Arise Documentation

Release 2.7

Arise.io

January 31, 2014

Contents:

CHAPTER 1

Setup your first A/B test

1.1 Overview

1.1.1 What is A/B testing?

Testing is a way of showing users different variations of your app, and then measuring how each variation affects your goals. For example, if you're interested in having your users make more in-app purchases, you might test the phrase "buy more coins now!" versus "save time by buying coins". Arise's A/B testing tools can tell you which one will make you more money. You can also use Arise to test new in app-features, evaluate performance and roll out changes without resubmitting your app.

1.1.2 What is a conversion?

Tests are only good when they can measure some metric and show you which path is performing better. To help out our system, that metric should be a "conversion". Good examples of goals are things like an in-app purchase, a new account created, or even a tap on an advertisement. We can then calculate the conversion rate (Number of conversion/ Number of views).

1.2 Getting Started

1.2.1 1. Create an account

If you do not have an account yet, register on the dashboard. The dashboard is your tool to configure your experiments.

1.2.2 2. Create a new app

Enter an application name without any space. An application will contain a set of experiments. The application name will be used later in your code.

1.2.3 3. Create and configure your variation

Click on "Create a new experiment" to create your first experiment.

Two variations are created by default: variation A (original) and variation B (test).

Tag

The tag will be the identifier of your experiment in your iOS or Android code.

Variation values

Variation values will be transmitted to your app. You can experiment:

- call to actions
- headline, product descriptions (words, style, number of words)
- in-app purchases (prices, multiple contents)
- forms (types of fields, length, layout, error handling)
- layout and design (position and grouping of content)
- images

You need to set different values for each variation. We will consider the variation A as the default variation (original). You will be able to create up to 8 variations. You can create a new variation by clicking on the blue '+' button. However, you will get more reliable and faster results by testing as few variations as possible.

Distribution

You can set the distribution of your variations. The total of all the variations must always be 100%. You can for example set 70% on the variation A (original) and 30% on the variation B (test). Once your app is deployed, you will be able to view the conversion rate for each variation.

1.2.4 5. Start your experiment

Once you have configured your experiment, you will need to click on the "Start experiment" button. When an experiment is started, experiment values are not allowed to be modified. You will also not be able to add or remove a variation. However, the distribution can still be modified. Requests from new devices will use the new distribution. Devices that have already an experiment assigned won't change.

1.2.5 6. Install the Arise SDK in your app

We currently support iOS and Android. Please read the following documentation to install the Arise SDK in your app:

- *Arise iOS SDK*
- *Arise Android SDK*

1.2.6 7. End your experiment

Once you have significant results for an experiment, you have the choice between keeping the variation A (original) or using a test variation (B,C,etc.). All the devices will display the selected variation when you validate your choice.

1.2.7 8. Analyse your report

- *Understand your report*

1.2.8 9. Restart your experiment with new values

- *Restart an experiment*

CHAPTER 2

Arise iOS SDK

This document will help you to install the Arise iOS client in your application.

2.1 Installation steps

2.1.1 1. Add the Arise library to your project

First, download the Arise SDK for iOS. Unzip it and drag it inside your project's Frameworks folder in XCode. Ensure that you select 'Copy items into the destination group's folder' when the dialogue box appears.

2.1.2 2. Add dependencies

In XCode:

- select your project in the project navigator
- in the project settings select your target
- click on the Build Phases tab
- Open the Link With Libraries collapsible panel
- Click on '+'
- Search for libsqlite3.dylib and click on the Add button.

2.1.3 3. Initialize the framework

Add the following line in your AppDelegate.h:

```
#import <Arise/Arise.h>
```

Add the following line under application:didFinishLaunchingWithOptions of your AppDelegate.m file to initialize the framework:

```
[Arise initializeWithKey:@"9c51b5e8f06ebd26728f29954365098f052c68c8" appName:@"AngryElephants"];
```

Replace the value of the key and the app name by your own key and your application name. You can find it on your dashboard. This function will trigger a asynchronous sync with the server (experiments values are retrieved and events are sent).

2.1.4 4. Get the experiment value

You can request a variation value using the getVariation: method with the experiment tag name (same as the one displayed in your dashboard). You have also to set a default value in case of no connection to the server. "Buy it now" is the default value in the following code snippet.

```
[ABTest getVariation:@"ExpTag1" defaultValue:@"Buy it now" data:^(NSString *value){
    [_purchaseButton setTitle:value forState:UIControlStateNormal];
}];
```

Do not forget to import the Arise SDK in your header file:

```
#import <Arise/Arise.h>
```

The default value will only be printed if the application has never succeeded to connect to the server. We recommend to set the default value to the same value as your variation A value.

We also recommend to call this function as late as possible in your app.

2.1.5 5. Record events

Now that you have setup your application for testing, you will need to record views and conversion events. You need to provide the experiment tag name as a parameter to associate the events with an experiment.

Record a view:

```
[ABTest recordView:@"ExpTag1"];
```

Record a conversion:

```
[ABTest recordConversion:@"ExpTag1"];
```

Views and conversions events are stored on the device until an internet connection is available. Our framework does work properly even in case of no connectivity.

2.2 Full code example

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    // Get and setup the variation
    [ABTest getVariation:@"ExpTag1" defaultValue:@"Buy it now" data:^(NSString *value){
        // Use the variation value to customize our application
        // ...

        // For example :
        // Change the title of the purchase button
```

```
        [_purchaseButton setTitle:value forState:UIControlStateNormal];
    }];
}

- (void)onLoadPurchasePage
{
    // the user is viewing the item purchase page
    // record a view event
    [ABTest recordView:@"ExpTag1"];
}

- (IBAction)onPurchase:(id)sender
{
    // the user has bought the item
    // record a conversion event
    [ABTest recordConversion:@"ExpTag1"];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

2.3 Notes

The Arise iOS SDK supports iOS 5.0 and later.

CHAPTER 3

Arise Android SDK

This documentation will help you to install the Arise Android client in your application.

3.1 Installation steps

3.1.1 1. Add the Arise library to your project

First, download the Arise library jar file and drag it inside your project's /libs/ folder.

3.1.2 2. Add permissions

In your project, right click on AndroidManifest.xml Click on Open With/Android Common XML Editor. Add after the <uses-sdk .... /> tag:

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

3.1.3 3. Initialize the framework

In the onCreate function of your main activity, you need to initialize the framework:

```
// Initialize the arise library
String authKey = "9c51b5e8f06ebd26728f29954365098f052c68c8";
String appName = "AngryElephants";
Arise.initialize(getApplicationContext(), authKey, appName);
```

Replace the value of the key and the app name by your own key and your application name. You can find it on your dashboard. This function will trigger a asynchronous sync with the server (experiments values are retrieved and events are sent).

3.1.4 4. Get the experiment value

When you plan to run the experiment, you will need to call the getVariationWithListener with the experiment tag name (same as the one displayed in your dashboard) to get the experiment data. You have also to set a default value in case of no connection to the server. "Buy it now" is the default value in the following code snippet.

```
// Get and setup the variation
ABTest.getVariationWithListener("ExpTag1", "Buy it now", new VariationListener() {
    @Override
    public void onVariationAvailable(String value) {
            final String buyMessage = value;
    }
});
```

The default value will only be printed if the application has never succeeded to connect to the server. We recommend to set the default value to the same value as your variation A value.

We also recommend to call this function as late as possible in your app.

3.1.5 5. Record events

Now that you have setup your application for testing, you will need to record views and conversion events. You need to provide the experiment tag name as a parameter to associate the events with an experiment.

Record a view:

```
ABTest.recordView("ExpTag1");
```

Record a conversion:

```
ABTest.recordConversion("ExpTag1");
```

Views and conversions events are stored on the device until an internet connection is available. Our framework does work properly even in case of no connectivity.

3.2 Full code example

```
package com.example.shoestore;

import io.arise.ABTest;
import io.arise.Arise;
import io.arise.VariationListener;
import android.os.Bundle;
import android.app.Activity;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            // Initialize the arise library
            String authKey = "9c51b5e8f06ebd26728f29954365098f052c68c8";
            String appName = "AngryElephants";
            Arise.initialize(getApplicationContext(), authKey, appName);

            // Get and setup the variation
            ABTest.getVariationWithListener("ExpTag1", "Buy it now", new VariationListener() {
                    @Override
                    public void onVariationAvailable(String value) {
                            // Change the button label
```

```
                        final String buyMessage = value;
                        // Use the buyMessage to customize our application
                        // ...

                }
        });

    }

    private void onLoadPurchasePage(){
        // the user is viewing the item purchase page
        // record a view event
        ABTest.recordView("ExpTag1");
    }

    private void onPurchaseCompleted(){
        // the user has bought the item
        // record a conversion event
        ABTest.recordConversion("ExpTag1");
    }
}
```

3.3 Notes

The Arise Android SDK supports Android 2.3.3 (API level 10) and later.

CHAPTER 4

Arise PhoneGap SDK

This documentation will help you to install the Arise PhoneGap SDK in your html5 application.

4.1 Installation steps

4.1.1 1. Add the Arise PhoneGap plugin to your project

At the root of your project, run:

```
phonegap local plugin add https://github.com/poiuytrez/ArisePhoneGap.git
```

The plugin will be automatically downloaded and installed in your application. On an iOS project, you need to drag the ArisePhoneGap/libs/ios/Arise.framework folder inside your project's Frameworks folder in XCode. Ensure that you select 'Copy items into the destination group's folder' when the dialogue box appears.

4.1.2 2. Initialize the framework

You need to initialize the framework in the deviceReady function of your application:

```
// Initialize the arise library
var authKey = "9c51b5e8f06ebd26728f29954365098f052c68c8";
var appName = "AngryElephants";
// arise.initialize(success, error, authKey, appName);
arise.initialize(
    function(){ console.log("successfully initialized"); },
    function(error){ console.log("an error occurred:" + error); },
    authKey,
    appName
);
```

Replace the value of the key and the app name by your own key and your application name. You can find it on your dashboard. This function will trigger a asynchronous sync with the server (experiments values are retrieved and events are sent).

4.1.3 3. Get the experiment value

When you plan to run the experiment, you will need to call the getVariationWithListener with the experiment tag name (same as the one displayed in your dashboard) to get the experiment data. You have also to set a default value in case of no connection to the server. "Buy it now" is the default value in the following code snippet.

```
// arise.getVariationWithListener(success, error, experimentTag, defaultValue);
arise.getVariationWithListener(
    function(value){
        // print the variation
        console.log(value);
    },
    function(error){
        console.log("an error occurred:" + error);
    },
    "ExpTag1",
    "MyDefaultValue"
);
```

The default value will only be printed if the application has never succeeded to connect to the server. We recommend to set the default value to the same value as your variation A value.

We also recommend to call this function as late as possible in your app.

4.1.4 4. Record events

Now that you have setup your application for testing, you will need to record views and conversion events. You need to provide the experiment tag name as a parameter to associate the events with an experiment.

Record a view:

```
// arise.recordView(success, error, experimentTag);
arise.recordView(
    function(){ console.log("success")},
    function(error){ console.log("an error occurred:" + error) },
    "ExpTag1"
);
```

Record a conversion:

```
// arise.recordConversion(success, error, experimentTag);
arise.recordConversion(
    function(){ console.log("success")},
    function(error){ console.log("an error occurred:" + error) },
    "ExpTag1"
);
```

Views and conversions events are stored on the device until an internet connection is available. Our framework does work properly even in case of no connectivity.

4.2 Full code example

```
<!DOCTYPE html>
<html>
<head>
    <script>
        // Click on Initialize button
        function initialize(){
            // Initialize Arise
            // Initialize the arise library
            var authKey = "9c51b5e8f06ebd26728f29954365098f052c68c8";
            var appName = "AngryElephants";
```

```
            // arise.initialize(success, error, authKey, appName);
            arise.initialize(
                    function(){ console.log("successfully initialized"); },
                    function(error){ console.log("an error occurred:" + error); },
                    authKey,
                    appName
            );
        }

        // Click on GetVariation button
        function getVariation(){
            // arise.getVariationWithListener(success, error, experimentTag, defaultValue);
            arise.getVariationWithListener(
                    function(value){
                        // print the variation
                        alert(value);
                    },
                    function(error){
                        console.log("an error occurred:" + error);
                    },
                    "ExpTag1",
                    "MyDefaultValue"
            );
        }

        // Click on Record View button
        function recordView(){
            // arise.recordView(success, error, experimentTag);
            arise.recordView(
                    function(){ console.log("success")},
                    function(error){ console.log("an error occurred:" + error) },
                    "ExpTag1"
            );
        }

        // Click on Record Variation button
        function recordConversion(){
            // arise.recordConversion(success, error, experimentTag);
            arise.recordConversion(
                    function(){ console.log("success")},
                    function(error){ console.log("an error occurred:" + error) },
                    "ExpTag1"
            );
        }
    </script>
</head>
<body>
<button onclick='initialize()'>Initialize</button>
<button onclick='getVariation()'>Get variation</button>
<button onclick='recordView()'>Record view</button>
<button onclick='recordConversion()'>Record conversion</button>

<script type="text/javascript" src="phonegap.js"></script>

</body>
</html>
```

4.3 Notes

The Arise PhoneGap SDK supports PhoneGap 3.X on Android and iOS.

CHAPTER 5

Understand your report

5.1 Overview

Your experiment report is your tool to understand which variation was the most effective.

5.2 Views and conversions

A variation is assigned by the server as soon as you run the initialize function. The system will add one view or one conversion to the report for each event. One user can record more than one view or conversions.

If the Arise client could not reach the server, the client will fallback on the default variation value and no events will be recorded (views or variations).

5.2.1 Examples

Example 1

	Variation A	Variation B
views	1290	1300
conversions	130	340
conversion rate	10%	26%

This table means that:

- 1290 view events has been recorded for the the variation A
- 1300 view events has been recorded for the the variation B
- 130 conversion events has been recorded for the the variation A
- 340 conversion events has been recorded for the the variation B

The conversion rate is the total number of conversion events divided by the total number of view events.

Keep in mind that:

- one user might record multiple view events for the variation which has been assigned to him.
- one user might record multiple conversion events for the variation which has been assigned to him.

In this example, variation B was better than variation A.

Example 2

	Variation A	Variation B
views	2340	2251
conversions	2100	3450
conversion rate	89%	153%

Seems like a bug in our system? No, it's a feature! This experiment could have been implemented in a very addictive game.

A game over screen is shown to the player when he has no more life:

Variation A	**Variation B**
Game over Buy more lives to continue [1 life] [5 lives] [25 lives]	You were soooooo closed to the final boss! Wanna refill? Lives are even better than energy drinks. [1 life] [5 lives] [25 lives]

One view event is recorded when the screen is displayed.
One conversion event is recorded **for each life** bought.

In this example, the results mean that a player purchases in average 0.89 life with the variation A and 1.53 lives with the variation B. The revenue is higher with the variation B than with the variation A!

This is why in some cases we can have more conversion events than views! The goal here was to optimize the number of lives purchased.

You might now want to *Restart an experiment* with new values.

CHAPTER 6

Restart an experiment

You might want to restart the **same experiment** but with **new values** once an experiment is finished.

The trick is to **create a new experiment with the same tag** as the old experiment. The **clone** button on the archived experiment will do it for you. A new experiment will be created with the same tag as the old experiment. This way when you will publish your new experiment, your mobile application will automatically pull variation values from the new experiment. To sum up:

1. Archive your old experiment by clicking on “Use Variation X”
2. Clone the old experiment –> A new experiment will be created with the same tag as the old experiment
3. Set values for the new experiment
4. Start your new experiment –> You mobile app will automatically show variations from the new experiment

CHAPTER 7

Flurry Analytics integration

This document will help you to integrate Arise with your Flurry account.

7.1 Introduction

If you use Flurry to measure your app traffic, you can now see Arise data alongside the rest of your Flurry data. Arise will push its data to Flurry directly from the mobile app. You will then be able to segment any Flurry report by experiment variation. Integrating Arise with Flurry will let you compare two variation using advanced metrics provided by Flurry.

7.2 Implementation instructions

7.2.1 1. Add Flurry to your project

Follow the Flurry Analytics Getting started tutorial. This tutorial will help you to install Flurry in your application.

7.2.2 2. Activate Flurry reporting from Arise

Right after the initializeWithKey method, call enableFlurry to enable Arise to report its events to Flurry.

On iOS:

```
[Arise initializeWithKey:@"9c51b5e8f06ebd26728f29954365098f052c68c8" appName:@"AngryElephants"];
[Arise enableFlurry];
```

On Android:

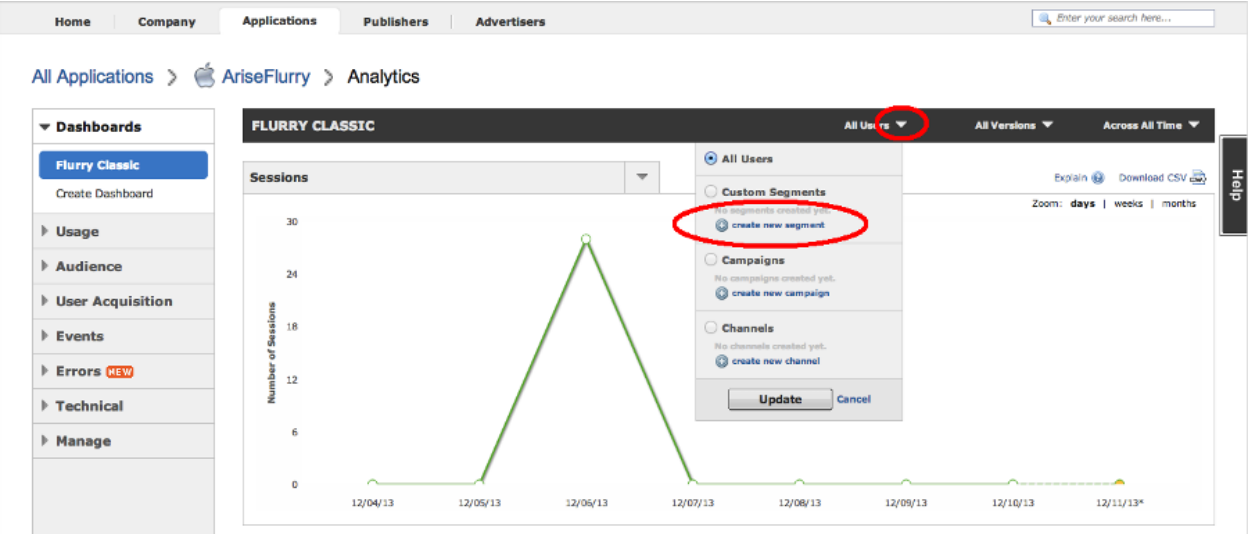
```
Arise.initialize(getApplicationContext(), "9c51b5e8f06ebd26728f29954365098f052c68c8", "AngryElephants
Arise.enableFlurry();
```

Arise will send custom events with experiments and variations data to Flurry.

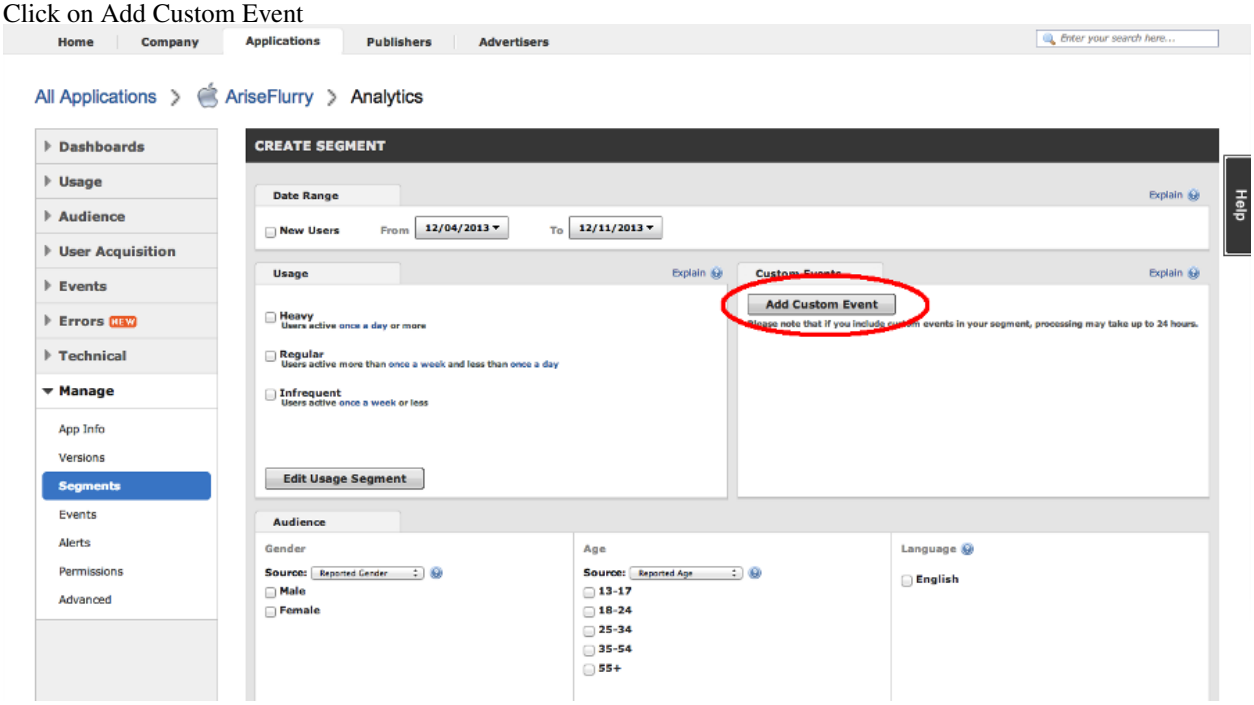
7.2.3 3. Segment your data in Flurry

Flurry might take up to a day to show events data. An event is a view or a conversion. For each event, a Type ("View" or "Conversion") and a Variation ("A" or "B") is attached. As an example, we are going to segment our users for the Variation A and Variation B of the Experiment1.

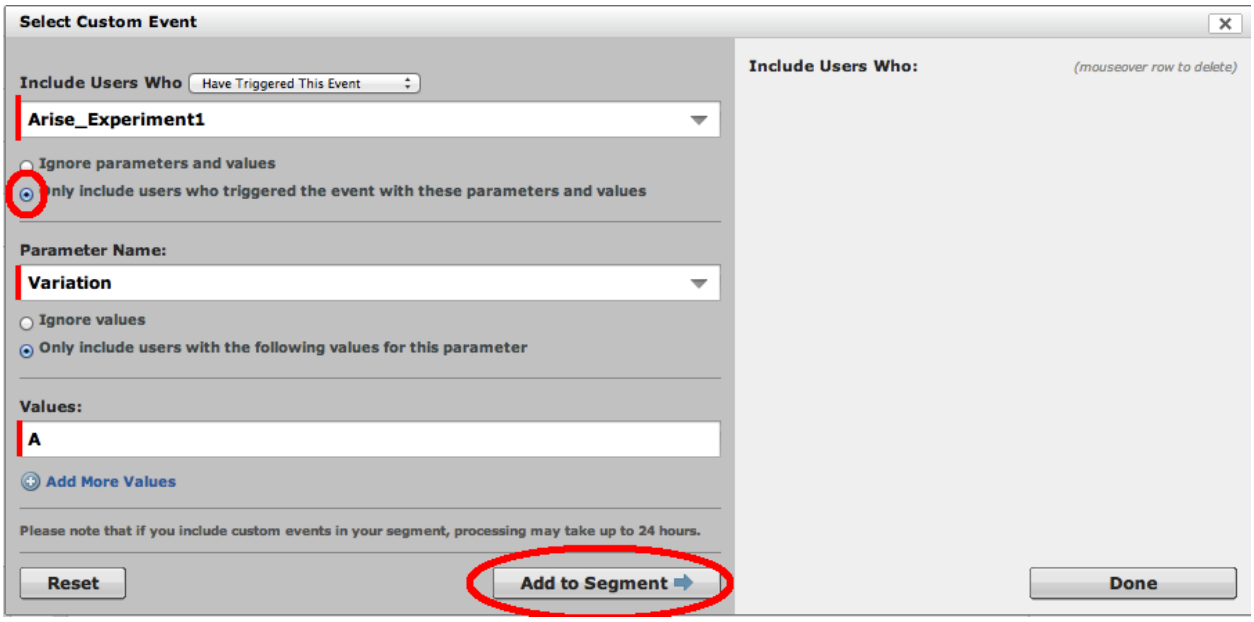
In Dashboards -> Flurry Classic, click on All users and the top and select create new segment.



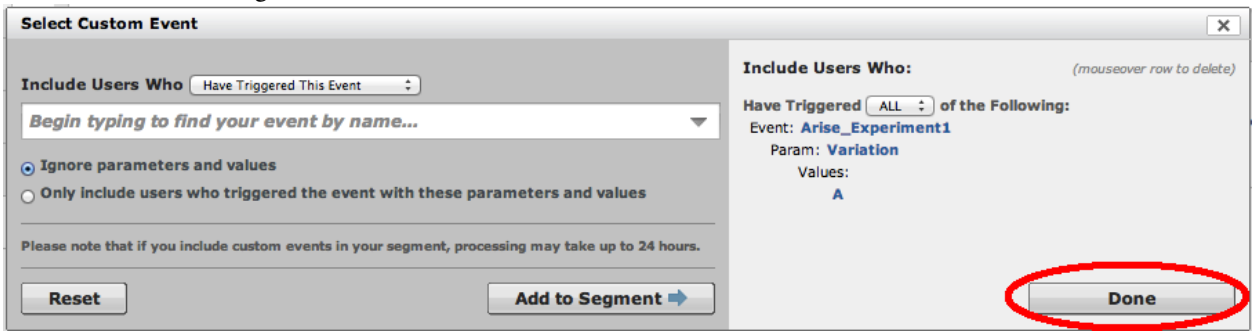
Click on Add Custom Event



- Enter Arise_Experiment1 for the name of your event (select it from the drop down menu).
- Select the “Only include users who triggered the event with these parameters and values” option.
- Select “Variation” as Parameter Name
- Select the “Only include users with the following values for this parameters” option
- Enter “A” as a value



Click on the "Add to Segment" button.

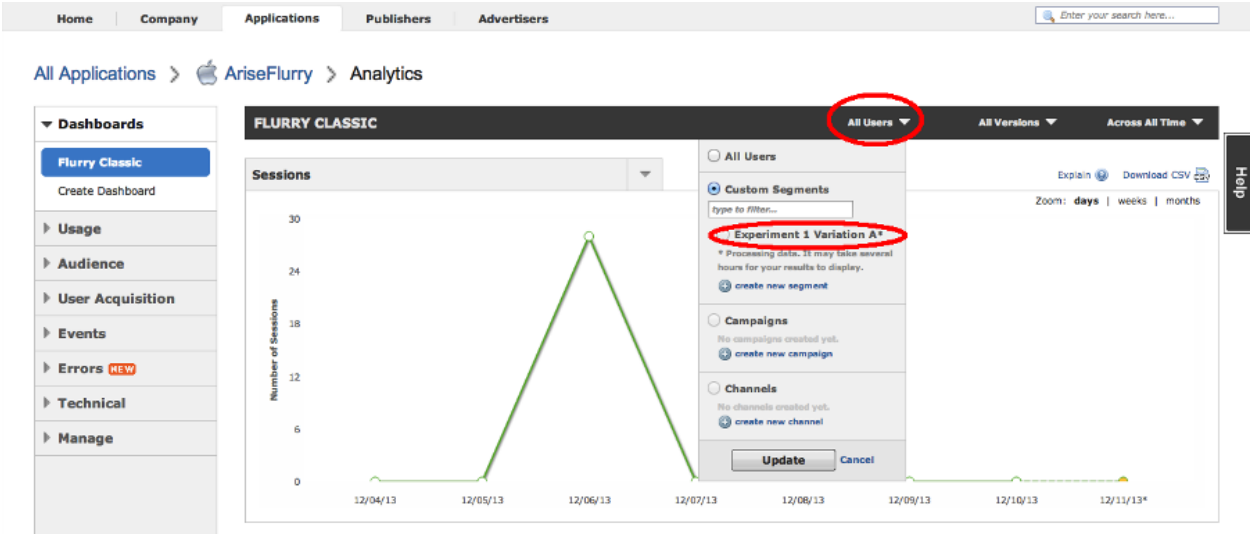


Click on the "Done" button. At the bottom of the Create Segment panel, enter "Arise Experiment 1 Variation A" as segment name and click on "Create this segment".

Repeat all the step above to add a segment for each variation of your experiment.

7.2.4 4. Analyze your users behavior

Flurry usually takes around 48 hours before reporting events. To view all the Flurry metrics for one variation, click on "All Users" at the top of the page and select your experiment variation in the Custom Segment section.


Home
Company
Applications
Publishers
Advertisers
Enter your search here...
All Applications
AriseFlurry
Analytics
Dashboards
Flurry Classic
Create Dashboard
Usage
Audience
User Acquisition
Events
Errors
NEW
Technical
Manage
FLURRY CLASSIC
All Users
All Versions
Across All Time
Sessions
Explain
Download CSV
Zoom: days | weeks | months
Number of Sessions
30
24
18
12
6
0
12/04/13
12/05/13
12/06/13
12/07/13
12/08/13
12/09/13
12/10/13
12/11/13*
All Users
Custom Segments
type to filter...
Experiment 1 Variation A*
* Processing data. It may take several hours for your results to display.
create new segment
Campaigns
No campaigns created yet.
create new campaign
Channels
No channels created yet.
create new channel
Update
Cancel
Help

CHAPTER 8

Segmentation

This document will help you to understand, define and integrate segments.

8.1 Introduction

Segmentation helps you to run an experiment only on a segment of a population. A "Dimension" is a key concept in segmentation. A dimension is a characteristic of your population. For example the gender is a characteristic of a population. A dimension is divided into multiple segment. For example the "genre" dimension is divided between "male" and "female". Arise can help you segment your users for a single dimension. Arise do not support yet multiple dimensions. Few examples of dimensions that can be implemented:

- genre
- location
- already a buyer
- etc.

You have to define the segment value of the dimension using the Arise SDK. On the dashboard you will be able to run your experiment only on a specific segment. For example, you can set the gender value with the Arise SDK (male or female) then run an experiment only on the female segment to test a new promotion.

8.2 Implementation instructions

8.2.1 1. Define the segment value

Add the segment parameter to the initialize method. In this example we set the segment value for the genre. The segment value should be retrieve for the info

On iOS:

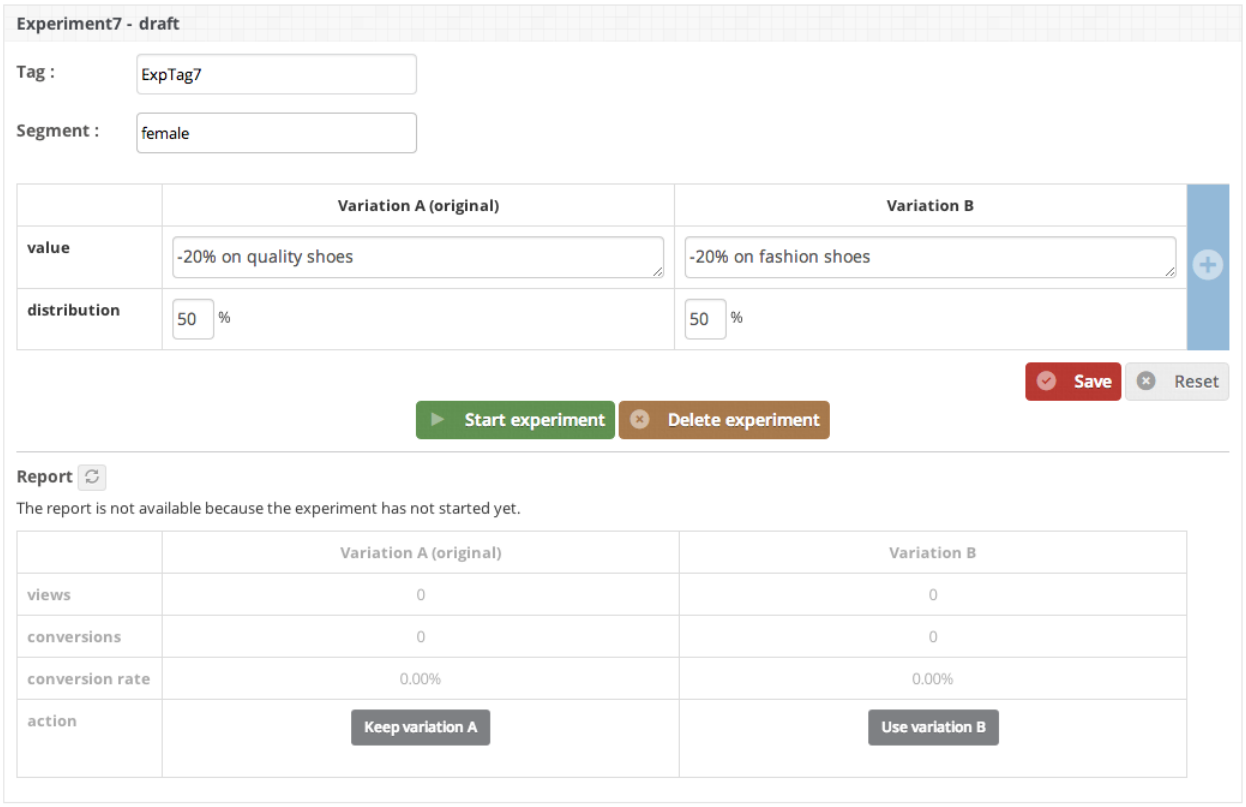
```
[Arise initializeWithKey:key appName:appname setSegment:@"female"];
```

On Android:

```
Arise.initialize(getApplicationContext(), authKey, appName, "female");
```

8.2.2 2. Segment your experiment

On your dashboard, you can enter the segment value for the experiment. If you want to run the experiment only on females, just enter "female" in the Segment input box. Other segment(s) will not be part of the experiment and will display a default value. Events of users not in the experiments will be also ignored.



CHAPTER 9

Release notes

9.1 Version 2.8

January 31th 2014

9.1.1 iOS

- Segmentation support

9.1.2 Android

- Segmentation support

9.2 Version 2.7

December 5th 2013

9.2.1 iOS

- Flurry Analytics integration

9.2.2 Android

- Flurry Analytics integration
- Potential conflict with external loopj library removed

9.3 Version 2.6.3

November 26th 2013

9.3.1 iOS

- Events ignored when recorded to quickly bug fix

9.4 Version 2.6.2

September 17th 2013

9.4.1 Html5

- Added compatibility with PhoneGap 3.0.0 for iOS.

9.5 Version 2.6.1

September 4th 2013

9.5.1 Html5

- Added compatibility with PhoneGap 3.0.0 for Android.

9.6 Version 2.6

August 30th 2013

Backward compatible with iOS and Android Arise SDK version 2.5.

9.6.1 Dashboard

- Adding support experiments tags. Mobile apps will the this tag to refer to the experiment instead of the experiment name. It means that you can run a new experiment with new values without resubmitting your app to the store.
- Other experiments are automatically collapsed when a new one is created
- Fix issue with empty distribution value

9.6.2 iOS

- Support for experiment tags
- AFNetworking library renamed to avoid conflicts with an existing library

9.6.3 Android

- Support for experiment tags

9.6.4 Migration notes from 2.5 to 2.6

SDK 2.5 will still work with experiments names. To use experiment tags:

1. Update your SDK
2. Replace your experiments names with experiments tags names in your source code

9.7 Version 2.5

August 27th 2013

9.7.1 Dashboard

- Adding support for multiple applications. Each application will be able to contain an unlimited number of experiments
- Experiments collapsible status are now saved
- One step registration
- Migration of existing Arise version 1.0 users

9.7.2 iOS

- Experiment values are now returned instantly if the system has not retrieved any values from the server. We had a 5 seconds waiting time on the previous version
- Support for application names on initialization

9.7.3 Android

- Experiment values are now returned instantly if the system has not retrieved any values from the server. We had a 5 seconds waiting time on the previous version
- Support for application names on initialization

9.8 Version 2.4

August 22th 2013

9.8.1 Dashboard

- Adding support up to 8 variations per experiment (A/B/N testing)
- Stunning performances (no page reload)

9.8.2 iOS

- A default value can be defined
- Multiple variations support
- Platform support lowered form iOS 6.0 and up to iOS 5.0 and up.

9.8.3 Android

- A default value can be defined
- Multiple variations support

9.9 Version 2.3

August 6th 2013

9.9.1 Dashboard

- Multiple experiments support
- Experiment status (draft, active, archived)
- Experiments info and report merged on the same page

9.9.2 iOS

- Better getVariation callback handling (works even at the first launch of the app)
- Less http requests (the registration with the server is now cached)
- Multiple experiments support

9.9.3 Android

- Less http requests (the registration with the server is now cached)
- Multiple experiments support

9.10 Version 2.2

July 28th 2013

First release of the new Arise platform (2.x). Versions 2.0 and 2.1 were never released to the public.

CHAPTER 10

Indices and tables

- *genindex*
- *modindex*
- *search*