

# Footmarks iOS SDKv3 Detailed Overview

**Patent Notice**: Various use cases and technical information provided herein are protected by a series of provisional and non-provisional patent applications.

Copyright © 2013 Footmarks, Inc. – All Rights Reserved Worldwide – Patent Pending

This document may not be reproduced without the expressed permission of Footmarks, Inc.

Any reproduction, without authorization, is an infringement of Footmarks copyright.

This document contains confidential and proprietary information.

Table of Contents

Footmarks iOS SDKv3 Detailed Overview 1

Document Overview 3

Handling Beacon & Experience Responses 4

FMBeaconManager Delegate Methods 4

ExperienceManager Delegate Methods 5

Experience Properties 6

Experience 6

VideoExp 7

ImageExp 9

AlertExp 10

CustomExp 11

# Document Overview

This document details the Footmarks SDK’s methods, properties, callbacks and experience response structure. Use this document to get an in-depth understanding on how to best utilize the Footmarks SDK in conjunction with the Footmarks Cloud Service.

# Handling Beacon & Experience Responses

## FMBeaconManager Delegate Methods

Below is a list of the beacon delegate methods defined in FMBeaconManagerDelegate.

- (void)beaconManager:(FMBeaconManager \*)manager

didRangeBeacons:(NSArray \*)beacons inRegion:(FMBeaconRegion \*)region;

**Description**: Returns all Footmarks beacons in range of the user.

-(void)beaconManager:(FMBeaconManager \*)manager

didEnterRegion:(FMBeaconRegion \*)region;

**Description**: Invoked when a user walks within range of a new Footmarks beacon.

**Important**: This is one of 2 methods that gets invoked even when a user does not have your app running and comes within range of a Footmarks beacon. Thus, it is extremely powerful. If you push a notification and the user clicks it, then your app starts running. If they do not click on the notification, you get approximately 5 seconds of processing time before your app goes back to not running again.

-(void)beaconManager:(FMBeaconManager \*)manager

didExitRegion:(FMBeaconRegion \*)region;

**Description**: Invoked when a user walks out of a Footmarks beacon’s broadcast range. Usually there is about a 30 second delay to determine that the user is completely out of range and not hovering on the beacon’s boundary.

**Important**: This is the second method that gets triggered even if your app is not running.

- (void) bluetoothTurnedOff;

**Description**: When a user’s Bluetooth is disabled, this method will get invoked. This is a good place to present an alert to request that the user enables Bluetooth.

- (void) locationServicesFailedWithError: (NSError \*)error;

**Description**: When a user’s Location Services is disabled entirely or disabled for your app, this method will get invoked.

## ExperienceManager Delegate Methods

Below is the method that gets invoked when the Footmarks Cloud service returns an experience in response to some user action (e.g. entering a beacon’s proximity).

- (void) didCompleteExperiences: (NSArray\*) experiences;

**Description**: Returns one or more Experience objects. Typically, 1 Experience will be returned. Video, Image & Alert Experiences will be returned as instances of their respective classes. In turn, you will not have to parse them. Custom experiences defined in our Cloud Service, will be returned as a CustomExp object, which will contain the raw JSON data. The various Experience objects and their respective fields can be found in the “Footmarks\_SDK\_VX.h” header file.

# Experience Properties

All experiences inherit from the base class, Experience. Currently, there are 4 experience types: Video, Image, Alert and Custom. Experiences are created in the Footmarks Management Console. The properties for the Experience base class and each subclass are described below.

## Experience

@property ExperienceType type: Enum that indicates the given Experience type

typedef enum : int

{

ExperienceTypeCustom = 0,

ExperienceTypeVideo = 1,

ExperienceTypeImage = 2,

ExperienceTypeAlert = 3

} ExperienceType;

@property ExperienceAction action: Enum that instructs how to act on the given experience.

typedef enum : int

{

// Do not show the Experience content to the user. Depending on

// your implementation, it may make sense to store the Experience

// and display it sometime in the future

ExperienceActionDoNothing = 0,

// Display the data contained in the Experience right away without

// giving the user a choice. For example, if a VideoExp arrived,

// it would auto play the next time the user opened the app.

ExperienceActionAutoShow = 1,

// Ask the user if they would like to see the Experience content

// prior to presenting it.

ExperienceActionAskPermission = 3

} ExperienceAction;

@property NSString\* customDetails: Additional details added to the experience that did not fall into any of the standard properties.

@property NSString\* expTitle: The Experiences title. Typically Experiences will be displayed with a Title and Description

@property NSString\* expDescription: The Experiences description. This field contains the message that you are trying to get across to your customers.

@property NSString\* notificationTitle: This field should contain the text you would like to display in a Notification to the user.

@property NSString\* notificationDescription: The notificationDescription is an additional field that can be used to better describe what you are notifying the user about.

@property BOOL showNotif: Indicates whether or not to display a notification to the user for the given Experience.

@property NSDictionary\* attributes: Contains various values that differ across each Experience type. The expected values for each Experience type are described below.

## VideoExp

VideoExp contains video content that should be made available to the user. This class also provides accessory methods to allow you to easily play the video if you would like.

@property DisplayType displayType: Enum that indicates what mode to display the video in.

typedef enum : int

{

fullscreen = 0,

large = 1,

small = 2

} DisplayType;

@property ContentProvider contentProvider: Enum that informs you of where the video is hosted.

typedef enum : int

{

ContentProviderCustom = 0,

ContentProviderYoutube = 1,

ContentProviderVimeo = 2

} ContentProvider;

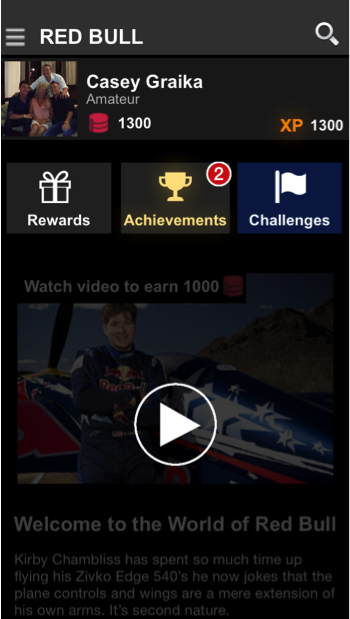
@property (nonatomic, retain) NSString \*vidURL: A URL that points to the video.

@property (nonatomic, retain) NSString \*vidPlayPictureURL: A URL that points to a picture the user must click on for the video to play. This field may or may not exist depending on your app’s style.

#### Example VideoExp

Below is an example of how you could display a video experience. You could use the properties however you like, but the intention and most common means of using the following properties is shown below. Depending on how you configure your experiences in the Footmarks Management Console, you may not populate data for all of the properties. In this case, remember to do proper error checking and just retrieve what attributes makes sense.

## 



vidPlayPictureURL

expTitle

expDescription

## ImageExp

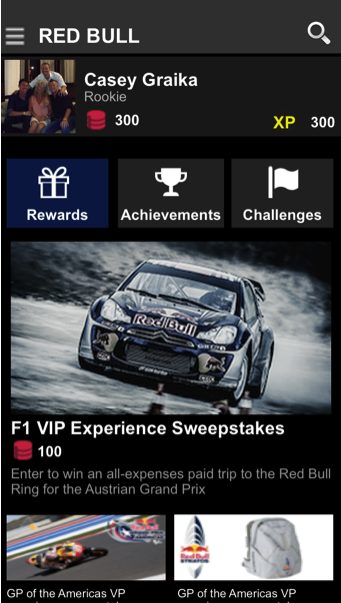
ImageExp contains an image that should be presented to the user. This class contains a helper method to load the Image data using a background thread and upon completion presents the image in a UIImageView of your choosing.

@property (nonatomic, retain) NSString \*imgURL: A URL that points to the image.

#### Example ImageExp’s

Below are 2 examples of how an ImageExp could be displayed. One using only the expTitle property and the other using both the expTitle and expDescription properties.

## AlertExp



expTitle

expDescription

imgURL

**Example:**

**expTitle & expDescription**



expTitle

imgURL

**Example:**

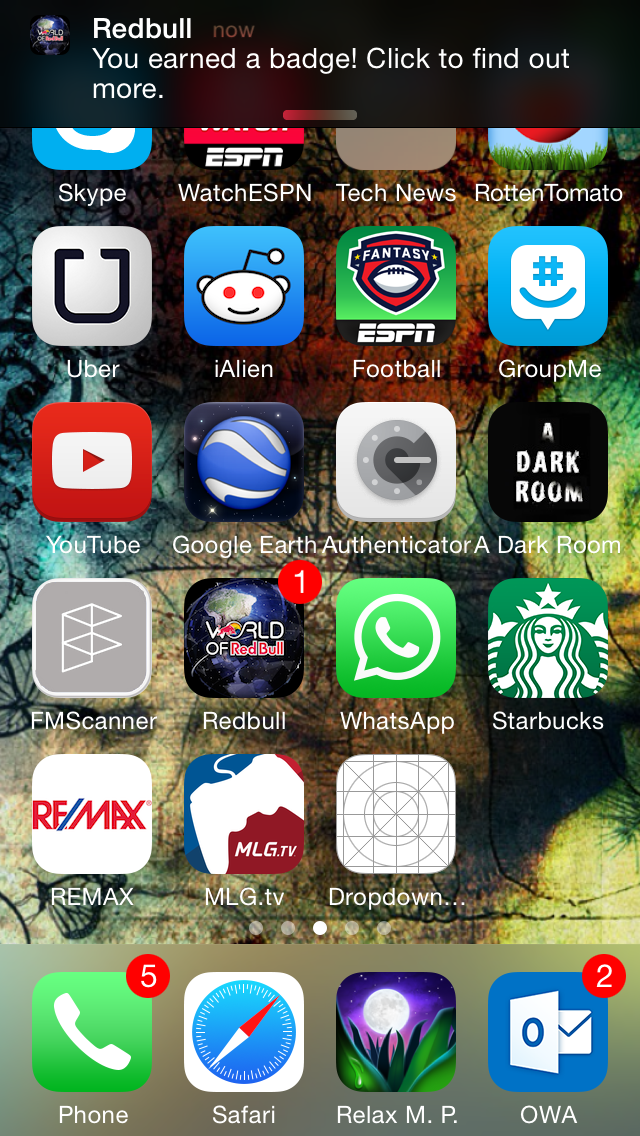
**expTitle only**

An Experience that is intended to simply present a notification to the user. This class contains a helper method, showAlert(). showAlert() parses out the notificationTitle attribute and presents an alert with this string.

#### Example AlertExp

Below is an example of how the AlertExp may be used. As you will see, the text displayed in the UILocalNotification is pulled from the Experience’s notificationTitle property.

notificationTitle



## CustomExp

This is a catchall Experience that can be modified to fit your use case in the Footmarks Management Console. If used, the customized data will be returned in the base classes Attributes property.