Quantcast Measure for PhoneGap (Cordova) SDK

Thank you for downloading the Quantcast PhoneGap SDK! This SDK lets you measure your PhoneGap app on either iOS or Android using Quantcast Measure for Mobile Apps. This implementation guide provides steps for integrating the SDK, so you can take advantage of valuable, actionable insights:

- **Traffic Stats & Return Usage** - see your audience visitation trends
- **Combined Web / App Audiences** - understand your aggregate audience across all screens by taking a few additional steps outlined in this guide.
- **Labels** – segment your traffic by customizing views within your app.

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

For help creating a PhoneGap project, see PhoneGap Documentation

Integrating Quantcast Measure for PhoneGap

Project Setup

To integrate Quantcast Measure into your PhoneGap project, you will to add both the Quantcast Measure PhoneGap plugin and the appropriate native Quantcast Measure for Mobiles Apps SDK to your project.

Begin by cloning the Quantcast PhoneGap SDK's git repository. Open the Terminal application in your Mac and issue the following commands:

```
git clone https://github.com/quantcast/phonegap-measurement.git ./quantcast-phonegap-sdk
```

Once you have downloaded the Quantcast PhoneGap SDK's code, perform the following steps:

1. Determine which platform, iOS or Android, you need and copy the files found in the appropiate directory with the Quantcast Measure for Phonegap SDKinto your projects "Plugins" directory, then add the source code to the project.
    1. **iOS** projects will need to add the files found under the `iOS-plugin` folder in the Quantcast Measure for Phonegap SDK
    2. **Android** projects will need to add the files found the `android-plugin` folder in the Quantcast Measure for Phonegap SDK

2. Copy QuantcastMeasurement.js to the www/js folder in your PhoneGap project.

3. In config.xml, add `xml <plugin name="QuantcastMeasurementPlugin" value="QuantcastMeasu` under the plugins tag.

4. Include the platform specific Quantcast Measure SDK to your project. Note that you do not have to do the code integration indicated in the native SDK documentation. You just need to follow the **Project Setup** instructions for the SDK.
    1. The iOS SDK can be found here.
    2. The Android SDK can be found here.

See Quantcast Measure for Mobile for more information on registering your app with Quantcast.

Required Code Integration

You have an option to use either the Consolidated Setup or the Detailed Setup when performing the required code integration to set up Quantcast Measure in your PhoneGap project.

Consolidated Setup Integration

The consolidated method can be used for simpler implementations of the Quantcast SDK. Projects that use constant or no Event Labels will benefit the most. This method automatically sets up the pause/resume/end methods for you so at a minimum this will be the only call you need to make. You are still free to use the optional features of Quantcast Measure, such asTracking App Events.

1. In the 'deviceready' event listener usually found in the index.js file, we can begin the measurement session by calling the following:

   ```
   QuantcastMeasurement.setUpQuantcastMeasurement("<*Insert your API Key Here*>", userIde
   ```

   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 userIdentifierStrOrNull parameter is a string that uniquely identifies an individual user, such as an account login. 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 available at the time your app launches. If the user identifier is not known at the time the 'deviceready' event is called, the user identifier can be recorded at a later time. Please see the Combined Web/App Audiences section for more information.

   The last parameter may be null and is discussed in more detail in the Event Labels section under Optional Code Integrations.

Detailed Code Integration

If you need more control over pause and resume events to accomplish things such as setting specific labels, then you will have to a couple more simple steps.

1. In the 'deviceready' event listener usually found in the index.js file, we can begin the measurement session by calling the following:

   ```
   QuantcastMeasurement.beginMeasurementSession( callbackOrNull, "<*Insert your API Key H
   ```

   The first parameter is a function callback to receive your userIdentifier one-way hash. It is returned for your reference so in most cases this can be null.

   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 userIdentifierStrOrNull parameter is a string that uniquely identifies an individual user, such as an account login. 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 available at the time your app launches. If the user identifier is not known at the time the 'deviceready' event is called, the user identifier can be recorded at a later time. If a user identifier is given, we will create a 1-way hash with the value and return it back via the callbackOrNull argument. Please see the Combined Web/App Audiences section for more information.

The last parameter may be null and is discussed in more detail in the Event Labels section under Optional Code Integrations.

2. We are also going to need to add additional event listeners for the 'pause' and 'resume' events. This is usually done at the top of the 'deviceready' event listener by adding the following lines:

```
document.addEventListener(&#39;pause&#39;, app.onPause, false);
document.addEventListener(&#39;resume&#39;, app.onResume, false);
```

Remember that the scope of "this" is the event in an event listener and not the object that contains it. So above (and in the default PhoneGap template) we had to explicitly call "app. ".

3. We also need to handle these new events by adding the following functions to index.js

```
onPause: function() {
   QuantcastMeasurement.pauseMeasurementSession(labelsOrNull);
},
onResume: function() {
    QuantcastMeasurement.resumeMeasurementSession(labelsOrNull);
},
```

Again the parameter may be null and is discussed in more detail in the Event Labels section under Optional Code Integrations.

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 here.

User Opt-Out

You can give users the option to opt out of Quantcast Measure by providing access to the Quantcast Measure Opt-Out dialog. This should be accomplished with a button or a table view cell (if your options are based on a grouped table view) in your app's options view with the title "Measurement Options" or "Privacy". When a user taps the button you provide, call the Quantcast’s Opt-Out dialog using the following method:

```
QuantcastMeasurement.displayUserPrivacyDialog(callbackOrNull);
```

The parameter is an optional function callback that returns a boolean if the privacy settings have changed. An example callback function might look like this:

```
optOutCallback: function(optedOut){
    alert("Quantcast Opt out status changed to: \r\n"+optedOut );
}
```

Note: when a user opts out of Quantcast Measure, the 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 any 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

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:

```
QuantcastMeasurement.logEvent(theEventStr, labelsOrNull);
```

theEventStr 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 due to the lack of a statistically significant population.

Event Labels

Most of Quantcast SDK's public methods have an option to provide one or more labels, or null if no label is desired. A label is any arbitrary string that you want associated with an event. The label will create a second dimension in Quantcast Measure audience reporting. Normally, this dimension is a "user class" indicator. For example, you could use one of two labels in your app: one for users who have not purchased an app upgrade, and one for users who have purchased an upgrade.

The labels argument of most Quantcast SDK methods can either be a single string ("label") or multiple string in an Array (["label1", "label2"]) 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 function described under Tracking App Events.

Geo-Location Measurement

To turn on geo-tracking, insert the following call into your 'deviceready' event listener after you call either form of the beginMeasurementSession methods:

```
QuantcastMeasurement.setGeoLocation(true);
```

Note that you should only enable geo-tracking if your app has some location-aware purpose.

The Quantcast SDK will automatically pause geo-tracking while your app is in the background. This is done for both battery life and privacy considerations.

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 your 'deviceready' event listener method via the beginMeasurementSessionWithAPIKey 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:

```
QuantcastMeasurement.recordUserIdentifier(callbackOrNull, userIdentifierStr, labelsOrNull);
```

The current user identifier is passed in the userIdentifierStr argument.

Note that in all cases, the Quantcast iOS SDK will immediately 1-way hash the passed app user identifier, and return the hashed value for your reference via the callbackOrNull function. You do not need to take any action with the hashed value and can set the callback as null if desired. The callback function will have one argument which is the 1-way hash value. For example:

```
hashcallback: function(hash){
    alert("This is my hash: \r\n"+hash );
}
```

SDK Customization

Logging and Debugging

You may enable logging within the Quantcast PhoneGap Plugin for debugging purposes. By default, logging is turned off. To enable logging, call the following method at any time, including prior to calling the beginMeasurementSession:

```
QuantcastMeasurement.setDebugLogging(true);
```

You should not release an app with logging enabled.

Event Upload Frequency

The Quantcast PhoneGap Plugin 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 by setting the uploadEventCount property. For example, if you wish to upload your app's events after 20 events have been collected, you would make the following call:

```
QuantcastMeasurement.uploadEventCount(20);
```

You may change this property multiple times throughout your app's execution and this number must be greater than 1.

License

to the License and (2) are in compliance with the License. See the License for the specific language governing permissions and limitations under the License.