Quantcast Measurement API for Android

Implementation Guide

# Introduction

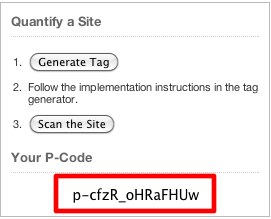
This guide is intended for developers who want to integrate the Quantcast Measurement API into their Android applications.

The guide assumes you have familiarity with the Android SDK and Eclipse. To use the Quantcast Measurement API, you must have these installed on your system. See <http://developer.android.com/sdk/index.html> for information on installing and updating these items.

The current version of the Quantcast SDK requires Android 2.0 or later.

# Obtaining a Publisher Code

Sign up for a Quantcast account at <http://www.quantcast.com>. Once you log into your account, find the “P-Code” assigned to you by Quantcast.



The P-Code is a string that identifies your data to Quantcast.

# Eclipse Project setup

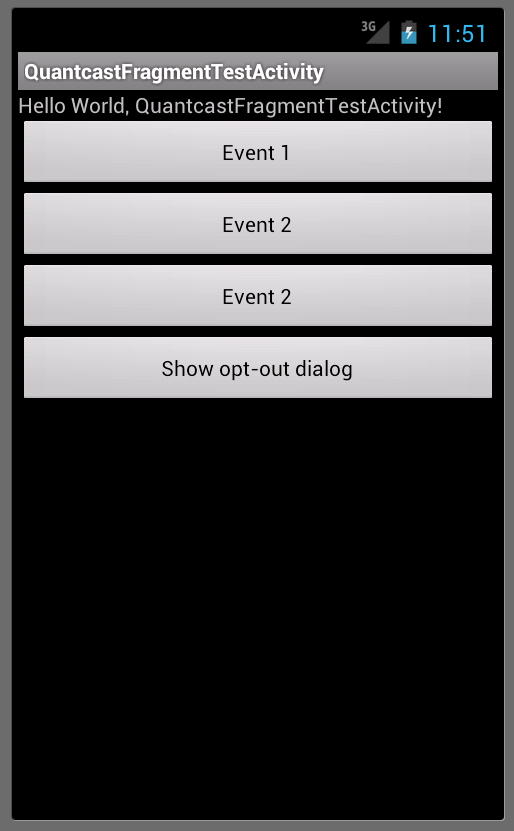
In Eclipse, import the QuantcastExampleApp project to your workspace. (Note: if the compiler complains about missing directories in any project add the needed directories)

# QuantcastExampleApp

You can run the QuantcastExampleApp in the Android emulator or on a connected Android device.

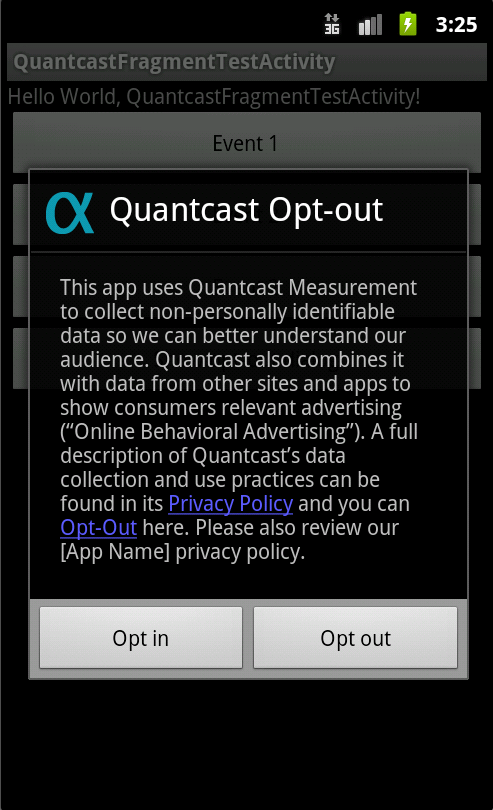
## Example App

You should see the following screen:



## Opt-out screen

The default collection policy is opt in. Your activity should have a button for users to opt out. The screen will ask you if you want to participate in the Quantcast service. If you press Opt out, data will not be collected or uploaded to Quantcast from your app. Press Opt in to proceed. Refer to *Show the Quantcast Collection Policy and Opt Out screen* for more information.



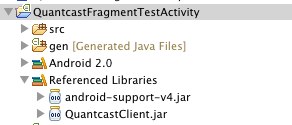
Pressing the buttons Event 1 or Event 2 logs a sample event.

Events are uploaded to Quantcast when a session begins or ends and on a timed pattern.

# Implementing the Quantcast Measurement API

To add the Quantcast API to your application, follow these steps:

Add QuantcastService.jar to your project. In Eclipse, create a folder named **libs** in your project. Add QuantcastService.jar to the **libs** folder. If your app uses an Android API version prior to Android 3.0 (API 11) you will also need to add android-support-v4.jar to the **libs** folder.



Right click on each jar file and select **Build Path** > **Add to Build Path**. These should now appear in Referenced Libraries in your project.

In your project AndroidManifest.xml, add declarations to enable the Quantcast Service as follows.

## Permissions

These permissions are placed inside the <manifest> tag. They allow the Quantcast service to collect and report events.

<!--

Permissions required for the Quantcast Measurement Service.

Copy these into your AndroidManifest.xml.

-->

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

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>

<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>

<!-- end of Quantcast permissions -->

## Activity

This activity should be added inside the <application> tag.

<activity android:name="com.quantcast.service.AboutQuantcastScreen" >

</activity>

## Set up your publisher code, application name, and version

In your main activity, import the Quantcast client:

import com.quantcast.service.QuantcastClient;

In your main activity, declare the publisher code, application name, and application name, and declare a field for the QuantcastClient:

private static final String PUBLISHER\_CODE = "your-p-code";

## Start a New Quantcast Measurement Session

As session must be started in order to log any other events. In your main activity’s onCreate() add a call to initiate a new Quantcast session:

QuantcastClient.beginSession(this, PUBLISHER\_CODE, "optional", “labels”);

## Show the About Quantcast Screen

You can show the About Quantcast Screen by calling:

QuantcastClient.showAboutQuantcastScreen(activity);

This allows the user to opt-out of the measurement service. When a user opts-out all save events are deleted.

## Get the Opt Out Status of the Service

Note: the SDK handles controlling event recording based on the user’s opt-out status. There is no need to check if collection is enabled before logging an event. This method is available if a change in app behavior based on opt-out status is desired.

You can check if collection is enabled for the Quantcast Service by calling:

QuantcastClient.isCollectionEnabled(this, new QuantcastClient.CollectionEnabledCallback() {

@Override

public void callback(boolean collectionEnabled) {

// do something with collectionEnabled

}

});

There is a possible delay in the availability of the user’s opt-out status so the status is communicated via a callback interface. The delay should be considered to be negligible in most situations.

If collection is not enabled the user has opted-out.

## Record a User Identifier

The user identifier is a means of identifying a user. It could be a user’s name or login, or a session id. Recording a new user id will start a new session and the same user identifier will be used for all further sessions during the applications lifetime until a new user id is recorded. The Quantcast Service will hash the user id will be hash the user id before uploaded.

To record a user identifier, invoke:

QuantcastClient.recordUserIdentifier(“some user id”);

This may be done at any time after a Quantcast Session has been initialized.

## Logging Pause

You should call this in your main activity’s onPause() method.

QuantcastClient.pauseSession("optional", “labels”);

## Logging Resume

You should call this in your main activity’s onResume() method.

QuantcastClient.pauseSession("optional", “labels”);

## Finishing a Quantcast Measurement Session

In your main activity’s onDestroy() method, call the following:

QuantcastClient.endSession("optional", “labels”);

This will clean up all of the service’s resources.

## Logging App-Defined Events

You may create your own app defined event by evoking:

QuantcastClient.logEvent("eventName", “optional", “labels”);

# Problems

## Debugging the Quantcast service

Check the Android logcat output, and look for q.\* tags.

You can set the log level of the service by calling:

QuantcastClient.setLogLevel(Log.VERBOSE);

The log level should be one of Log.VERBOSE, Log.DEBUG, Log.INFO, Log.WARN, Log.ERROR. The default log level for the Quantcast Service is Log.ERROR.