Quantcast iOS SDK

This implementation guide provides steps for integrating the Quantcast 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.

Integrating Quantcast for Mobile Apps

To integrate Quantcast's SDK into your iOS app, you must use Xcode 4.5 or later. The Quantcast SDK supports apps built for iOS 7.0 and later.

Download the SDK

There are two ways to get the SDK. You can download it directly from [the Quantcast website] (https://www.quantcast.com/user/quantcast-app-measurement-sdk.zip "Quantcast Measure for Apps SDK"), or you can use GitHub. If you download the file from our site, unzip the file before continuing to the section Set Up Your Xcode Project (#set-up-your-xcode-project).

(optional) Getting the Quantcast SDK from GitHub

If you use Git for your version control, we recommend that you make the Quantcast SDK repository a submodule in your project's Git repository. To do so, open the Terminal application in your Mac, cd into your Git repository folder, and issue the following commands:

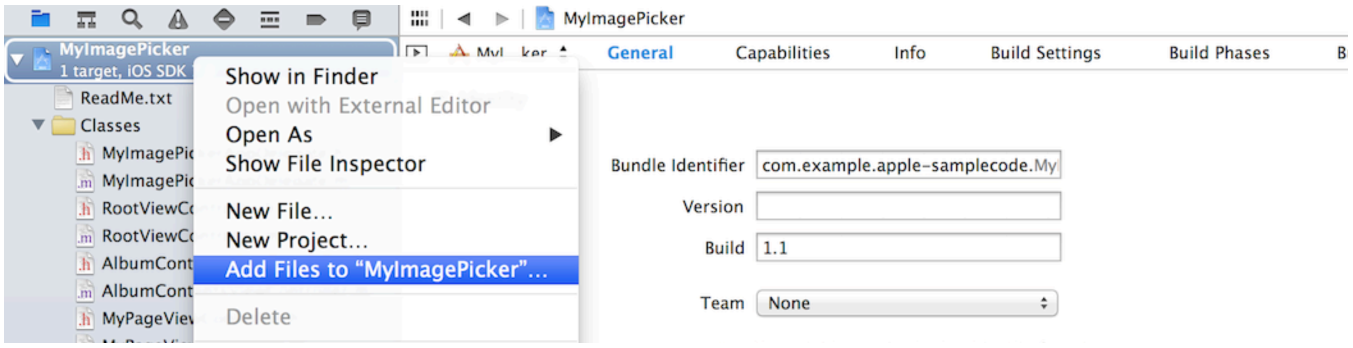
```
git submodule add https://github.com/quantcast/ios-measurement.git
git submodule update --init
cd ios-measurement
git submodule update --init
```

If you don't want to make the Quantcast SDK repository a submodule in your repository, you can use Git to clone the Quantcast iOS SDK's Git repository and initialize all of its submodules.

```
git clone https://github.com/quantcast/ios-measurement.git ./quantcast-ios-sdk
cd ./quantcast-ios-sdk/
git submodule update --init
```

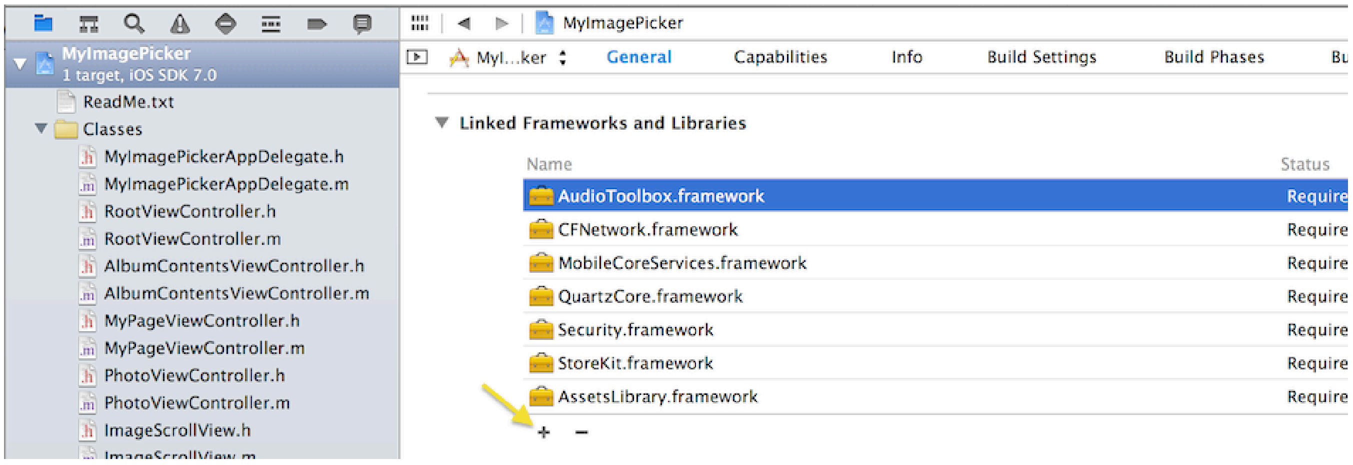
Set Up Your Xcode Project

1. Import the Quantcast SDK code into your project. In Xcode, select your project and choose the option "Add Files to ", then select the Quantcast-iOS-Measurement folder from your download.



Bundle Identifier: com.example.apple-samplecode.My
Version:
Build: 1.1
Team: None

2. Link the following iOS frameworks and libraries to your project if they are not already. From your project properties, go to the "General" section, the scroll down to "Linked Frameworks and Libraries" and hit the "+" at the bottom left. Then use the Command key to multiselect and add the following:
    1. `CoreGraphics`
    2. `CoreTelephony`
    3. `Foundation`
    4. `libsqlite3.dylib`
    5. `libz.dylib`
    6. `Security`
    7. `SystemConfiguration`
    8. `UIKit`



Name	Status
AudioToolbox.framework	Require
CFNetwork.framework	Require
MobileCoreServices.framework	Require
QuartzCore.framework	Require
Security.framework	Require
StoreKit.framework	Require
AssetsLibrary.framework	Require

Use of the AdSupport Framework

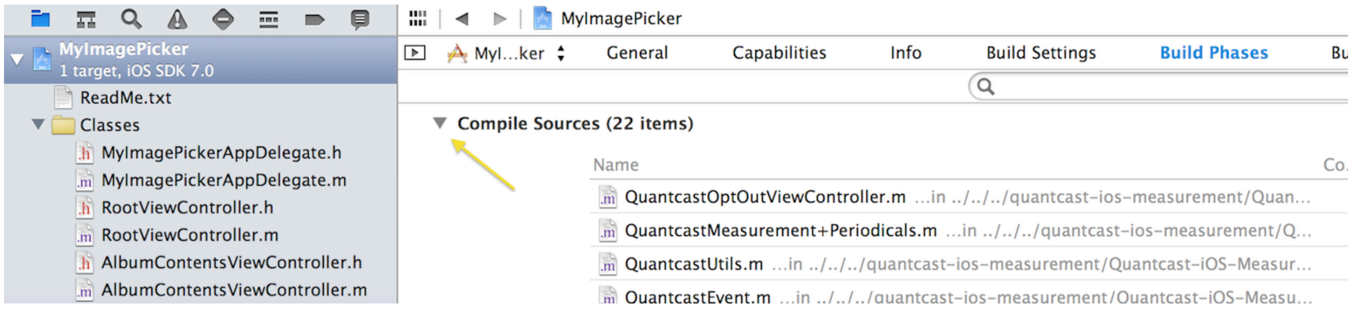
Linking to the AdSupport framework is not strictly necessary for integration with the Quantcast SDK. However, if you choose to link to the AdSupport framework in order to serve ads in your app or track installation attribution, that framework will allow Quantcast to have to access to IDFAs, which we can use to improve the quality of demographic data we provide about your users in our Measure service.

Remember, if your code or any third party code in your app utilizes the IDFA, you will need to state that in the app submission process, in response to the question "Does this app use the Advertising Identifier (IDFA)?" Thus, if you're integrated with the Quantcast SDK and you choose to link to the Ad Support framework, you should answer that question "yes." Additionally, in the app submission process, you will need to confirm that your app – and everyone that interfaces with your app – honors the Limit Ad Tracking setting, which the Quantcast SDK does.

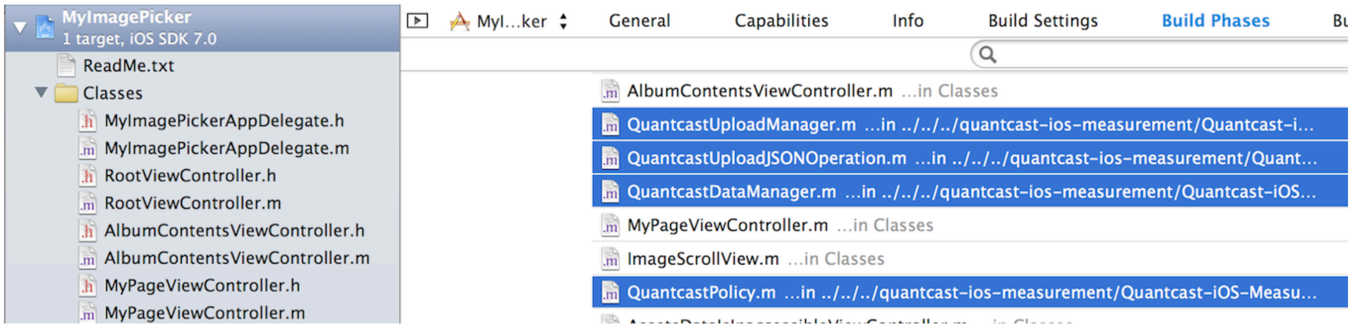
If You Are Not Using Automatic Reference Counting (ARC)

If your project does not use automatic reference counting (ARC), take the following steps to set a compiler flag for Quantcast source files. Otherwise, skip to the next section.

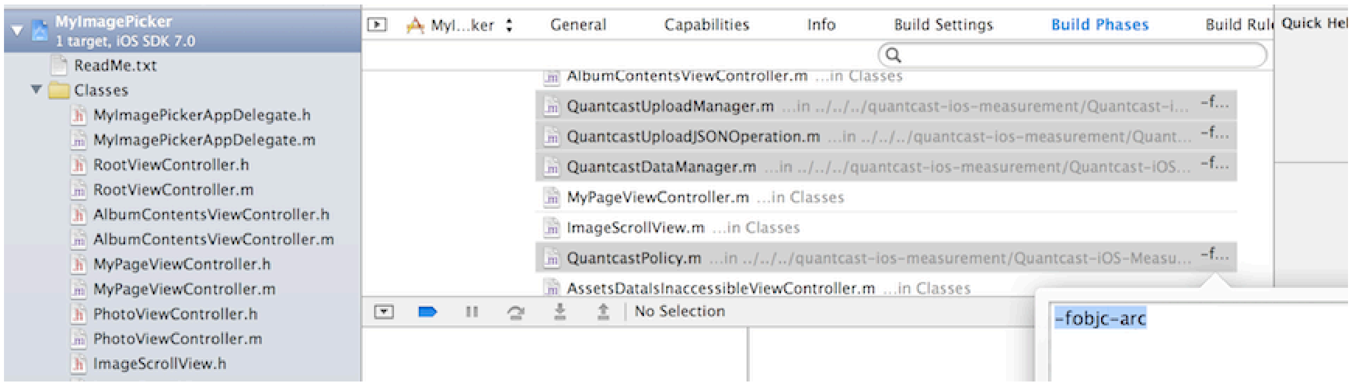
1. In your project configuration screen, click on the “Build Phases” section, then expand “Compile Sources”.


MyImagePicker
1 target, iOS SDK 7.0
ReadMe.txt
Classes
MyImagePickerAppDelegate.h
MyImagePickerAppDelegate.m
RootViewController.h
RootViewController.m
AlbumContentsViewController.h
AlbumContentsViewController.m
MyI…ker
General
Capabilities
Info
Build Settings
Build Phases
Compile Sources (22 items)
Name
QuantcastOptOutViewController.m …in ../../../quantcast-ios-measurement/Quan…
QuantcastMeasurement+Periodicals.m …in ../../../quantcast-ios-measurement/Q…
QuantcastUtils.m …in ../../../quantcast-ios-measurement/Quantcast-iOS-Measur…
QuantcastEvent.m …in ../../../quantcast-ios-measurement/Quantcast-iOS-Measu…

2. Multi-select every Quantcast source file by holding down the Command button and choosing every filename that begins with “Quantcast”.


MyImagePicker
1 target, iOS SDK 7.0
ReadMe.txt
Classes
MyImagePickerAppDelegate.h
MyImagePickerAppDelegate.m
RootViewController.h
RootViewController.m
AlbumContentsViewController.h
AlbumContentsViewController.m
MyPageViewController.h
MyPageViewController.m
MyI…ker
General
Capabilities
Info
Build Settings
Build Phases
AlbumContentsViewController.m …in Classes
QuantcastUploadManager.m …in ../../../quantcast-ios-measurement/Quantcast-i…
QuantcastUploadJSONOperation.m …in ../../../quantcast-ios-measurement/Quant…
QuantcastDataManager.m …in ../../../quantcast-ios-measurement/Quantcast-iOS…
MyPageViewController.m …in Classes
ImageScrollView.m …in Classes
QuantcastPolicy.m …in ../../../quantcast-ios-measurement/Quantcast-iOS-Measu…

3. Hit enter to bring up a text input box, then type in “-fobjc-arc” and hit enter.

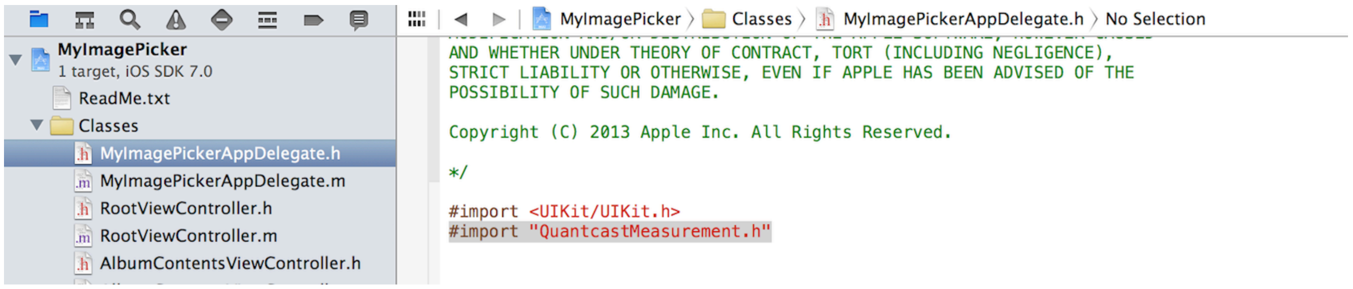

MyImagePicker
1 target, iOS SDK 7.0
ReadMe.txt
Classes
MyImagePickerAppDelegate.h
MyImagePickerAppDelegate.m
RootViewController.h
RootViewController.m
AlbumContentsViewController.h
AlbumContentsViewController.m
MyPageViewController.h
MyPageViewController.m
PhotoViewController.h
PhotoViewController.m
ImageScrollView.h
MyI…ker
General
Capabilities
Info
Build Settings
Build Phases
Build Rul
Quick Hel
AlbumContentsViewController.m …in Classes
QuantcastUploadManager.m …in ../../../quantcast-ios-measurement/Quantcast-i… -f…
QuantcastUploadJSONOperation.m …in ../../../quantcast-ios-measurement/Quant… -f…
QuantcastDataManager.m …in ../../../quantcast-ios-measurement/Quantcast-iOS… -f…
MyPageViewController.m …in Classes
ImageScrollView.m …in Classes
QuantcastPolicy.m …in ../../../quantcast-ios-measurement/Quantcast-iOS-Measu… -f…
AssetsDataIsInaccessibleViewController.m …in Classes
No Selection
-fobjc-arc

SDK Integration

The recommended way to integrate the Quantcast SDK requires only a single line of code:

1. Import `QuantcastMeasurement.h` into your `UIApplication` delegate class

```
#import "QuantcastMeasurement.h"
```


MyImagePicker 〉 Classes 〉 MyImagePickerAppDelegate.h 〉 No Selection
MyImagePicker
1 target, iOS SDK 7.0
ReadMe.txt
Classes
MyImagePickerAppDelegate.h
MyImagePickerAppDelegate.m
RootViewController.h
RootViewController.m
AlbumContentsViewController.h

```
AND WHETHER UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE),
STRICT LIABILITY OR OTHERWISE, EVEN IF APPLE HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 2013 Apple Inc. All Rights Reserved.

*/

#import <UIKit/UIKit.h>
#import "QuantcastMeasurement.h"
```

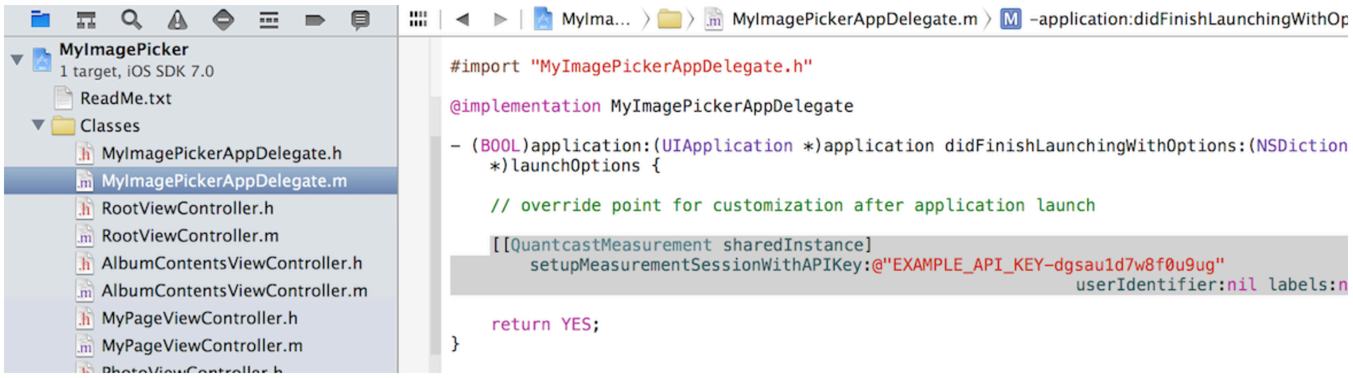
2. In your `UIApplication` delegate's `application:didFinishLaunchingWithOptions:` method, place the following:

   ```
   [[QuantcastMeasurement sharedInstance] setupMeasurementSessionWithAPIKey:@"<Insert you
   userIdentifier:nil labels:nil];
   ```

   Replace "<*Insert your API Key Here*>" with your Quantcast API Key. The API Key can be found in the file “api-key.txt” in your Quantcast SDK folder. All your API keys can also be found on your Quantcast dashboard: [https://www.quantcast.com/user/resources?listtype=apps] (https://www.quantcast.com/user/resources?listtype=apps). For more information about how and when to use the API Key, read [Understanding the API Key] (#optional-understanding-the-api-key).

   The `userIdentifier:` parameter accepts 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. Please see the Combined Web/App Audiences section for more information.

   The `labels:` parameter may be nil and is used to create Audience Segments. Learn more in the Audience Labels section.


MyIma... 〉 〉 MyImagePickerAppDelegate.m 〉 M -application:didFinishLaunchingWithOp
MyImagePicker
1 target, iOS SDK 7.0
ReadMe.txt
Classes
MyImagePickerAppDelegate.h
MyImagePickerAppDelegate.m
RootViewController.h
RootViewController.m
AlbumContentsViewController.h
AlbumContentsViewController.m
MyPageViewController.h
MyPageViewController.m

```
#import "MyImagePickerAppDelegate.h"

@implementation MyImagePickerAppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDiction
    *)launchOptions {

    // override point for customization after application launch

    [[QuantcastMeasurement sharedInstance]
        setupMeasurementSessionWithAPIKey:@"EXAMPLE_API_KEY-dgsau1d7w8f0u9ug"
                                                         userIdentifier:nil labels:n

    return YES;
}
```

(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.

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 iOS 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 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 default Opt-Out dialog using the following method:

```
[[QuantcastMeasurement sharedInstance] displayUserPrivacyDialogOver:currentViewController w
```

The `currentViewController` argument is the current view controller. The SDK needs to know this due to how the iOS SDK presents modal dialogs (see Apple's documentation for `presentViewController:animated:`). The delegate is an optional parameter and is explained in the `QuantcastOptOutDelegate` protocol header.

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 isOptedOut property directly instead of using the default dialog. For example:

```
[QuantcastMeasurement sharedInstance].isOptedOut = YES;
```

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:

```
[[QuantcastMeasurement sharedInstance] displayQuantcastPrivacyPolicy];
```

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.

Optional Code Integrations

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 `UIApplication` delegate's `application:didFin` method via the `beginMeasurementSessionWithAPIKey:userIdentifier:labels:` 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 sharedInstance] recordUserIdentifier:userIdentifierStr withLabels:ni
```

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. You do not need to take any action with the hashed value.

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] (https://www.quantcast.com/help/using-audience-segments).

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

```
NSArray *myUserSegmentMembership = @[@"purchaser.ebook",@"sharer.onFB"];
[QuantcastMeasurement sharedInstance].appLabels = myUserSegmentMembership;
```

Setting the `appLabels` property 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.

```
NSString *newLabel = @"sharer.firstShare";
NSString *theEventStr = @"tweeted";
[[QuantcastMeasurement sharedInstance] logEvent:theEventStr withLabels:newLabel];
```

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 the `appLabels` property 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 the `appLabels` property, “sharer.onFB” and “purchaser.ebook”, will register a visit for that session.

The `labels:` argument of most Quantcast SDK methods is typed to be an `id` pointer. However, it only accepts either a `NSString` object representing a single label, or a `NSArray` object containing one or more `NSStr` objects representing a collection of labels to be applied to the event.

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:withLabels:` 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:

```
[[QuantcastMeasurement sharedInstance] logEvent:theEventStr withLabels:nil];
```

`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.

Geo-Location Measurement

To turn on geo-location measurement, please take the following steps:

1. Link your project to the `CoreLocation` framework
2. Ensure that the `QuantcastGeoManager.m` compile unit , which can be found in the `Optional` folder of the SDK, has been added to your project.
3. Add the following line to your project's pre-compiled header file:

```
objective-c    #define QCMEASUREMENT_ENABLE_GEOMEASUREMENT 1
```

1. Insert the following call into your `UIApplication` delegate's `application:didFinishLaunchingWithOp` method after you call either form of the `beginMeasurementSession:` methods:

```
objective-c    [QuantcastMeasurement sharedInstance].geoLocationEnabled = YES;
```

You may also safely change the state of the `geoLocationEnabled` at any point after your app has launched. The Quantcast SDK will always adhere to its current setting.

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

The Quantcast iOS SDK will automatically pause geo-tracking while your app is in the background.

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 Periodicals header file which can be found in the SDK source folder at `Optional/QuantcastMeasurement+Periodicals.h`.

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/QuantcastMeasurement+Advertising.h`. Then add the log ad impression methods when advertisements are shown or refreshed:

```
[[QuantcastMeasurement sharedInstance] logAdImpressionForCampaign:inCampaignOrNil media:inM
```

Where inCampaignOrNil is an NSString of the campaign identifier being displayed for the ad impression, inMediaOrNil is an NSString of the ad creative identifier being displayed, and inPlacementOrNil is a NSString 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 iOS SDK for debugging purposes. By default, logging is turned off. To enable logging, call the following method at any time, including prior to calling either of the `beg` methods:

```
[QuantcastMeasurement sharedInstance].enableLogging = YES;
```

You should not release an app with logging enabled.

Event Upload Frequency

The Quantcast iOS 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 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 sharedInstance].uploadEventCount = 20;
```

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

Secure Data Uploads

As of version 1.5.0, secure downloads are turned on by default. This was changed due to Apple's new App Transport Security update The Quantcast iOS SDK can still support non-https calls if that is preferred. In order to disable secure data uploads, add following preprocessor macro definition to your project's precompiled header file (the file that ends with '.pch'):

```
#define QCMEASUREMENT_USE_SECURE_CONNECTIONS 0
```

Note that if you turn off secure connections, then you will need to set up the proper exceptions using the `NSApp` key in you info.plist. The `NSExceptionDomains` key should be set to `quantcount.com` and included subdirectories `NSIncludesSubdomains`. Please read more about setting up exceptions here

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" within Xcode, 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 button on the device being tested in order to put your app into the background and thus trigger a usage data upload to Quantcast. * 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