- Platform
- Collection
  - Web Tracking
  - Android SDK
    - Overview
    - Dev Guide
    - Reference
    - Resources
    - V1 (Legacy)
  - iOS SDK
  - Measurement Protocol
  - Legacy Libraries
- Configuration
- Reporting
- Solutions

Community & Updates

Developer Program

Google Analytics SDK for Android v2 (Beta) - Overview

The Google Analytics SDK for Android makes it easy for developers to collect user engagement data form their apps. This document will provide an overview of the value of the SDK as well as a guide to getting started measuring your app using a single property ID and EasyTracker.

> **Migrating from v1.x?** Check out our Migration Guide to get started using v2.

Introduction

The Google Analytics SDK for Android makes it easy for developers to collect user engagement data from their apps. Developers can then use Google Analytics reports to measure:

- The number of active users are using their applications.
- From where in the world the application is being used.
- Adoption and usage of specific features.
- In-app purchases and transactions.
- The number and type of application crashes.
- And many other useful metrics.

Additionally, the Google Analytics SDK for Android gives you the tools to monitor the success of mobile marketing campaign by providing end-to-end visibility into the performance of your marketing channels, from Google Play installs through in-app purchases and transactions.

Before you Begin

Before begin implementing the SDK, make sure you have the following:

- Android developer SDK (available for Windows, Mac OS X, and Linux)
- Google Analytics SDK for Android v2 (with `libGoogleAnalyticsV2.jar` included in your project's `/libs` directory and build path)
- An Android app that you can use to implement the Google Analytics SDK for Android v2
- A new Google Analytics app property and profile.

Getting Started

There are three steps to getting started with the SDK:

1. Update AndroidManifest.xml
2. Add EasyTracker methods
3. Create your analytics.xml file

After completing these steps, you'll be able to measure the following with Google Analytics:

- App installations
- Active users and demographics
- Screens and user engagement
- Crashes and exceptions

1. Updating AndroidManifest.xml

Update your AndroidManifest.xml file by adding the following permissions:

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

2. Adding EasyTracker methods

Add the send methods to the `onStart()` and `onStop()` methods of each of your `Activities` as in the following example:

```
/**
 * An example Activity in your app with Analytics
 * implemented.
 */
public class myTrackedActivity extends Activity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }

  @Override
  public void onStart() {
    super.onStart();
    ... // The rest of your onStart() code.
    EasyTracker.getInstance().activityStart(this); // Add this method.
  }

  @Override
  public void onStop() {
    super.onStop();
    ... // The rest of your onStop() code.
    EasyTracker.getInstance().activityStop(this); // Add this method.
  }
}
```

Note that EasyTracker requires `Context` before you can call its methods. In the above example, this line:

```
EasyTracker.getInstance.activityStart(this);
```

takes care of setting the context. However if you need to make EasyTracker calls in other classes or methods, you'll need to call EasyTracker's `setContext(Context ctx)` method first:

```
// Set Context before using EasyTracker. Note that the SDK will
// use the application context.
EasyTracker.getInstance().setContext(this);

// EasyTracker is now ready for use.
```

3. Creating your analytics.xml file

In version 2 of the Google Analytics SDK for Android, configuration options are managed from an xml resource file, called `analytics.xml` in this guide. You'll need to create this file in your project's `res/values` directory and add your tracking ID. The following example shows how you can add your tracking ID, and enable `Activity` and exception measurement:

```
<?xml version="1.0" encoding="utf-8" ?>

<resources>
  <!--Replace placeholder ID with your tracking ID-->
  <string name="ga_trackingId">UA-XXXX-Y</string>

  <!--Enable automatic activity tracking-->
  <bool name="ga_autoActivityTracking">true</bool>

  <!--Enable automatic exception tracking-->
  <bool name="ga_reportUncaughtExceptions">true</bool>
</resources>
```

Your lint checker may warn you about the use of the figure dash ('-') in your tracking ID. You can suppress that warning by adding additional attributes to your `<resources>` tag:

```
<resources xmlns:tools="http://schemas.android.com/tools"
tools:ignore="TypographyDashes">
```

Important: Do not encode dashes in the `ga_trackingId` string. Doing so will prevent you from seeing any data in your reports.

See the analytics.xml parameters reference for the complete list of parameters you can use to configure your implementation.

Congratulations! Your app is now setup to send data to Google Analytics.

Next steps

You can do much more with Google Analytics, including measuring campaigns, in-app payments and transactions, and user interaction events. See the following developer guides to learn how to add these features to your implementation:

- Advanced Configuration – Learn more about advanced configuration options, including using multiple trackers.
- Measuring Campaigns – Learn how to implement campaign measurement to understand which channels and campaigns are driving app installs.
- Measuring Events – Learn how to measure user engagement with interactive content like buttons, videos, and other media using Events.
- Measuring In-App Payments – Learn how to measure in-app payments and transactions.
- User timings – Learn how to measure user timings in your app to measure load times, engagement with media, and more.
- Analytics.xml parameters – See the complete list of `analytics.xml` configuration parameters.

Last updated April 16, 2013.