Welcome to Flurry Analytics!

This file contains:

1. Introduction
2. Integration Instructions
3. Optional Features
4. Recommendations
5. FAQ

1. Introduction

The Flurry iOS Analytics Agent allows you to track the usage and behavior of your iOS application on users' phones 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 3.2.5 or above. If you need an SDK for an older Xcode version please email support.

This archive should contain these files:
- **ProjectApiKey.txt** : This file contains the name of your project and your project's API key.
- **Analytics-README.pdf** : This file containing instructions on how to use Flurry Analytics.
- **FlurryAnalytics/FlurryAnalytics.h** : The required header file header file containing methods for Flurry Analytics.
- **FlurryAnalytics/libFlurryAnalytics.a** : The required library containing Flurry's collection and reporting code.

There are additional folders for use with Flurry AppCircle. These optional libraries provide alternate streams of revenue for your apps. If you would like to use Flurry AppCircle please refer to AppCircle-README.pdf.

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).

We also recommend calling FlurryAnalytics from the main thread. FlurryAnalytics is not supported when called from other threads.

2. Integration

1. In the finder, drag FlurryAnalytics/ into project's 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: **Project > Add to project > FlurryAnalytics**
      - Choose 'Recursively create groups for any added folders'

3. In your Application Delegate:
- Import FlurryAnalytics and inside "applicationDidFinishLaunching:" add: [FlurryAnalytics startSession:@"YOUR_API_KEY"];

```objc
#import "FlurryAnalytics.h"
- (void)applicationDidFinishLaunching:(UIApplication *)application {
[FlurryAnalytics startSession:@"YOUR_API_KEY"];
//your code
}
```

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

3. Optional / Advanced Features

You can use the following methods to report additional data.

Tracking User Behavior
```objc
[FlurryAnalytics 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).

```objc
[FlurryAnalytics 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 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. Your application is currently limited to counting occurrences for 100 different event ids (maximum length 255 characters). Maximum of 10 event parameters per event is supported.

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

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

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

```objc
[FlurryAnalytics endTimedEvent:@"EVENT_NAME" withParameters:YOUR_NSDictionary];
```
Use endTimedEvent to end timed event before app exists, otherwise timed events automatically end when app exists. 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.

```objc
[FlurryAnalytics logAllPageViews:navigationController];
```
To enable Flurry agent to automatically detect and log page view, pass in an instance of UINavigationController or UITabBarController to countPageViews. Flurry agent will create a delegate on your object to detect user interactions. Each detected user interaction will automatically be logged as a page view. Each instance needs to only be passed to Flurry agent once. Multiple UINavigationController or UITabBarController instances can be passed to Flurry agent.

```objc
[FlurryAnalytics logPageView];
```

In the absence of UINavigationController and UITabBarController, you can manually detect user interactions. For each user interaction you want to manually log, you can use logPageView to log the page view.

Tracking Application Errors

```
[FlurryAnalytics 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.

Tracking Demographics

```
[FlurryAnalytics setUserID:@"USER_ID"];
```

Use this to log the user's assigned ID or username in your system after identifying the user.

```
[FlurryAnalytics setAge:21];
```

Use this to log the user's age after identifying the user. Valid inputs are 0 or greater.

```
[FlurryAnalytics 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;
[FlurryAnalytics 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

```
[FlurryAnalytics setSessionReportsOnCloseEnabled:(BOOL)sendSessionReportsOnClose];
```

This option is on by default. When enabled, Flurry will attempt to send session data when the app is exited as well as it normally does when the app is started. This will improve the speed at which your application analytics are updated but can prolong the app termination process due to network latency. This option mostly applies for devices running < iOS 3.2 that do not enable multi-tasking.

```
[FlurryAnalytics setSessionReportsOnPauseEnabled:(BOOL)sendSessionReportsOnPause];
```

This option is off by default. When enabled, Flurry will attempt to send session data when the app is paused as well as it normally does when the app is started. This will improve the speed at which your application analytics are updated but can prolong the app pause process due to network latency.

```
[FlurryAnalytics setSecureTransportEnabled:(BOOL)secureTransport];
```

This option is off by default. When enabled, Flurry will send session data over SSL when the app is paused as well as it normally does when the app is started. This has the potential to prolong the app pause process due to added network latency from secure handshaking and encryption.

4. Recommendations

We recommend adding an uncaught exception listener to your application (if you don't already have one) and use logError to record any application crashes.
Adding an uncaught exception listener is easy; you just need to create a function that looks like the following:

```
void uncaughtExceptionHandler(NSException *exception) {
[FlurryAnalytics logError:@"Uncaught" message:@"Crash!" exception:exception];
```

```
        }
```

You then need to register this function as an uncaught exception listener as follows:

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
NSSetUncaughtExceptionHandler(&uncaughtExceptionHandler);
[FlurryAnalytics startSession:@"YOUR_API_KEY"];
        ....
}
```

Note that you can name the function whatever you'd like and record whatever error information you'd like in the error name and event fields.

---

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 Agent will send the stored metrics data to Flurry servers when the app starts, 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`:
`[FlurryAnalytics setSessionReportsOnCloseEnabled:NO];`

Sending metrics data when the app pauses, but not enabled by default. You can enable sending data on pause with `startSession`:
`[FlurryAnalytics setSessionReportsOnPauseEnabled:YES];`

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.

If AppCircle is enabled, Flurry Agent will also send AppCircle user interaction data. More information is available in the AppCircle documentation.

Does the Agent support iOS OS 3.x?

To support OS 3.x, please set Base SDK to iOS 4.3 and iOS Deployment Target to iOS 3.x. Extra linker flags may be needed if NSConcreteGlobalBlock and UIBackgroundTaskInvalid runtime error occur under 3.x.

The linker flags are:
-weak_framework UIKit
-weak_library /usr/lib/libSystem.B.dylib

What version of XCode is required?

Going forward, the Flurry SDK will only support Xcode 3.2.5 and above. Please email support if you need to use

older versions of the Flurry SDK. This version of the Flurry SDK is compatible with Xcode 3.2.5 and designed for OS 4.3 (iOS) applications.

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