AppCircle Clips

iOS SDK Instructions

SDK version 3.0
Updated: 10/21/2011

Welcome to Flurry AppCircle Clips!

This README contains:

---

1. Welcome

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

2. Introduction

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

Clips 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 4.3, but supports devices running iOS 3.0 and above.

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

- **Clips-README.pdf** : This pdf contains instructions to use Flurry Clips.
- **FlurryClips/FlurryAdDelegate.h** : The header file containing methods for Flurry Clips delegate.
- **FlurryClips/FlurryClips.h** : The header file containing methods for Flurry Clips.
- **FlurryClips/FlurryVideoOffer.h** : The header file containing the structure of the Flurry Video container (for use with peekVideoOffer method).
- **FlurryClips/libFlurryClips.a** : The required library containing Flurry's video 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 Clips Integration

The following instruction is to integrate AppCircle Clips 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 Clips into your iOS application, you must invoke these calls before `startSession` is called:

1. Include Reference to MediaPlayer framework in your project

2. In the finder, drag FlurryClips/ into project's file folder

3. Now add it to your project:
File > Add Files to “Your Project” … > FlurryAnalytics
- 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

4. In your source code, import FlurryClips and enable it before calling Analytics' startSession method:

5. Set FlurryAdDelegate

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

When you start the session, Clips related data will be downloaded to the device. A couple of seconds may elapse on initial application launch before all data is downloaded. Once Clips data is downloaded, it is cached for use on subsequent application launches.

4. AppCircle Clips Integration Methods

AppCircle Clips is currently available as a Full Screen takeover page and as a custom integration. The custom integration allows you to display the offer for watching a video, however, the player is controlled by the Flurry Lib.

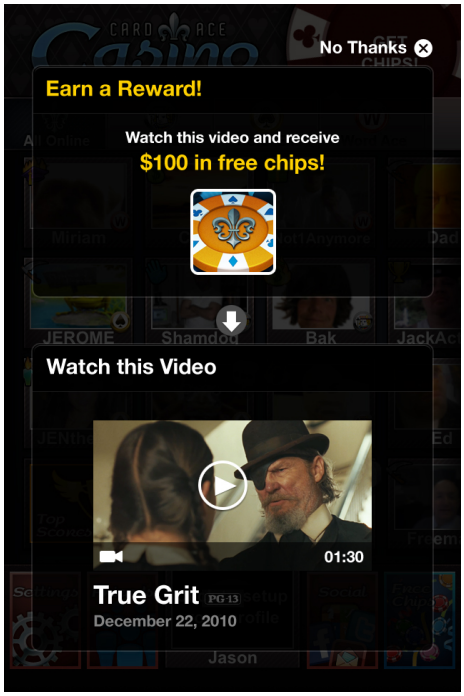
4.1. AppCircle Hooks

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 video on the splash screen and also as part of an in app purchase view. In this case you would have two hook names, such as “VIDEO_SPLASH_HOOK” and “VIDEO_IAP_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 in app purchase store view).
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 receiving video referrals as well as a hook for receiving app referrals on the splash screen. In this case you might have the following two hook names “VIDEO_SPLASH_HOOK” and “APP_SPLASH_HOOK”.

4.2 Clips Takeover

A Clips takeover is a formatted, full screen display that will present your user with the option of viewing a video to receive a reward (reward is optional). A sample display of the AppCircle Clips Takeover is seen below:



A Clips Takeover integration will include the following steps:

1. Verify a video is available for viewing.
2. Call method to show video takeover.
3. Implement FlurryAdDelegate methods to resume flow of control and properly reward users (if applicable).

4.2.1 Verify Video is Available

Before presenting the user with the option to watch a rewarded video, you should **always** first check the availability of a video for a specific hook. This is important as the video meta data may still be downloading from the server or the user may have reached the frequency cap for all videos currently available on their device. Checking if the video 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 video is available call

```
[FlurryClips videoAdIsAvailable:@"HOOK_NAME_TO_DEFINE_CLIPS_PLACEMENT"];
```

More information about the parameter:

- **`videoAdIsAvailable`** uses an unique hook name that will define the video's placement in the app (e.g. - @"VIDEO_SPLASH_SCREEN".

4.2.2 Show Video Takeover

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

```
[FlurryClips openVideoTakeover:@"HOOK_NAME_TO_DEFINE_CLIPS_PLACEMENT"
           orientation:@"portrait"
           rewardImage:nil
           rewardMessage:nil
           userCookies:nil
           autoPlay:NO];
```

More information about the parameters:

- **`openVideoTakeover`** uses an unique hook name that will define the video's placement in the app (e.g. - @"VIDEO_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 Catalog based on the current orientation of the device.
- **`rewardImage`** is a transparency enabled png image with dimension of 120x120 used for the user reward icon image. The default setting is nil.

- **`rewardMessage`** controls the reward message for apps participating in AppCircle Rewards. The takeover provides the text “Watch this video and receive” for all reward messages. The rewardMessage parameter should therefore only provide the reward amount (e.g. - @”100 Free Coins”). 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.
- **`autoPlay`** defines whether the video will play automatically without presenting an offer to watch the video. This is useful for pre-roll of video or if your app will handle the presentation of the offer to watch the video. The default setting is NO.

4.2.3 Implement FlurryAdDelegate Selectors

You must set your ad delegate prior to calling `startSession` in order to receive callbacks from FlurryClips. This will notify you when the video flow is complete so that you may reward the user (if applicable) and take the user to the next phase of your app.

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 FlurryClips. 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 resume control when the video viewing was not completed
// Currently just output log message
- (void)videoDidNotFinish:(NSString *)hook {
    NSLog(@"Flurry video did not finish for hook %@", hook);
}

// Implement to reward user (if reward was not passed as nil) and resume
// flow of control
// Currently just output log message
- (void)videoDidFinish:(NSString *)hook withUserCookies:(NSDictionary *)cookies {
    NSLog(@"Flurry video completed for hook %@", hook);
}
```

5. Clips Rewards

Clips can be set to pass reward information containing virtual currency or other incentives for a completed video viewing.

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

- Implement the `videoDidFinish:withUserCookies:` delegate callback
- Set `userCookies` with reward information in `openVideoTakeover:orientation:rewardImage:rewardMessage:userCookies:autoPlay:`
- Reward user when Flurry invokes the `videoDidFinish:withUserCookies:` delegate. Use the `withUserCookies` parameter to retrieve the information you passed in `openVideoTakeover` to credit the user accordingly.

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 delegate callback. An example:

```
// Reward user with 10 gold for downloading and launching the following offer
// Video will be offered in In App Purchase store (provide hook such as
IAP_VIDEO_HOOK)
BOOL videoAvailable = [FlurryClips videoAdIsAvailable:@"IAP_VIDEO_HOOK"];
if (videoAvailable) {
```

```
      // Create user cookie with reward information
      NSDictionary *dictionary =
      [NSDictionary dictionaryWithObjectsAndKeys:@"gold",
                                          @"rewardCurrency",
                                          @"10",
                                          @"rewardAmount",
                                          nil];

      // render ad format here with reward message
      UIImage *rewardImage = ....
      [FlurryClips openVideoTakeover:@"IAP_VIDEO_HOOK"
           orientation:@"portrait"
           rewardImage:rewardImage
           rewardMessage:@"10 Free Gold"
           userCookies:dictionary
           autoPlay:NO];
    ...
}

// Follow instructions in section "4.2.3 Implement FlurryAdDelegate Selectors"
```

You can also render your own custom offer message that is integrated with your app and only show the video. To do this simply pull the assets for the next video preview:

```
FlurryVideoOffer *flurryVideoOffer = [[FlurryVideoOffer alloc] init];
BOOL validOffer = [FlurryClips peekVideoOffer:@"IAP_VIDEO_HOOK"
                        withFlurryVideoOfferContainer:flurryVideoOffer];
if (validOffer) {
                  // Pull assets for custom display (see FlurryVideoOffer.h for all
properties)
      // e.g. - flurryVideoOffer.videoName, flurryVideoOffer.videoLength

      // When user clicks on your offer, you MUST call the openVideoTakeover
selector with autoPlay:YES. The peekVideoOffer does not return a valid videoURL.
}
```

Note if you do not want to show the user a video that was peeked (e.g. - they previously declined to watch this video), you may get the next video in the queue by calling:

```
[FlurryClips rotateVideoOffer:@"IAP_VIDEO_HOOK"];
```

---

6. Recommendations

- If you have to contact a server to reward a user and it is temporarily unavailable, we recommend persisting the user reward information you receive in the `videoDidFinish:withUserCookies:` callback. This will allow you to guarantee you will reward users if their network connection prevents them from being rewarded immediately.

---

7. FAQ

I do not want to set up a local delegate for rewarding the user. Can you invoke a call to my server when the user completes the video view?

Yes, please contact your account manager to set up a callback to your server or if you do not have one, please

email appcircleclips@flurry.com.

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

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).

If the user does stops the video before completing will they be presented with the next video?

Yes, if there are additional videos for the user to watch the next time the user is presented with a video to view it will be the next one in the queue.

I want to set up a custom offer and play the video with my own controller. Can I do this using the peekVideoOffer selector?

No, the peekVideoOffer can only be used to display the offer. The videoUrl will always be nil when returned from peekVideoOffer. The FlurryLib must ensure a completed view in full in order to process a reward for the user.

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
appcircleclips@flurry.com