FLURRY

Flurry Analytics

tvOS SDK Instructions

SDK version 1.0.0
Updated: 02/17/2016

Welcome to Flurry Analytics!

This file contains:

1. Introduction

The Flurry tvOS Analytics Agent allows you to track the usage and behavior of your tvOS application on users' Apple TVs for viewing in the Flurry Analytics system. It is designed to be as easy as possible with a basic setup complete in under 5 minutes.

Please note that this SDK will only work with Xcode 7 or above.

Flurry Agent does not require CoreLocation framework and will not collect GPS location by default. Developers who use their own CLLocationManager can set GPS location information in the Flurry Agent (see Optional Features for more information).

2. Integration

The integration for native tvOS apps is identical to integration for iOS apps. For the Client-Server apps see the Client-Server Integration section.

1. In the finder, drag FlurrytvOS/ into projects file folder. *(NOTE: If you are upgrading the Flurry iOS SDK, be sure to remove any existing Flurry library folders from your project's file folder before proceeding.)*
2. Now add it to your project. File > Add Files to "Your Project" … > Flurry
   - 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. Add Security.framework to your app.

4. Add SystemConfiguration.framework to your app. This is required for Reachability to manage network operations efficiently
5. In your Application Delegate:
   - Import Flurry and inside “application:didFinishLaunchingWithOptions:”
   add [Flurry startSession:@”YOUR_API_KEY”];

Objective - C

```
#import "Flurry.h"

- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    [Flurry startSession:@"YOUR_API_KEY"];
    //your code
}
```

You’re done! That’s all you need to do to begin receiving basic metric data.

6. FlurryDelegate:

Flurry tvOS analytics can take a delegate to callback on session getting created. A new protocol, FlurryDelegate is added to Flurry Analytics. The delegate object for Flurry needs to implement protocol, FlurryDelegate and need to be set as delegate using the following API call:

```
[Flurry setDelegate:<object that implements FlurryDelegate>];
```

Once delegate is set and the delegate responds to selector `flurrySessionDidCreateWithInfo:`, delegate gets callback on session getting created every time. The info provided in the callback is a NSDictionary object which contains data, “apiKey” and “sessionId”.

To integrate with Swift applications:
- In your Swift application, add a new Objective-C .m file to your project.
- When prompted with creating a bridge header file approve the request.
- Remove the unused .m file.
- In the bridge header file import the Flurry.h header file.
- In your application’s AppDelegate.swift file, inside application didFinishLaunchingWithOptions add the Flurry startSession call.

Swift Code :

```
func application(application:UIApplication, didFinishLaunchingWithOptions
launchOptions: NSDictionary) -> Bool {
    Flurry.startSession("YOUR_API_KEY")
   //your code
   return true
}
```

Note On Advanced Features

Flurry strongly recommends that you read the Advanced Features section below. These features are optional only because they require configuration on your part. It is most often the case that users of Flurry Analytics find as much or more value in the Advanced Features listed below as in the features enabled by the above steps. Examples include:

- Custom Events - Understand the interaction between your users and specific areas of functionality in the app.
- Demographics - Analyze and segment your users by demographic group based on the data they share with your app.

Flurry tvOS currently does not support Flurry Pulse and Crash Analytics.

3. Advanced Features

The following methods are recommended to report additional data and enhance understanding of user behavior in your app. For a list of all APIs support by the Flurry Analytics SDK please see: ??

Tracking User Behavior

Utilizing Custom Events in your app, you can track the occurrence and state of most actions taken by users or even your app itself. Custom Events support functionality within Flurry Analytics such as Funnels, Segments, User Paths and others. To gain a better understanding of the value of Custom Events, see the video walkthrough available here.

Note that each of the logEvent APIs will return a status code that will indicate if the event was successfully recorded or not. Your application is currently limited to counting occurrences for 100 different event ids (maximum length 255 characters) during one application session. Maximum of 10 event parameters per event is supported. There is limit of 1000 events in total for a session (for instance, you can have 100 different event ids each occurring up to 10 times during the application session) .

```
[Flurry logEvent:@"EVENT_NAME"];
```
Use *logEvent* to count the number of times certain events happen during a session of your application. This can be useful for measuring how often users perform various actions, for example. Your application is currently limited to counting occurrences for 300 different event ids (maximum length 255 characters) across all its sessions. When deciding on events to track consider adding dynamic parameters to capture various incarnations of an event (rather than creating multiple dynamic events)

```
[Flurry logEvent:@"EVENT_NAME" withParameters:YOUR_NSDictionary];
```
Use this version of logEvent to count the number of times certain events happen during a session of your application and to pass dynamic parameters to be recorded with that event. Event parameters can be passed in as a NSDictionary object (immutable copy) where the key and value objects must be NSString objects. For example, you could record that a user used your search box tool and also dynamically record which search terms the user entered.

An example NSDictionary to use with this method could be:
```
NSDictionary *dictionary =
[NSDictionary dictionaryWithObjectsAndKeys:@"your dynamic parameter
value",
                                                @"your dynamic parameter name",
                                                nil];
```

`[Flurry logEvent:@"EVENT_NAME" timed:YES];`
Use this version of *logEvent* to start timed event.

`[Flurry logEvent:@"EVENT_NAME" withParameters:YOUR_NSDictionary timed:YES];`
Use this version of *logEvent* to start timed event with event parameters.

`[Flurry endTimedEvent:@"EVENT_NAME" withParameters:YOUR_NSDictionary];`
Use *endTimedEvent* to end timed event before app exits, otherwise timed events automatically end when app exits. When ending the timed event, a new event parameters NSDictionary object can be used to update event parameters. To keep event parameters the same, pass in nil for the event parameters NSDictionary object.

Tracking Application Errors
`[Flurry logError:@"ERROR_NAME" message:@"ERROR_MESSAGE" exception:e];`
Use this to log exceptions and/or errors that occur in your app. Flurry will report the first 10 errors that occur in each session. The message parameter has been deprecated as of release 4.2.2 and the passed in message will not be viewable on the Flurry developer portal.

For the following features, please call these APIs before calling
`[Flurry startSession:@"YOUR_API_KEY"]`:

Tracking Demographics
`[Flurry setUserID:@"USER_ID"];`
Use this to log the user's assigned ID or username in your system after identifying the user.

`[Flurry setAge:21];`
Use this to log the user's age after identifying the user. Valid inputs are 0 or greater.

`[Flurry setGender:@"m"];`
Use this to log the user's gender after identifying the user. Valid inputs are *`m`* (male) or *`f`* (female)

Tracking Location

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];
[locationManager startUpdatingLocation];

CLLocation *location = locationManager.location;
[Flurry setLatitude:location.coordinate.latitude
           longitude:location.coordinate.longitude
           horizontalAccuracy:location.horizontalAccuracy
           verticalAccuracy:location.verticalAccuracy];
```

This allows you to set the current GPS location of the user. Flurry will keep only the last location information. If your app does not use location services in a meaningful way, using CLLocationManager can result in Apple rejecting the app submission.

Controlling Data Reporting

```
[Flurry setBackgroundSessionEnabled:(BOOL)backgroundSessionEnabled];
```

This option is disabled by default. When enabled, Flurry will not finish the session if the app is paused for longer than the session expiration timeout. The session report will not be sent when the application is paused and will only be sent when the application is terminated. This allows for applications that run in the background to keep collecting events data. The time application spends in the background contributes to the length of the application session reported when the application terminates.

```
[Flurry setTVEventFlushCount:(short)count];
```

By default the SDK will send a partial session report once there are 10 events queued. This method can be used to change this default value. The minimum and maximum values are 5 and 50 respectively.

```
[Flurry setTVSessionReportingInterval:(short)count];
```

By default the SDK will send a partial session report once 5 minutes of session has elapsed. This method can be used to change this default value. The minimum and maximum values are 5 and 60 respectively.

4. Client-Server App Integration

Server-Client tvOS Apps require the Application Delegate to conform to TVApplicationControllerDelegate. Inside the “appController:evaluageAppJavaScriptInContext” of TVApplicationControllerDelegate you need to register the JSContext with the Flurry tvOS SDK.

Objective-C Code :

```
- (void)appController:(TVApplicationController *)appController
evaluateAppJavaScriptInContext:(JSContext *)jsContext {
    [Flurry registerJSContextWithContext:jsContext];
}
```

Swift Code:

```
func appController(appController: TVApplicationController,
evaluateAppJavaScriptInContext jsContext: JSContext) {
        Flurry.registerJSContextWithContext(jsContext)
    }
```

This makes the following JavaScript methods globally available to the JavaScript domain.

- flurryLogEvent({String} eventName)
- flurryLogEvent({String} eventName, {object} params)
- flurryLogTimedEvent({String} eventName)
- flurryLogTimedEvent({String eventName, {object} params)
- flurryEndTimedEvent({String eventName, {object} params)
- flurryLogError({String} eventName, {String} message, {object} error)
    - error -> { errorDomain: {String}, errorID: {Number}, userInfo: {object}}
- flurrySetUserID({String} userID)
- flurrySetGender({String} gender)
- flurrySetAge({Number} age)
- flurrySetLocation({Number} latitude, {Number} longitude, {Number} horizontalAccuracy, {Number} verticalAccuracy)

5. FAQ

How much does the Flurry Analytics SDK add to my app size?

The Flurry SDK will typically add 150 KB to the final app size.

When does the Flurry Agent send data?

By default, the Flurry tvOS Agent will send the stored metrics data to Flurry servers when the app starts, the event count or duration watermarks are hit, the app pauses, resumes, and terminates. To override default Agent behavior, you can turn off sending data on termination by adding the following call before you call `startSession`:

```
[Flurry setSessionReportsOnCloseEnabled:NO];
```

You can modify the values for the event count and duration watermarks but setting your own custom values using the `[Flurry setTVEventCountThreshold:(short)count]` and `[Flurry setTVDurationThreshold:(short)count]`

How much data does the Agent send each session?

All data sent 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.

What data does the Agent send?

The data sent by the Flurry Agent includes time stamps, logged events, logged errors, and various device specific information. This is the same information that can be seen in the custom event logs on in the Event Analytics section. We do not collect personally identifiable information.

What version of XCode is required?

The Flurry tvOS SDK will support Xcode 7 and above.

Does the Agent support Bitcode?

Yes, Flurry tvOS Analytics is bit code compatible.

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

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