used for

3. In your source code, import FlurryReengage and enable it by calling setReengagementEnabled before calling Analytics' startSession method:

4. Set FlurryAdDelegate if required (please see section 4.2.4 below for more on FlurryAdDelegate)

```
#import “FlurryReengage.h”
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    [FlurryReengage setReengagementEnabled:YES];
    [FlurryReengage setReengagementDelegate:self]; //Can be set with any
object
    [FlurryAnalytics startSession:@"YOUR_API_KEY"];
    //your code
}
```

When you start the session, Reengagement related data will be downloaded to the device. A couple of seconds may elapse on initial application launch before all data is downloaded.

4. AppCircle Reengagement Integration Methods

AppCircle Reengagement is currently available as a Full Screen Takeover page and as Banners.

4.1. AppCircle Hooks

All AppCircle integrations should use a unique hook name to denote its placement in your app. For example, you may have the option of your users viewing Reengagements on the splash screen and one on a bonus screen. In this case you would have two hook names, such as “ReengageMENT_SPLASH_HOOK” and “ReengageMENT_BONUS_HOOK”. The hook name is a critical non-empty unique identifier that will be used for many purposes to include:

1. Will be provided in the callback delegates to allow you to resume proper flow of control (e.g. - the next step after the splash screen may differ from the next step after the bonus screen).
2. View the performance of each of the hook placements separately in the Flurry Dev Portal so you can determine which placements are most effective.
3. The hook name should be used to define the type of content you receive. For example, you may have a hook for Clips and Reengagement on the splash screen. In this case you might have the following two hook names “CLIPS_SPLASH_HOOK” and “ReengageMENT_SPLASH_HOOK”. It is good to prefix hook names by product so they are easily distinguishable in your app and Publisher analytics reporting.

4.2 Reengagement Takeover

A Reengagement takeover is a formatted, full screen display that will present your user with the option to re-engage an app to receive a reward (reward is optional). A Reengagement Takeover integration will include the following steps:

1. Verify a Re-engagment ad is available for viewing.
2. Call method to show a Reengagement takeover.
3. Implement FlurryAdDelegate methods to resume flow of control (if applicable).

4.2.1 Verify Reengagement is Available

Before presenting the user with the option to watch a Reengagement ad, you should **always** first check the availability of a Reengagement Ad for a specific hook. This is important as the ad meta data may still be downloading from the server. Checking if the ad is available in your app workflow will allow you to present the user with other options (e.g. - an app recommendation or just proceeding to the next step in your app).

To check if a Reengagement Ad is available call

```
[FlurryReengage reAdIsAvailable:@"HOOK_NAME_TO_DEFINE_REENGAGEMENT_PLACEMENT"];
```

More information about the parameter:

- **`reAdIsAvailable`** uses an unique hook name that will define the Reengagement ad placement in the app (e.g. - @"ReengageMENT_SPLASH_SCREEN").

NOTE: reAdIsAvailable will always return NO on a simulator. Re-engagement requires that the device (in this case the simulator) has the advertiser apps and has not returned to them in some timeframe. This does not apply for the simulator. Please test reengagement on a device.

4.2.2 Show Reengagement Takeover

The takeover view is optimized for iPhone and iPad resolution. To display the full screen takeover page call

```
[FlurryReengage openTakeover:@"HOOK_NAME_TO_DEFINE_REENGAGEMENT_PLACEMENT"
           orientation:@"portrait"
           rewardImage:nil
           rewardQuantity:nil
           rewardUnits:nil
           userCookies:nil];
```

More information about the parameters:

- **`openTakeover`** uses an unique hook name that will define the ad placement in the app (e.g. - @"ReengageMENT_SPLASH_SCREEN").
- **`orientation`** controls the takeover orientation. The values are @"portrait", @"portraitUpsideDown", @"landscapeRight", and @"landscapeLeft". You may pass in `nil` to let Flurry position the Takeover based on the current orientation of the device.
- **`rewardImage`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to nil.
- **`rewardQuantity`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to nil.
- **`rewardUnits`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to nil.
- **`userCookies`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to nil.

4.2.3 Show Reengagement Banner

Reengagement enables you to place a banner on an UIView. Clicking on the banner will display an expanded banner canvas that gives more information about the banner. Similar to Takeovers the expanded canvas will present your user with the option to re-engage an app to receive a reward (reward is optional). 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.

```
[FlurryReengage getHook:@"HOOK_NAME_TO_DEFINE_REENGAGEMENT_PLACEMENT"
                  xLoc:0
                  yLoc:0
         bannerIsAtTop:YES
                  view:parentView]
```

bannerIsAtTop boolean must correctly reflect the position of the collapsed banner within the overall screen space and not just within the parent view. It affects the banner expand behavior and expand/

collapse arrow images for the banner.
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 320x50 in iPhone. @"landscape" sets the banner dimension at 1024x90 in iPad and 480x50 in iPhone. You may pass in nil to let Flurry control the length of the banner based on the current device orientation. The default setting is nil.
- **`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 YES.
- **`rewardQuantity`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to nil.
- **`rewardUnits`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to nil.
- **`userCookies`** Reengagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time. Set this to 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:

```
[FlurryReengage 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:

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

Note: After all re-engagement ads have been displayed the banner will be hidden. Implement the bannerHidden delegate that's declared in FlurryAdDelegate. See the section below for more details on implementing delegates.

4.2.4 Implement FlurryAdDelegate Selectors

You can pass in a delegate to FlurryReengage prior to calling `startSession` in order to receive callbacks from FlurryReengage. This is an optional feature that will notify you when the takeovers close and you can use it to resume flow of control (if applicable).

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

In AppDelegate.h:

```
// Add FlurryAdDelegate.h file to the project
#import "FlurryAdDelegate.h"

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

In AppDelegate.m:

```
// Implement to do something when the takeover will open
// Currently just output log message
- (void) takeoverWillDisplay:(NSString *)hook {
    NSLog(@"Flurry takeover will display for hook %@", hook);
}

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

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

5. FAQ

Why am I getting duplicate symbol errors when trying to add both FlurryAppCircle and FLurryReengage?

This will occur if the -ObjC or -all_load flags are used in your build configuration. These 2 flags can not be used with Flurry's modular sdk. However, Apple does provide a -force_load flag that is both more efficient and does not cause issue when applied to other libraries your app uses (note: force_load should only be used on static libraries exhibiting this behavior. It should not be applied to the Flurry Libs). More details can be found in this link (http://developer.apple.com/library/mac/#qa/qa1490/_index.html).

Why are re-engagement ads never available when I test on the simulator?

[FlurryReengage reAdIsAvailable:@"hook"] will always return NO on a simulator. Re-engagement requires that the device (in this case the simulator) has the advertiser apps and has not returned to them in some timeframe. This does not apply for the simulator. Please test re-engagement on a device.

Can I reward users as the publisher?

Re-engagement Incentives are being provided by the advertiser. Publisher incentives are not supported at this time.

Please let us know if you have any questions. If you need any help, just email your support contact!

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