Push IO

Integration Guide

Version 2.0 RC : 7/27/2012

THIS DOCUMENT IS PRELIMINARY - INCOMPLETE AND SUBJECT TO CHANGE.

Billing
For information on your account and billing, please contact sales@push.io

Contact Info
Push IO
1035 Pearl Street, Suite 400
Boulder, CO 80302 USA
303-335-0903
support@push.io
http://push.io

Table of Contents

Preface

Push IO is a leading provider of real-time push notification alerts and mobile data delivery. This document provides the necessary information to leverage Push IO for your mobile application.

This document corresponds to version 1 of the Push IO Public API as of 7/27/2012 and the following components:

PushIOManager for iOS version 2.0.1

Definitions

API Key

Sender Secret

Category

Audience

Test Device

App Integration

Overview
Push IO provides a lightweight PushIOManager library for each supported platform.

The library provides simple methods for registering so Push IO can send push notifications to a device via the platform gateway. The library also provides interfaces for segmenting your users into groups called *Categories***.** For instance, you may want to send targeted push notifications only to those users who have expressed an interest in Bird Watching.

Push IO is the only push notification provider which allows you to understand how push notification engagement leads your users to high-value actions like in-app purchases, premium content viewing, and more. The PushIOManager library provides a simple mechanism to capture this push conversion information, which is available to you via the Push IO dashboard at https://manage.push.io.

Integration Prerequisites
Before continuing, be sure you have what you need to integrate Push IO into your app.

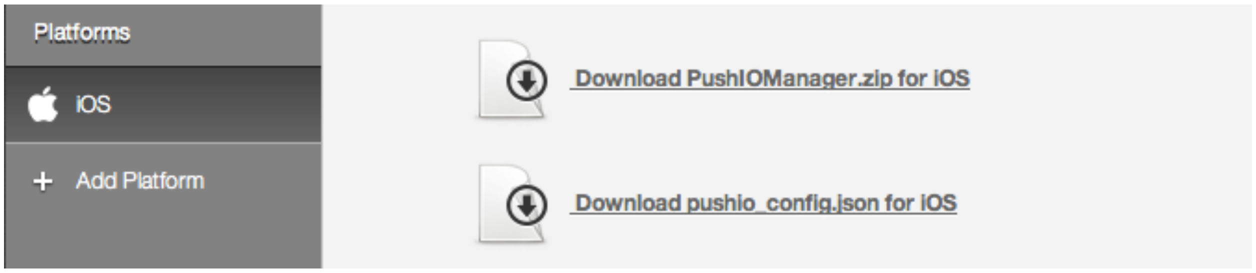
1. Sign up for an account at https://manage.push.io
2. Download our Push IO Mobile Dashboard app to easily send your first push!
3. Setup your app and platform(s) at https://manage.push.io
4. Download the PushIOManager library for each platform from Set Up > [platform]
5. Download the pushio_config.json for each platform from Set Up > [platform]

All set? Now you're ready to integrate Push IO into your app and send your first push!

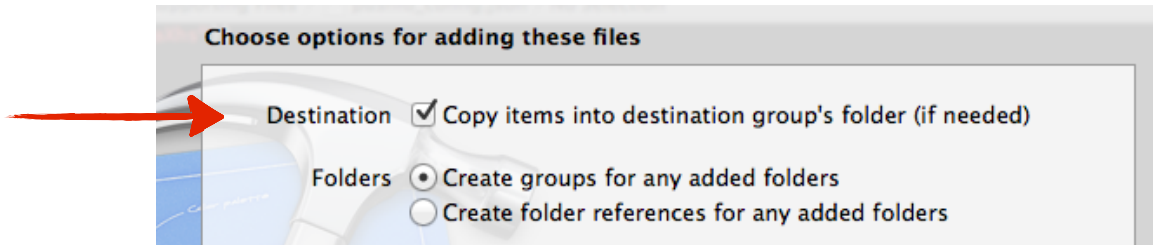
PushIOManager for iOS

In order to add the PushIOManager framework to your application, follow these steps:

Step 1: Locate the two files you downloaded from Set Up > [platform] from your Push IO management dashboard.


Platforms
iOS
Add Platform
Download PushIOManager.zip for iOS
Download pushio_config.json for iOS

Step 2: Drag the PushIOManager.framework bundle into your project. Be sure to click "Copy items into destination group's folder"


Choose options for adding these files

Destination ✓ Copy items into destination group's folder (if needed)

Folders ● Create groups for any added folders
○ Create folder references for any added folders

Step 3: Drag pushio_config.json file to your project. A great location is in the same place your AppDelegate class is stored.

The pushio_config.json must be in your application bundle in order for the PushIOManager read your API key and properly connect to the backend service.

Step 4: Import the PushIOManager header file in your AppDelegate.m

```
#import <PushIOManager/PushIOManager.h>
```

Step 5: Integrate the PushIOManager into your application lifecycle in AppDelegate.m

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
     // Initial setup
     [[PushIOManager sharedInstance] setDelegate:self];
     [[PushIOManager sharedInstance]
     didFinishLaunchingWithOptions:launchOptions];

     // Requests a device token from Apple
     [[UIApplication sharedApplication]
     registerForRemoteNotificationTypes:UIRemoteNotificationTypeAlert
     | UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeSound];
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
     [[PushIOManager sharedInstance] applicationWillEnterForeground];
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
     [[PushIOManager sharedInstance] applicationDidBecomeActive];
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
     [[PushIOManager sharedInstance] applicationDidEnterBackground];
}
```

Step 6: Implement the Apple Push Notification Service callbacks

```
- (void)application:(UIApplication *)application
  didRegisterForRemoteNotificationsWithDeviceToken:(NSData
  *)deviceToken
{
     [[PushIOManager sharedInstance]
     didRegisterForRemoteNotificationsWithDeviceToken:deviceToken];
}

- (void)application:(UIApplication *)application
  didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
{
     [[PushIOManager sharedInstance]
     didFailToRegisterForRemoteNotificationsWithError:error];
}
```

.....

```
- (void)application:(UIApplication *)application
  didReceiveRemoteNotification:(NSDictionary *)userInfo
{
     [[PushIOManager sharedInstance]
     didReceiveRemoteNotification:userInfo];
}
```

Step 7: Implement PushIOManager delegate protocol

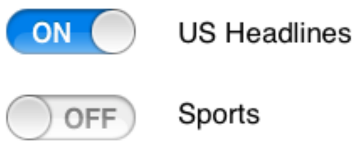
```
- (void)readyForRegistration
{
     // PushIOManager has a proper device token, so now you
     // are ready to register.
}

- (void)registrationSucceeded
{
     // Push IO registration was successful
}

- (void)registrationFailedWithError:(NSError *)error statusCode:
  (int)statusCode
{
     // Push IO registration failed
}
```

Step 8: Register with Push IO

You can register a device with Push IO one of two ways. First, if your application gives users a way to specify a preference or favorite, you may want to register for that category you can you push to them relevant content.



This kind of registration would be tied to a UI action, such as a UISwitch on/off action.

```
// Register for US Headlines
[[PushIOManager sharedInstance] registerWithCategory:@”US”];

// Unregister for US Headlines
[[PushIOManager sharedInstance] unregisterWithCategory:@”US”];
```

If you just want to be able to broadcast to all your users at once, you can register more generically.

```
// Register device, so broadcasts can be sent to the user.
[[PushIOManager sharedInstance] registerWithPushIO];
```