Quantcast Android SDK

This implementation guide provides steps for integrating the Quantcast Measure for Apps SDK, so you can take advantage of valuable, actionable insights:

- **Know Your Audience** - Quantcast uses direct measurement and machine learning to build accurate and detailed demographic profiles.
- **Compare and Compete** - Gauge user loyalty by analyzing visit frequency, retention and upgrades over time.
- **Attract Advertising** – Attract advertisers by showcasing your most powerful data points using a trusted source.
- **Improve Campaign Performance** – Increase your campaign performance by understanding characteristics of your best users and finding more people like them.

If you have any implementation questions, please email mobilesupport@quantcast.com. We're here to help.

Download the SDK

There are three ways to get the SDK. You can download it directly from the Quantcast website, you can use GitHub, or add a dependency via Gradle. If you download the file from our site, unzip the file before continuing to the section Integrate via External JAR.

Integrating Quantcast Measure for Mobile Apps

Integrate via Gradle

The Quantcast SDK for Android is now available via the JCenter. If your project supports the Gradle build system, this is the simplest solution. Just add the following line to your build.gradle file's dependencies section

```
compile &#39;com.quantcast.android.measurement:QuantcastAndroidSdk:1.4.+&#39;
```

Once completed you can skip to step 3 of the section SDK Integration.

Integrate via External JAR

Once you have the repository cloned, add the `QuantcastAndroidMeasurement.jar` within to your project by copying the file into your project's `libs/` directory. If you would like to keep the JAR external and are using Eclipse you can follow this guide.

Integrate via Source

You may also just drop the source files found in `QuantcastAndroidSdk/src/` directly into your project. This is as simple as moving the package directory into your projects `src/` folder. Unless you are using one of our more advanced feature (GPS, Network Measurement, etc.), you can ignore `QuantcastAndroidSdk/optional-s`

SDK Integration

1. In your project's `AndroidManifest.xml` you must ask for the required permissions by adding the following lines within the `<manifest>` tag before the `<application>` tag:

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

You can optionally add the following permissions to gather more information about your user base:

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

Finally to collect referrer data from the Google Play Store add the following lines within the `<applicati` tag:

```
<receiver android:name="com.quantcast.measurement.service.QCReferrerReceiver" android:
        <intent-filter>
            <action android:name="com.android.vending.INSTALL_REFERRER" />
        </intent-filter>
    </receiver>
```

2. Setup Android Advertising ID by including the Google Play Services 4.0+ into your project and add the following line to your project's `AndroidManifest.xml` as a child of the `<application>` tag: `xml` For additional information please see the More About Android Advertising ID section.

3. (Note: If you only support API Level 14 and above please see the One-Step Application Integration section) Import the `QuantcastClient` into ***every*** `Activity` in your project by adding the following import:

   ```
   import com.quantcast.measurement.service.QuantcastClient;
   ```

4. In the `onStart()` method of ***every*** `Activity` in your project, place the following to initialize the measurement service:

   ```
   QuantcastClient.activityStart(this, <*Insert your API Key Here*>, userIdentifier, segm
   ```

   Replace "<*Insert your API Key Here*>" with your Quantcast API Key, which can be generated in your Quantcast account homepage on the Quantcast website. The API Key is used as the basic reporting entity for Quantcast Measure. The same API Key can be used across multiple apps (i.e. AppName Free / AppName Paid) and/or app platforms (i.e. iOS / Android). For all apps under each unique API Key, Quantcast will report the aggregate audience among them all, and also identify/report on the individual app versions.

   The `userIdentifier` parameter is a `String` that uniquely identifies an individual user, such as an account login. This should not be confused with a unique device identifier. Passing this information allows Quantcast to provide reports on your combined audience across all your properties: online, mobile web and mobile app. This parameter may be `null` if your app does not have a user identifier. If the user identifier is not known at the time the `onStart()` method is called, the user identifier can be recorded at a later time. Please see the Combined Web/App Audiences section for more information.

   The `segments` parameter may be nil and is discussed in more detail in the Audience Labels section under Optional Code Integrations.

5. In the `onStop()` method of ***every*** `Activity` in your project place the following :

   ```
   QuantcastClient.activityStop();
   ```

(optional) More About Android Advertising ID

Android advertising ID is a user specific, unique, anonymous identifier that was recently released in October

2013 as a new Google service. Because it enables greater user privacy, Quantcast strongly recommend that it be used whenever available. Please follow the Google Play Setup Instructions to link to Google Play and enable the Quantcast SDK to use the advertising ID. On devices that support the advertising ID, the Quantcast SDK will no longer collect the Android device ID.

(optional) Understanding the API Key

The API key is used as the basic reporting entity for Quantcast Measure. The same API Key can be used across multiple apps (i.e. AppName Free / AppName Paid) and/or app platforms (i.e. iOS / Android). For all apps under each unique API Key, Quantcast will report the aggregate audience among them all, and also identify/report on the individual app versions.

(optional) One-Step Application Integration

Apps targeting Ice Cream Sandwich (API Level 14) and above can use a much simpler integration method by extending Android's `Application` class.

```
import com.quantcast.measurement.service.QuantcastClient;
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        QuantcastClient.startQuantcast(this, <*Insert your API Key Here*>, userIdentifier, 
    }
}
```

With this integration you do not have to explicitly call the `activityStart` and `activityStop` methods. You also need to be sure to register your new application class in your manifest file. This is done by adding `androi` to the application tag. For example

```
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:name=".MyApplication">
```

Compile and Test

You're now ready to test your integration. Build and run your project. Quantcast Measure will record activities and events from your emulator, as long as you quit your app properly (as opposed to closing the emulator window while the app is running). After finishing an app session, you will see your session recorded in your Quantcast Measure dashboard the following day. If you don't, you can refer to our troubleshooting guide for tips. Questions? Please email us at mobilesupport@quantcast.com.

Congratulations! Now that you've completed basic integration, explore how you can enable powerful features to understand your audience and track usage of your app.

- Read about User Privacy disclosure and options.
- Learn about Audience Segments, which you implement via Labels.
- If you have a web property, get a combined view of your mobile app and web audiences.
- Read about all the additional ways you can use the SDK, including Geo Location and Digital Magazine measurement.

User Privacy

Privacy Notification

Quantcast believes in informing users of how their data is being used. We recommend that you disclose in your privacy policy that you use Quantcast to understand your audiences. You may link to Quantcast's privacy policy: https://www.quantcast.com/privacy.

User Opt-Out

You can give users the option to opt out of Quantcast Measure by providing access to the About Quantcast Screen. This should be a button in your app's preferences `Activity` with the title "About Quantcast". When the user taps the button you provide, call `AboutQuantcastScreen` with the following:

```
QuantcastClient.showAboutQuantcastScreen(activity);
```

Also add the following lines within the `<application>` tag to allow the `AboutQuantcastScreen` to show:

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

Alternatively, if you would like to provide your own custom control over the Quantcast's opt out process, it is possible to set the opt out preference by setting the collection property directly instead of using the default dialog. For example:

```
QuantcastClient.setCollectionEnabled(false);
```

When not using the default dialog we strongly recommend that you also have a button to display Quantcast's Privacy Policy. You can display this by calling:

```
QuantcastClient.showQuantcastPrivacyPolicy(activity);
```

Note: when a user opts out of Quantcast Measure, the Quantcast Android SDK immediately stops transmitting information to or from the user's device and deletes any cached information that may have retained. Furthermore, when a user opts out of a single app on a device, the action affects all other apps on the device that are integrated with Quantcast Measure the next time they are launched.

Optional Code Integrations

Audience Labels

Use labels to create Audience Segments, or groups of users that share a common property or attribute. For instance, you can create an audience segment of users who purchase in your app. For each audience segment you create, Quantcast will track membership of the segment over time, and generate an audience report that includes their demographics. If you have implemented the same audience segments on your website(s), you will see a combined view of your web and app audiences for each audience segment. Learn more about how to use audience segments, including how to create segment hierarchies using the dot notation, here: https://www.quantcast.com/help/using-audience-segments.

There are two ways to assign labels. The first is via the `appLabels` setter. Use setAppLabels to record labels related to user properties. For example, to assign two labels, “purchaser.ebook” and “sharer.onFB”, you could do this:

```
QuantcastClient.setAppLabels("purchaser.ebook", "sharer.onFB");
```

Using this has the effect of passing these labels with every method call of the Quantcast SDK. At any time however, you can temporarily add to the labels you’ve assigned using `appLabels` by setting the `labels:` argument in your Quantcast method call.

Here is an example that adds the label “sharer.firstShare” in addition to the labels you’ve already assigned (“sharer.onFB”, “purchaser.ebook”) via the `appLabels` property. This example uses the `logEvent:withLabels` method, which you can learn about under Tracking App Events.

```
String additionalLabel = "sharer.firstShare";
String theEventStr = "tweeted";
QuantcastClient.logEvent(theEventStr, additionalLabel);
```

All labels that are set during the course of an app session will register a visit for that app session, but only labels set via setAppLabels will persist across sessions. A session is started when an app is launched, or when it is woken from the background after more than 30 minutes. A session is defined as ended when an app is closed or when a new session starts. In the example above, the session will register a visit on audience segments: “sharer.onFB”, “purchaser.ebook”, and “sharer.firstShare”. If the app is then suspended for more than 30 minutes, then awakened, our servers will record a new app session. If no additional calls are made to QuantcastMeasurement, only the segments assigned via setAppLabels, “sharer.onFB” and “purchaser.ebook”, will register a visit for that session.

While there is no specific constraint on the intended use of the label dimension, it is not recommended that you use it to indicate discrete events; in these cases, use the `logEvent` method described under Tracking App Events.

Tracking App Events

Quantcast Measure can be used to measure audiences that engage in certain activities within your app. To log the occurrence of an app event or activity, call the following method:

```
QuantcastClient.logEvent(eventName);
```

`eventName` is the `String` that is associated with the event you are logging. Hierarchical information can be indicated by using a left-to-right notation with a period as a separator. For example, logging one event named "button.left" and another named "button.right" will create three reportable items in Quantcast Measure: "button.left", "button.right", and "button". There is no limit on the cardinality that this hierarchal scheme can create, though low-frequency events may not have an audience report on due to the lack of a statistically significant population.

Geo-Location Measurement

Change: The geo-location library has be moved starting in version 1.1.0 to the optional-src directory in order to remove any LocationManager code from applications that do not use it. In order to add geolocation either add the QCLocation class into the src folder or add optional-src as another source location in your project. This file is automatically included when using the jar or gradle integration methods.

To get geo-location aware reporting, turn on geo-tracking in the `onStart()` method of every `Activity` in your project before you call `activityStart()` with the following:

```
QCLocation.setEnableLocationGathering(true);
```

You also must add either of the following permissions to gather more information about your user base:

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

Note: only enable geo-tracking if your app has some location-aware purpose.

Combined Web/App Audiences

Quantcast Measure enables you to measure your combined web and mobile app audiences, allowing you to understand the differences and similarities of your online and mobile app audiences, or even the combined audiences of your different apps. To enable this feature, you will need to provide a user identifier, which Quantcast will always anonymize with a 1-way hash before it is transmitted from the user's device. This user identifier should also be provided for your website(s); please see Quantcast's web measurement documentation for instructions.

Normally, your app user identifier would be provided in the `onStart()` method of any `Activity` of your project via the `QuantcastClient.startActivity()` method as described in the Required Code Integration section above. If the app's active user identifier changes later in the app's life cycle, you can update the user identifier using the following method call:

```
QuantcastClient.recordUserIdentifier(userIdentifier);
```

The `userIdentifier` parameter is a `String` containing the user identifier.

Note that in all cases, the Quantcast Android SDK will immediately 1-way hash the passed app user identifier, and return the hashed value for your reference. You do not need to take any action with the hashed value.

De-Duplicating Web Traffic

Sometimes you might want to direct a user to a webpage within your mobile application. If the webpage already tagged with a Quantcast Web Tag, then that user may mistakenly be counted as a mobile app user as well as a mobile web user. In order to get the best possible measurement on all platforms, you should get all your WebViews using this call:

```
QuantcastClient.newDeduplicatedWebView(this);
```

This will return a Webview specially tagged to stop duplications. If you need to extend a webview you can also extend `QCDeduplicatedWebView` instead of a normal WebView.

Digital Magazines and Periodicals

Quantcast Measure provides measurement features specific to digital magazines and periodicals. These options allow the measurement of specific issues, articles and pages in addition to the general measurement of the app hosting the magazine. In order to take advantage of this measurement, you must at a minimum tag when a particular issue has been opened and closed and when each page in that issue has been viewed (in addition to the basic SDK integration). You may also optionally tag when a particular article has been viewed. For more information, please refer to the documentation in the source file which can be found in the SDK source folder at optional-src/QCPeriodical.java.

Network/Platform Measurement

This feature should only be used by app networks, most notably app platforms, app development shops, and companies with a large number of branded apps where they want to maintain the app's brand when quantifying but still have the app traffic attributed to a parent network. Entities quantifying a single app or a number of apps under the same network should not use this feature. The Networks extension adds the ability to identify a parent network, referred to as an "attributed network", for each app in addition to or instead of the app's API Key. For more information, please refer to the documentation in the source file which can be found in the SDK source folder at optional-src/QCNetworkMeasurement.java.

Measuring Directly-Served Ad Campaigns

For apps that serve advertising and can access advertiser and campaign identifiers from their ad serving system, Quantcast can measure the audience exposed to these campaigns. If you want to additionally log the ad displays that occur within your app and have audience measurement against the exposed audience, first you must first add the optional source found in `optional-src/QCAdvertising.java`. Then add the log ad impression methods when advertisements are shown or refreshed:

```
QCAdvertising.logAdImpression(inCampaignOrNull, inMediaOrNull, inPlacementOrNull, appLabels
```

Where inCampaignOrNull is a String of the campaign identifier being displayed for the ad impression, inMediaOrNull is an String of the ad creative identifier being displayed, and inPlacementOrNull is a String of the placement identifier for the location. Note that the Campaign, Media and Placement strings are all optional, and also that any periods in their name will be treated like a label period, indicating the level of hierarchy.

You may also pass a dynamic audience label here. In this case, the label passed in the app labels argument will place the device user who saw the ad impression into the indicated audience segment. You might use this the ad impression label to categorize the type of ad product being displayed so that you can get aggregate reports on audience exposure. See Audience Labels section above for more information on Audience Segments.

SDK Customization

Logging and Debugging

You may enable logging within the Quantcast Android SDK for debugging purposes. By default, logging is turned off. To enable logging, set the log level of the Quantcast Android SDK by calling:

```
QuantcastClient.enableLogging(true);
```

Everything logged by the Quantcast Android SDK will have a tag beginning with "q.".

Event Upload Frequency

The Quantcast Android SDK will upload the events it collects to Quantcast's server periodically. Uploads that occur too often will drain the device's battery. Uploads that don't occur often enough will cause significant delays in Quantcast receiving the data needed for analysis and reporting. By default, these uploads occur when at least 100 events have been collected or when your application pauses (that is, it switched into the background). You can alter this default behavior via `QuantcastClient.setUploadEventCount()`. For example, if you wish to upload your app's events after 20 events have been collected, you would make the following call:

```
QuantcastClient.setUploadEventCount(20)
```

You may change this property multiple times throughout your app's execution.

Secure Data Uploads

The Quantcast Android SDK can support secure data uploads using SSL/TLS. In order to enable secure data uploads you must make the following call:

```
QuantcastClient.setUsingSecureConnections(true);
```

Note that using secure data uploads causes your app to use encryption technology. Various jurisdictions have laws controlling the export of software applications that use encryption. Please review your jurisdiction's laws concerning exporting software that uses encryption before enabling secure data uploads in the Quantcast Android SDK.

Trouble Shooting

Little or No App Traffic Showing Up In App's Profile On Quantcast.com Quantcast updates its website with your app's latest audience measurement data daily. If even after 1 day no data is showing up in your app's profile on quantcast.com, please check the following: * The Quantcast SDK does most of its data uploading when your app is transitioned to the background. If during your development and testing workflow in Xcode you regularly end a test run of your app by pressing "stop" or closing the emulator, your app has not necessarily had a chance to upload usage data. To ensure your app gets a chance to upload usage data to Quantcast while you are testing, be sure to click the Home or Back button on the device being tested in order to put your app into the background and thus trigger a usage data upload to Quantcast. * Still having weird data issues, make sure that you have both the 'activityStart' and 'activityStop' in every one of your application's Activities (not needed in Fragments). These calls help the SDK keep track of the lifecycle of the entire application. * If you encounter trouble with your build, please review the documentation for the project setup and SDK integration. The most common errors involve missing one of the steps outlined. If you have are still having trouble, please email mobilesupport@quantcast.com.

License