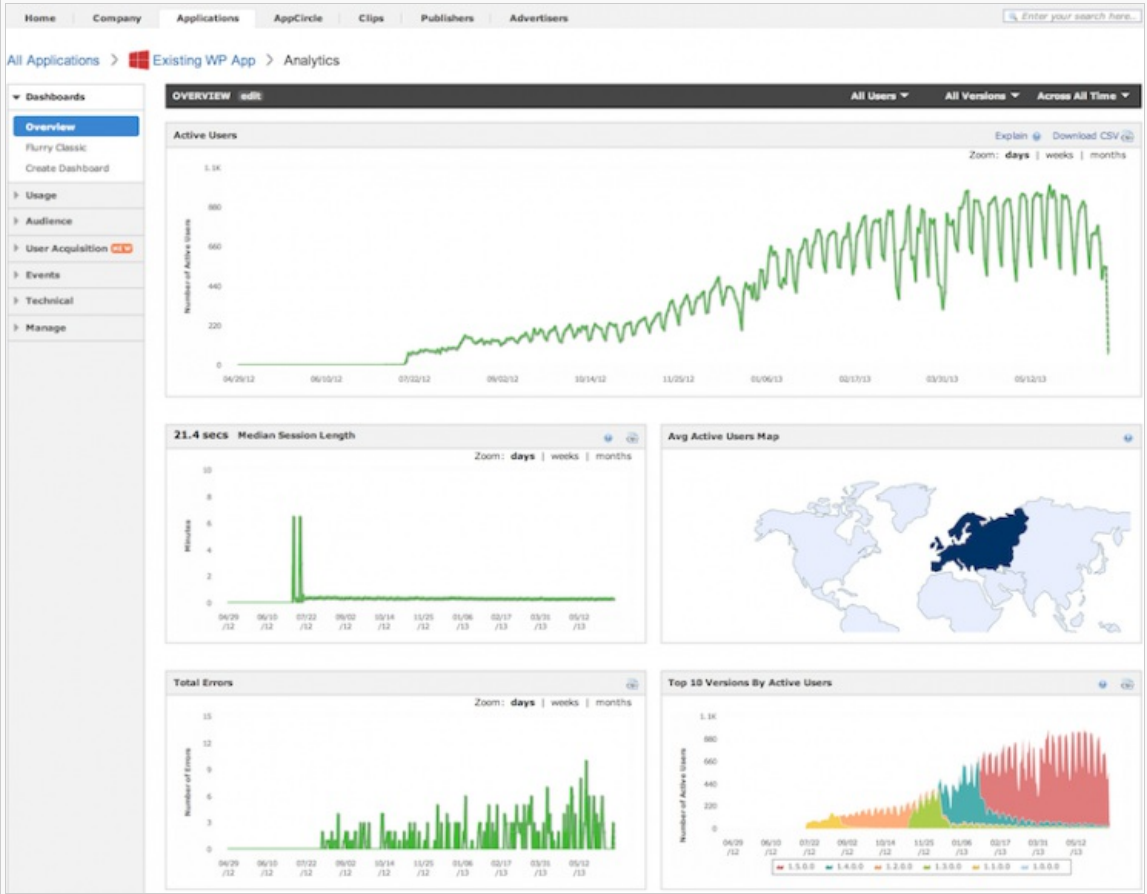
How to add Flurry Analytics to your Windows Phone app

This article explains how to add Flurry Analytics to your Windows Phone 7 or Windows Phone 8 app.

23 Jun 2013

Introduction

The Flurry Analytics Agent for Windows Phone allows you to track the usage and behavior of your Windows Phone 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. Flurry Analytics is also **FREE**.



App overview in Flurry Analytics

Use analytics to answer questions about your users and how your app is used over time:

- If you currently have a Windows Phone 7 app in the Windows Phone Store, are you curious to know what percentage of your users use WP7 versus WP8?
- Your app has many features but which are most popular and which are not used at all?
- The Windows Phone Dev Center may tell you the number of downloads, but how many people actively use your app daily/weekly/monthly?
- How many times and for how long do people use your app each day?

Prerequisites

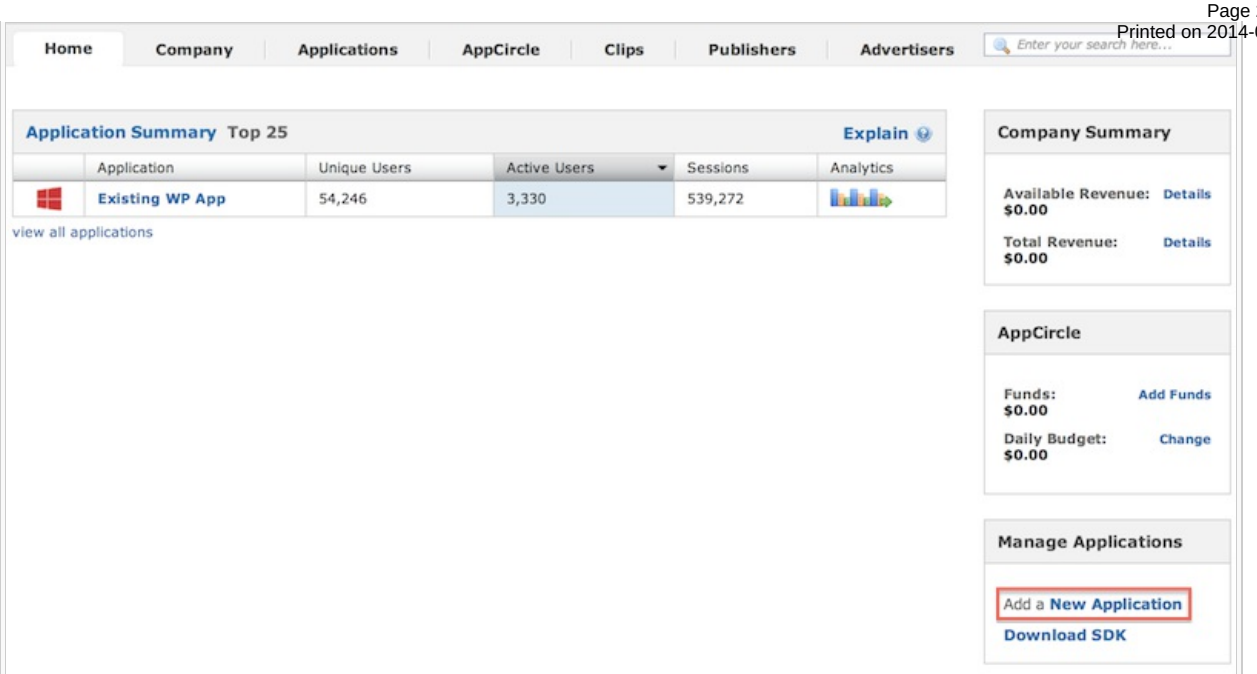
The following are required in order to use Flurry Analytics:

1. A Flurry Analytics account. Visit the Flurry website and sign up if you don't already one. It's **FREE**.
2. A Windows Phone 7 or Windows Phone 8 app.

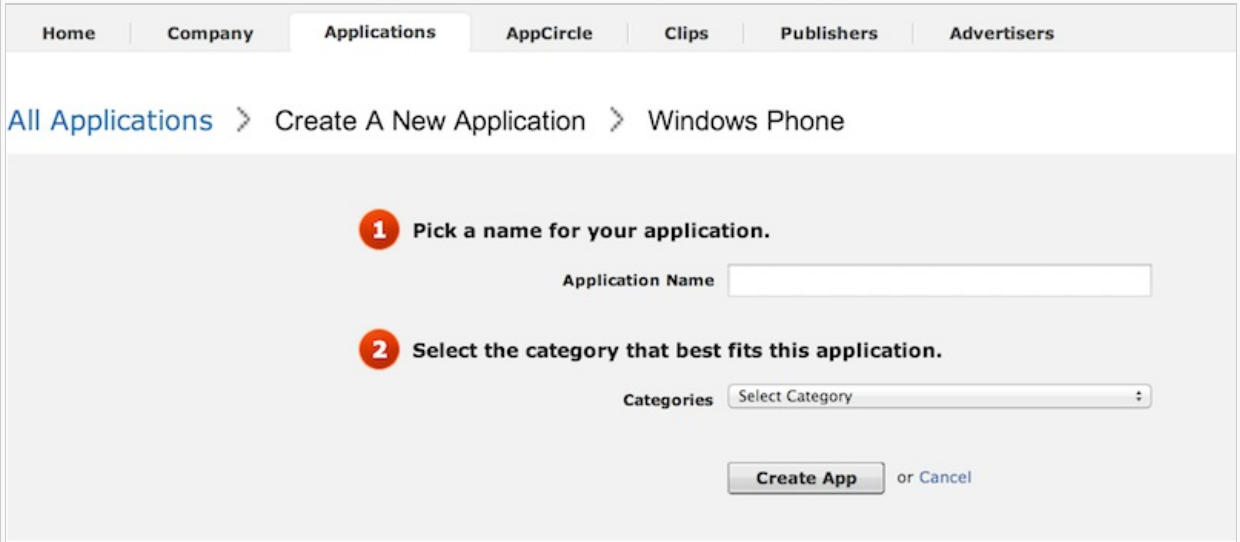
Flurry SDK and app API key

1. Add a new application to your Flurry account

(C) Copyright Nokia 2014. All rights reserved.

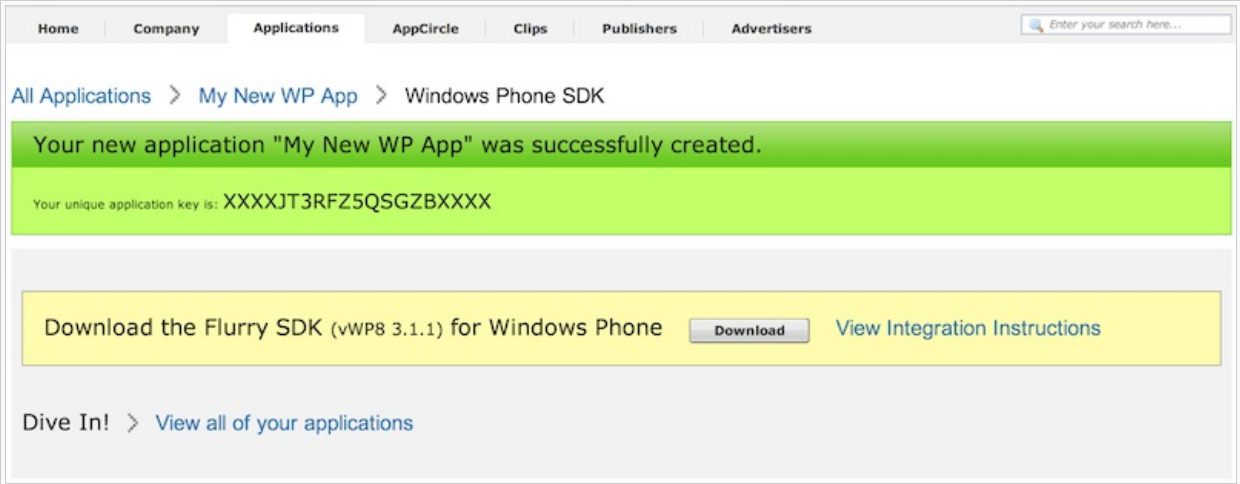


Add new application



Add an application name (you can change this later) and category

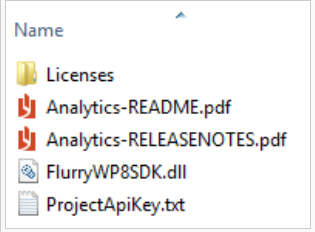
2. Download the SDK



Download the .ZIP file containing the SDK

3. Get the API key for your app from **ProjectApiKey.txt**

(C) Copyright Nokia 2014. All rights reserved.



Get API key unique to your app

App integration

1. Reference **FlurryWP8SDK.dll** in your application project
2. Configure your **WMAppManifest.xml** to have access to networking (`ID_CAP_NETWORKING`) and device information (`ID_CAP_IDENTITY_DEVICE`)
3. Incorporate the following two lines of Flurry code:

```
private void Application_Launching(object sender, LaunchingEventArgs e)
{
    FlurryWP8SDK.Api.StartSession("FLURRY_API_KEY");
    // ....
}
```

```
private void Application_Activated(object sender, ActivatedEventArgs e)
{
    // Restart session
    FlurryWP8SDK.Api.StartSession("FLURRY_API_KEY");
}
```

From the addition of those two simple lines of code, your app will report the following details back to your Flurry account:

Sessions	Active Users	New Users
Session Length	Frequency of Use	Benchmarks
Page Views	Custom User Segments	User Paths
Lifecycle Metrics	User Retention	Version Adoption
Cross-App Usage	Category Interests	Demographic Estimates
Time of Day Reporting	Geographic Usage	Language Metrics
Devices	Carriers	Firmware Versions

Custom tracking events

Log events

```
FlurryWP8SDK.Api.LogEvent(string eventId, bool timed, List<Parameter> parameters)
```

Use `LogEvent()` to track user Events that happen during a session. You can track how many times each Event occurs, what order Events happen in, as well as what the most common parameters are for each Event. This can be useful for measuring how often users take various actions, or what sequences of actions they usually perform. Each project supports a maximum of 300 Events. The timed argument and the parameters argument are both optional. Each Event id, parameter key, and parameter value must be no more than 255 characters in length. Each Event can have no more than 10 parameters. If the timed argument is true, that means you are logging a timed Event.

(C) Copyright Nokia 2014. All rights reserved.

Mark end of timed events

```
FlurryWP8SDK.Api.EndTimedEvent(string eventId, List<Parameter> parameters)
```

Use `EndTimedEvent` with the same `eventid` used to start the timed Event to end the timer.

Log errors

```
FlurryWP8SDK.Api.LogError(String message, Exception exception)
```

Use `LogError()` to report application errors. Flurry will report the last 10 errors to occur in each session. (max length 255 chars)

Optional configuration methods

Call these methods after calling `StartSession()` to change the configuration:

```
FlurryWP8SDK.Api.SetVersion(string versionName)
```

Change the version name your analytic data is reported under. If this is not specified, the version name is retrieved from the application descriptor.

Tracking Demographics

```
FlurryWP8SDK.Api.SetUserId(string userId)
```

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

```
FlurryWP8SDK.Api.SetAge(int age)
```

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

```
FlurryWP8SDK.Api.SetGender(Gender gender)
```

Use this to log the user's gender after identifying the user.

Tracking Location

```
FlurryWP8SDK.Api.SetLocation(double latitude, double longitude, float accuracy)
```

This allows you to set the current GPS location of the user. Flurry will keep only the last location information.

Controlling Data Reporting

```
FlurryWP8SDK.Api.SetSessionContinueSeconds(int seconds)
```

Pass a value to change the number of seconds for which paused sessions will be continued. After this amount of time has passed with no activity, a new session is assumed to have started.

```
FlurryWP8SDK.Api.SetReportDelay(int seconds)
```

If you want to delay the reporting of data from the beginning of a session, you can use this method to have the agent delay its

(C) Copyright Nokia 2014. All rights reserved.

report for this long.

There is also an API Class reference available.

Logging Exceptions and Stack Traces

To add logging of unhandled exceptions, add the following line of code to `Application_UnhandledException()` method of **App.xaml.cs**:

```
private void Application_UnhandledException ( object sender,
ApplicationUnhandledExceptionEventArgs e)
{
  if (System.Diagnostics.Debugger.IsAttached)
    System.Diagnostics.Debugger.Break ();

  FlurryWP8SDK.Api.LogError ( "Application_UnhandledException" , e.ExceptionObject);
}
```

Exception logging is very useful, but Stack Traces are even more valuable when tracking down a crash. To report a Stack Trace to Flurry, we can use a helper method which is called when there is an Exception. The helper method truncates the Stack Trace, as Flurry only allow strings of upto 255 chars, then reports the Exception and Stack Trace back to Flurry:

```
public static void LogStackTrace (Exception ex)
{
   string stacktrace = ex.StackTrace.Substring = (0, ex.StackTrace.Length> = 255 255:
ex.StackTrace.Length);
   FlurryWP7SDK.Api.LogError (stacktrace, ex);
}
```

Taken from Matthieu Vandenhende's blog

Windows Phone 7 support

By default, the Flurry website will supply the SDK that supports a Windows Phone 8 app.

If you're developing a Windows Phone 7 app, you can use the previous version: File:FlurryWP7SDK.zip

Summary

As you can see, it is very easy to integrate Flurry Analytics into your Windows Phone app.

The Flurry SDK allows for many different types of events to be captured (Timed events, Errors, Users details, Location) which can help you fine tune your app over time, removing or adjusting features based on their usage.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.