

iOS SDK v1.2.0 Documentation

Getting Started 3

1 Introduction 3

2 Prerequisites 3

3 Downloading the SDKs 3

4 Sample Projects 3

5 Core Concepts 4

5.1 Clients 4

5.2 Consumers 4

5.3 Offer 4

5.4 Geo-fence 5

5.5 Beacon 5

5.6 Passbook 5

Setting Up the SDK 6

1 Overview 6

2 Files Included in the SDK 6

3 Add the SDK Files to Your Existing App 7

4 Configure Preferences (PList File) 9

4.1 Koupon Media Entries 9

4.2 Gimbal Entries (if Applicable) 9

4.3 Optional Entries 9

4.4 Gimbal PList (if Applicable) 10

5 Update the App Delegate 11

5.1 Registering a Consumer 11

5.2 Registering for Push Notifications 12

5.3 Setting Up Advanced Push 12

5.4 Adding Gimbal to the App Delegate 12

5.5 Advanced Gimbal Approach 13

Tutorial 14

1 Quick Start (Standard SDK Use) 15

1.1 Register (or Re-register) a Consumer 15

1.2 Retrieve the Consumer’s Offer List 15

1.3 Mark an Offer as Viewed 16

1.4 Mark an Offer as Presented 16

2 Using the SDK as a URL Generator 16

Click on the numbers to the right to be taken directly to that page

Getting Started

# Introduction

The Koupon Media iOS SDK v1.2.0 contains the most up-to-date frameworks for integrating the Koupon Media platform into your own iOS applications. The Koupon Media SDK includes iOS libraries, developer documentation and a sample Xcode project to get you up and running quickly and easily.

Additionally, Koupon Media has partnered with Qualcomm for the purpose of leveraging the GimbalTM context-aware platform and its Location-based Services (LBS). This document includes instructions on augmenting the Koupon Media SDK with the Gimbal SDK so that you can enhance your applications with geo-fence monitoring, as well as content delivery triggered by geo-fence events and/or time proximity.

# Prerequisites

Before you get started with the Koupon Media SDK, let’s confirm that your system is set up correctly. Ensure that you are…

* Running OS X
* Using [Xcode](https://developer.apple.com/xcode/) 4.6+
* Targeting iOS 5.0+

# Downloading the SDKs

If you have not already downloaded the Koupon Media SDK (archive), click the button below to download it from the web **(NOTE: Downloads can take 15-45 seconds as the files are large).**

[Download Koupon Media iOS SDK v1.2.0](https://kouponmedia.s3.amazonaws.com/sdks/iOS/Koupon_SDK_v1.2.0.zip)

If you have purchased the LBS component, you can also download the Gimbal SDK (archive) from the internet.

[Download Gimbal iOS SDK v1.16.1](https://kouopnmedia.s3.amazonaws.com/docs/Quick%20Start%20Guide/SDKs/iOS/iOS_Gimbal_1.16.1.zip)

# Sample Projects

Want to see how the Koupon Media SDK works before you integrate it with an existing app? No problem!

We’ve created a sample project using the Koupon Media SDK so that you can see all the libraries and presets you need to get an iOS app running in Xcode. **(NOTE: Downloads can take 15-45 seconds as the files are large).**

[Download “KMSDK Sample” Xcode Project](https://kouponmedia.s3.amazonaws.com/sdks/iOS/SampleCode/Koupon%20SDK%20Sample_v1.2.0.zip)

If you’re ready to code, but want to start from scratch, we’ve made it really easy for you. Our starter Xcode project is empty except for the Koupon Media SDK that’s already been loaded into it.

[Download Blank Xcode Project w/ SDK](https://kouponmedia.s3.amazonaws.com/sdks/iOS/SampleCode/EmptyKouponProject%20v1.2.0.zip)

# Core Concepts

Koupon Media is a cloud-based platform built on the following service components:

* The **Koupon Manager**, a Content Management System (CMS) that supports the creation, management and distribution of mobile offers.
* **Push notifications**, a service that supports sending data and content to mobile application users via the Apple Push Notification Service (APNS).
* **Location-based services** (enabled by the Gimbal SDK), which engage a user based on his or her location.
* **Digital Wallet Manager**, a service that supports designing and distributing content to Apple Passbook.

We recommended familiarizing yourself with the terms below, as you will see them used throughout the documentation.

## Clients

“Clients” are approved organizations that have been provisioned with access to the Koupon Manager (based upon their contracted services options). Clients are provided with an AuthKey and AuthSecret, which are used as credentials to make API requests.

## Consumers

“Consumers” are the recipients of a Client’s offer. Apps developed using the Koupon Media SDK can register a consumer’s identity whenever they are launched.

## Offer

The Koupon Media SDK interfaces with the Koupon Manager and supports offer lifecycle management across the following states:

* Accessed – Offer is accessed and presented within a list.
* Viewed – Offer details such as terms and conditions.
* Presented – Redemption assets delivered to the consumer.

## Geo-fence

A Geo-fence is a virtual fence around a physical location (latitude and longitude) that, in conjunction with a geo-fence enabled mobile application, communicates arrival, departure and dwell times from the app-defined locations. Please refer to the documentation that is included with the Gimbal SDK for additional information.

## Beacon

Proximity beacons leverage the Bluetooth Low Energy (BLE) standard to derive a consumers’ location at a micro-level. When a consumer’s device is within physical proximity to the beacon and detects it, a mobile application can notify the consumer of location-relevant content, promotions and offers. Please refer to the documentation that is included with the Gimbal SDK for additional information.

## Passbook

Apple Passbook is an iOS application that functions as a container for defined-pass templates, including offers, tickets and loyalty cards.

Setting Up the SDK

|  |  |
| --- | --- |
| ! | **Are you starting your iOS app from scratch?** You can [download our starter Xcode project](https://kouponmedia.s3.amazonaws.com/sdks/iOS/SampleCode/EmptyKouponProject%20v1.2.0.zip), which is empty except for the pre-loaded Koupon Media SDK. |

# Overview

The purpose of this section is to explain the components of the Koupon Media SDK and to help you configure it for use with an existing iOS app. We assume that you have already downloaded the [Koupon Media SDK](https://kouponmedia.s3.amazonaws.com/sdks/iOS/Koupon_SDK_v1.2.0.zip) (or [our starter Xcode project](https://kouponmedia.s3.amazonaws.com/sdks/iOS/SampleCode/EmptyKouponProject%20v1.2.0.zip) if you do not have an existing app), and if applicable, the [Gimble SDK](https://kouopnmedia.s3.amazonaws.com/docs/Quick%20Start%20Guide/SDKs/iOS/iOS_Gimbal_1.16.1.zip).

# Files Included in the SDK

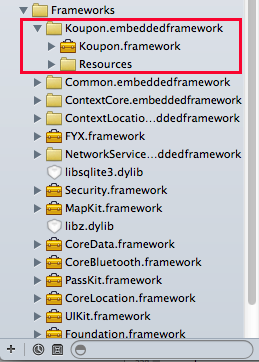
The following framework and header files are included in the Koupon Media SDK archive Koupon.embeddedframework:

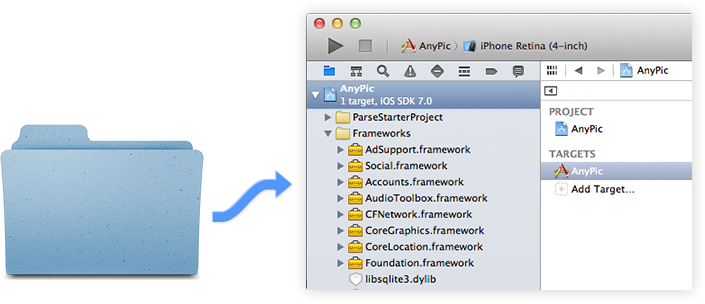
|  |  |
| --- | --- |
| **File Name** | **Description** |
| KouponResources.bundle | Contains all of the resources for Advanced Push. |
| KouponGimbalManager.h | A bridge for linking the KouponMedia SDK to the Gimbal SDK. |
| Koupon.h | The main Koupon object that houses all the functions for interacting with our APIs |
| KMRequest.h | The request object that holds all details for an API request. Details include the signed URL, API body (if necessary), and several shortcut functions for executing the request. |
| KMLocation.h | All the location services for the SDK. |
| KMProximityEvent.h | Returned from any Beacon Proximity events and contains proximity specific information. |
| KMGeofence.h | Returned from any Geofence events and contains geofence specific information. |

Note that the SDK archive also contains documentation.

# Add the SDK Files to Your Existing App

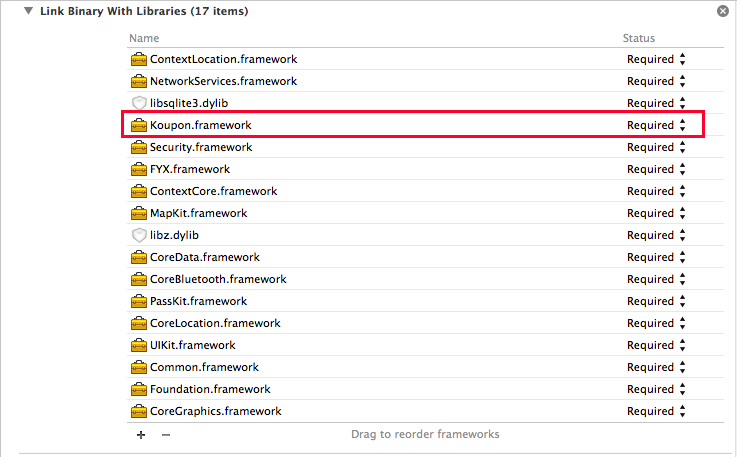
To add the Koupon Media SDK files to your existing Xcode project, unzip the archive and follow the steps below.

1. Drag Koupon.embeddedframework into your Xcode project’s **Frameworks** folder (be sure to select “**Copy the items to your destination’s group folder**”).

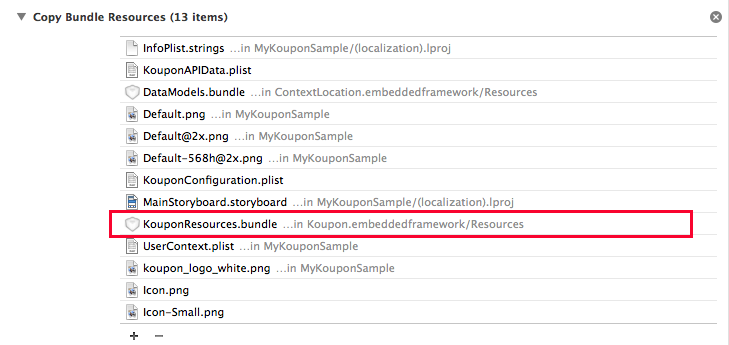


If you are also using the Gimbal SDK, refer to the documentation included in the Gimbal SDK package for instructions on adding frameworks.

1. Within Xcode, add the dependencies to your project by navigating to the screen **Targets > [*Your project name]* > Build Phases**. Here, expand the section entitled “**Link Binary with Libraries**.” If you are only using the Koupon Media SDK, verify that Koupon.framework is within the dependency list.



1. Expand the section entitled “Copy Bundle Resources” and verify KouponResources.bundle is included



If you are also using the Gimbal SDK, you need to include the following dependencies (in addition to Koupon.embeddedframework):

* Common.embeddedframework
* ContextCore.embeddedframework
* ContextLocation.embeddedframework
* ContextProfiling.embeddedframework
* FYX.framework
* NetworkServices.embeddedframework

# Configure Preferences (PList File)

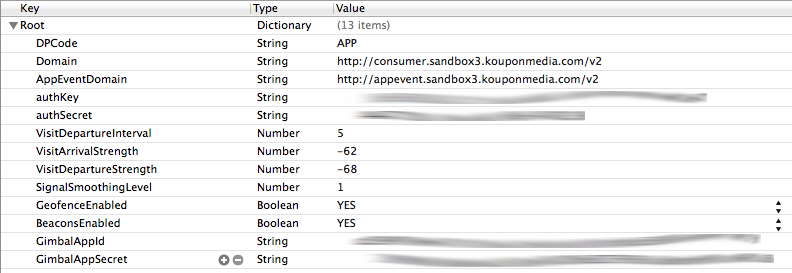
To configure the Koupon Media SDK preferences, create a file named KouponConfiguration.plist and add the entries below.

## Koupon Media Entries

|  |  |
| --- | --- |
| **Entry** | **Description** |
| DPCode | APP – The value of “APP” ensures that your app only pulls in offers that are destined to go inside of the app digital property |
| authKey | Your Koupon Media authKey |
| authSecret | Your Koupon Media authSecret |

## Gimbal Entries (if Applicable)

|  |  |
| --- | --- |
| **Entry** | **Description** |
| GeofenceEnabled | YES or NO |
| BeaconsEnabled | YES or NO |
| GimbalAppId | Your Gimbal App Id (provided by Koupon Media) |
| GimbalAppSecret | Your Gimbal App Secret (provided by Koupon Media) |



## Optional Entries

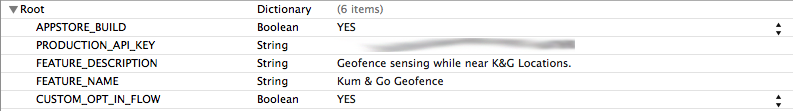
|  |  |
| --- | --- |
| **Entry** | **Value** |
| Domain | The base URL for Koupon API calls. You should only need to specify this if you are pointing to a non-production environment. Please refer to the note that follows this table. |
| AppEventDomain | The URL for AppEvent calls. You should only need to specify this if you are pointing to a non-production environment. Please refer to the note that follows this table. |
| VisitDepartureInterval | The number of seconds that a consumer needs to be away from a beacon in order to register a return. |
| VisitArrivalStrength | The radio signal decibel level that defines when a consumer is near a beacon. A reasonable range is -90 to -25 dB. |
| VisitDepatureStrength | The radio signal decibel level that defines when a consumer is sufficiently away from a beacon. A reasonable range is -90 to -25 dB. |
| SignalSmoothingLevel | Since the radio signal can be erratic, a smoothing function can be applied to eliminate discontinuities in the signal. |

|  |  |
| --- | --- |
| ! | The Koupon Media SDK supports production and non-production environments. During the integration and testing phases of application development, it is necessary to ensure the proper endpoint URL is used for the intended environment. If this value is not set, then the SDK will default to using the production environment. |

## Gimbal PList (if Applicable)

The Gimbal SDK requires another PList file named UserContext.plist with the following configuration entries:

|  |  |
| --- | --- |
| **Entry** | **Description** |
| FEATURE\_DESCRIPTION | Human readable description of what sort of events you will be tracking |
| FEATURE\_NAME | Broad title representing your app |
| PRODUCTION\_API\_KEY | Your Gimbal Production API Key (provided by Koupon Media) |
| APPSTORE\_BUILD | Always set to YES |
| CUSTOM\_OPT\_IN\_FLOW | Always set to YES |



# Update the App Delegate

## Registering a Consumer

There are several options for registering a consumer. Please refer to the use cases below and add the corresponding code to application.didFinishLaunchingWithOptions. Your AppDelegate.m will look similar to the following sample code:

// AppDelegate.m

// MyKouponSample

#import "AppDelegate.h"

#import <Koupon/Koupon.h>

@implementation AppDelegate

@synthesize apiData**;**

@synthesize kouponSDK**;**

@synthesize window**;**

@synthesize deviceTokenData**;**

**-** **(**BOOL**)**application**:(**UIApplication **\*)**application didFinishLaunchingWithOptions**:(**NSDictionary **\*)**launchOptions

**{**

//Init the Koupon SDK using the KouponConfiguration.plist file

kouponSDK **=** **[[**Koupon alloc**]** init**];**

**[**kouponSDK createConsumerIdentity**:[**NSMutableDictionary dictionary**]** withOptionalQueryData**:nil];**

1. You can register a consumer with an external Consumer ID (CID), such as a CRM identifier from a retailer’s IT system.

// Register a consumer using “CID”

NSMutableDictionary **\***consumerData **=** **[[**NSMutableDictionary alloc **]** init**];**

**[**consumerData setObject**:@”123456”** forKey**:**@“cid”**];**

1. You can register a consumer with an email address or mobile number (most likely obtained from a log-in page).

// Register a consumer using “email” or “mobile”

NSMutableDictionary **\***consumerData **=** **[[**NSMutableDictionary alloc **]** init**];**

**[**consumerData setObject**:** @“emailAddress@domain**.**com” forKey**:**@“email”**];**

1. You can register a consumer using all three identifiers (i.e., CID, email address and mobile number).

// Register a consumer using all three: “cid”, “email”,and “mobile”

NSDictionary **\***consumerdata **=** **[**NSDictionary dictionaryWithObjectsAndKeys**:**

**@"5555555555",** **@"mobile",**

**@"email@place.com",** **@"email"**

**@"123456",@"cid",** **nil];**

## Registering for Push Notifications

If you would like your application to register with Koupon Media as a Third Party Push Providers, add the following code to the App Delegate, under application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*)deviceToken

**[[**kouponSDK reportAppEvent**:@"pushreg"** withData**:[**NSMutableDictionary dictionaryWithObjectsAndKeys**:**deviceToken**,** **@"device\_id",** **@"ios",** **@"platform",** **nil]]**

sendRequestAsyncOnSuccess**:^(**NSMutableDictionary**\*** response**){**

NSLog**(@"MySampleApp->Successfully sent Device Token to Koupon Media”);**

**}**

**OnFailure:^(NSMutableDictionary\* response){**

**NSLog(@"**MySampleApp**->**Failed to send Device Token to Koupon Media");

**}];**

Note that the example above demonstrates how to asynchronously perform a network request using the Koupon Media SDK. For more information on setting up a third party push provider, please see the [official Apple Developer Documentation](https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html).

## Setting Up Advanced Push

Advanced push is a feature that allows you to

Make sure to add #import <Koupon/KMAdvancedPush> inside your appdelegate.

**-** **(**void**)**application**:(**UIApplication **\*)**application didReceiveRemoteNotification**:(**NSDictionary **\*)**userInfo

**{**

**[[**KMAdvancedPush shared**]** handleAdvancedPush**:userInfo];**

Somewhere inside your application didFinishLaunchingWithOptions: method, add the following.

**-** **(**BOOL**)**application**:(**UIApplication **\*)**application didFinishLaunchingWithOptions**:(**NSDictionary **\*)**launchOptions

**{**

**[[**KMAdvancedPush shared**]** handleAdvancedPush**:launchOptions];**

## Adding Gimbal to the App Delegate

|  |  |
| --- | --- |
| ! | Be sure to add the Gimbal SDK files, link the libraries, and prepare the PList entries described above. Also, you must include the following files in your project:   * KouponGimbalManager.h * KMProximityEvent.h * KMGeofenceEvent.h |

Update AppDelegate.h to include the following code:

#import <Koupon/KouponGimbalManager.h>  
#import <Koupon/KMProximityEvent.h>  
#import <Koupon/KMGeofenceEvent.h>

@interface KouponAppDelegate **:** NSObject **<**UIApplicationDelegate**,** KouponGimbalManagerDelegate**>**

Update the App Delegate function of application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions to include the following:

// Inside of ApplicationDidStartWithOptions

**[[**KouponGimbalManager sharedManager**]** initGimbalServicesWithDelegate**:self** KouponSDK**:**kouponSDK **];**

Note that the code above could also be implemented in another class delegate that inherits from the KouponGimbalManger. Your application can now respond to various events by implementing one or more of the following delegate methods:

**-** **(**void**)** beaconServiceStarted **{**

**[[**KouponGimbalManager sharedManager**]** startListeningForVisits**];**

**}**

**-** **(**void**)** beaconServiceFailedToStart**:** **(**NSError **\*)** error **{**

NSLog**(@"handle error");**

**}**

**-** **(**void**)** receivedSighting**:** **(**KMProximityEvent**\*)** proxEvent **{**

NSLog**(@"Sighting: %@ ---- RSSI: %@",** proxEvent**.**visit**.**transmitter**.**name**,** proxEvent**.**RSSI**);**

**}**

**-** **(**void**)** receivedArrival**:** **(**KMProximityEvent**\*)** proxEvent **{**

NSLog**(@"Arrival: %@",** proxEvent**.**visit**.**transmitter**.**name**);**

**}**

**-** **(**void**)** receivedDeparture**:** **(**KMProximityEvent**\*)** proxEvent **{**

NSLog**(@"Departure: %@ ---- dwellTime: %f",** proxEvent**.**visit**.**transmitter**.**name**,** proxEvent**.**visit**.**dwellTime**);**

**}**

**-** **(**void**)** didEnterGeofence**:** **(**KMGeofenceEvent**\*)** geoEvent**{**

NSLog**(@"You have entered %@.",** geoEvent**.**placeEvent**.**place**.**name**);**

**}**

**-** **(**void**)** didExitGeofence**:(**KMGeofenceEvent **\*)** geoEvent **{**

NSLog**(@"You have left %@",** geoEvent**.**placeEvent**.**place**.**name**);**

**}**

## Advanced Gimbal Approach

A more advanced approach utilizes a publish/subscribe model in lieu of the delegate functions, which allows multiple parts of your application to have access to your location events.

For this use case, KouponGimbalManager broadcasts/publishes events through the NSNotificationCenter, thereby notifying your app where controllers subscribe to notifications through the Notification Center.

The approach below synchronizes information across many parts of your application, rather than limiting it to a particular delegate for handling the business logic. The following code provides a brief sample of how to do that using several possible notifications.

//First, you need to subscribe to the notification

**[[**NSNotificationCenter defaultCenter**]** addObserver**:self** selector**:**@selector**(**didEnterGeofence**:)** name**:**KMGimbalManagerDidEnterGeofenceNotification object**:nil];**

**[[**NSNotificationCenter defaultCenter**]** addObserver**:self** selector**:**@selector**(**didExitGeofence**:)** name**:**KMGimbalManagerDidExitGeofenceNotification object**:nil];**

//Now you can pull the geoEvent from the NSNotification object

**-** **(**void**)** didEnterGeofence**:** **(**NSNotification**\*)** notification**{**

KMGeofenceEvent **\***geoEvent **=** **[**notification object**];**

NSLog**(@"You have entered %@.",** geoEvent**.**placeEvent**.**place**.**name**);**

**}**

**-** **(**void**)** didExitGeofence**:(**NSNotification **\*)** notification **{**

KMGeofenceEvent **\***geoEvent **=** **[**notification object**];**

NSLog**(@"You have left %@",** geoEvent**.**placeEvent**.**place**.**name**);**

**}**

Other possible notifications include the following:

* KMGimbalManagerBeaconServiceStartedNotification
* KMGimbalManagerDidExitGeofenceNotification
* KMGimbalManagerDidEnterGeofenceNotification
* KMGimbalManagerReceivedDepartureNotification
* KMGimbalManagerReceivedArrivalNotification
* KMGimbalManagerReceivedSightingNotification
* KMGimbalManagerBeaconServiceFailedToStartNotification

Tutorial

|  |  |
| --- | --- |
| ! | The Koupon Media SDK can act in two modes: (1) a mode that acts as signed URL generator and (2) a mode that allows the SDK to issue network calls on the developer’s behalf. Allowing the SDK to issue the network request often simplifies the logic for getting started. However, more advanced applications may want to manage batched network calls; and the signed URL generator supports these types of apps as well. |

# Quick Start (Standard SDK Use)

|  |  |
| --- | --- |
| ! | **Are you starting your iOS app from scratch?** If you want to save time configuring your new project, you can [download our starter Xcode project](https://kouponmedia.s3.amazonaws.com/sdks/iOS/SampleCode/EmptyKouponProject%20v1.2.0.zip), which is empty except for the pre-loaded Koupon Media SDK. |

Once you have configured your application for the Koupon Media SDK, the typical flow of calling SDK functions follows the pattern below.

1. [Register (or re-register) a consumer.](#_Register_(or_Re-register))
2. [Retrieve the consumer’s offer list.](#_Retrieve_the_Consumer’s)
3. [Mark offers as “Viewed.”](#_Mark_an_Offer)
4. [Mark offers as “Presented.”](#_Mark_an_Offer_1)

## Register (or Re-register) a Consumer

The code for registering a consumer was introduced in the section “Registering a Consumer,” above. To reiterate, update AppDelegate.m to register the consumer with a CID, email address, mobile number or all three identifiers.

// AppDelegate.m

// MyKouponSample

#import "AppDelegate.h"

@implementation AppDelegate

@synthesize apiData**;**

@synthesize kouponSDK**;**

@synthesize window**;**

@synthesize deviceTokenData**;**

**-** **(**BOOL**)**application**:(**UIApplication **\*)**application didFinishLaunchingWithOptions**:(**NSDictionary **\*)**launchOptions

**{**

//Init the Koupon SDK using the KouponConfiguration.plist file

kouponSDK **=** **[[**Koupon alloc**]** init**];**

**[**kouponSDK createConsumerIdentity**:[**NSMutableDictionary dictionary**]** withOptionalQueryData**:nil];**

## Retrieve the Consumer’s Offer List

To retrieve a list of offers for the consumer, call the Koupon Media SDK as follows:

// Create a dictionary of filters to pass into

// the GetConsumerOfferList function. See the SDK API documentation

// for a detailed list of filters or visit the Consumer API documentation.

KMRequest **\***request **=** **[**kouponSDK getConsumerOfferList**:**nil withOptionalQueryData**:nil];**

**[**request sendRequestAsyncOnSuccess**:^(**NSMutableDictionary **\*** response**)** **{**

NSData **\***jsonData**;**

NSString **\***jsonString**;**

**if(**response**)** **{**

jsonData **=** **[**NSJSONSerialization dataWithJSONObject**:**response options**:**0 error**:nil];**

**}**

**if(**jsonData**)** **{**

jsonString **=** **[[**NSString alloc**]** initWithBytes**:[**jsonData bytes**]** length**:[**jsonData length**]** encoding**:**NSUTF8StringEncoding**];**

**}**

NSLog**(@"The response is an NSDictionary containing the JSON response of the Consumer API. You could convert this dictionary to a JSON object: %@",** jsonString**);**

**}** OnFailure**:^(**NSMutableDictionary **\*** response**)** **{**

**}];**

## Mark an Offer as Viewed

To mark a consumer’s offers as “viewed,” which helps the Koupon Media system learn about consumer preferences based on what the consumer has or has not viewed, call the Koupon Media SDK as follows:

//PromoId is an integer representing the promotion you are trying to update.

KMRequest **\***kmRedeemRequest **=** **[**kouponSDK updateConsumerState**:@"VIEWED"** forOffer**:[**NSString stringWithFormat**:@"%d",**promoId**]** withOptionalData**:nil];**

**[**kmRedeemRequest sendRequestAsyncOnSuccess**:^(**NSMutableDictionary **\***viewedResp**)** **{**

//The "VIEWED" state returns

//See the Consumer API documentation for a complete list of all the available properties of this dictionary.

**}** OnFailure**:^(**NSMutableDictionary **\***viewedResp**)** **{**

//Handle Failure

**}];**

## Mark an Offer as Presented

Another common task for the Koupon Media SDK is to mark offers as used or “presented.” Note that offers can be configured as single-use or multi-use. When a consumer’s application marks a single-use offer as “presented” (i.e. the barcode has been presented to the screen for scanning), the offer will be removed from his or her offer list. Conversely, multi-use offers will be counted as used, but continue to appear in the offer list until the offer expires.

The following code block demonstrates how to use the Koupon Media SDK to issue a network call for marking an offerID as “presented.”

//PromoId is an integer representing the promotion you are trying to update.

KMRequest **\***kmRedeemRequest **=** **[**kouponSDK updateConsumerState**:@"PRESENTED"** forOffer**:[**NSString stringWithFormat**:@"%d",**promoId**]** withOptionalData**:nil];**

**[**kmRedeemRequest sendRequestAsyncOnSuccess**:^(**NSMutableDictionary **\***redeemResp**)** **{**

//Handle Success

//The "PRESENTED" state returns all of the redemption assets

//See the Consumer API documentation for a complete list of all the available properties of this dictionary.

**}** OnFailure**:^(**NSMutableDictionary **\***redeemResp**)** **{**

//Handle Failure

**}];**

# Using the SDK as a URL Generator

The Koupon Media SDK can work within existing network calling mechanics, thus you may find it helpful to utilize the SDK as a URL generator. The signed URLs generated by the SDK can be requested via your own scheme, which is demonstrated by the following code:

NSString **\***signedUrl **=** **[[**kouponSDK getConsumerOfferList**:nil** withOptionalQueryData**:nil]** url**];**

NSLog**(@"Generated url from Koupon Media SDK: %@",** signedUrl**);**

// Request this URL from a preferred network request mechanism