Handling Push Notifications in iOS applications

The handling in client side tutorials should explain: - how to setup push notifications support in iOS Xcode project (editing the podfile?) - how to setup push notifications support in Andrid Studio project (editing the builde.gradle file?) - how to setup push notifications support in Cordova applications - how to intercept and display notifications in the client

Overview

In this tutorial, you will be learning how to handle push notification for iOS applications in Swift.

Tag notifications are notification messages that are targeted to all the devices that are subscribed to a particular tag. Tags represent topics of interest to the user and provide the ability to receive notifications according to the chosen interest.

Broadcast notifications are a form of tag push notifications that are targeted to all subscribed devices. Broadcast notifications are enabled by default for any push-enabled MobileFirst application by a subscription to a reserved `Push.all` tag (auto-created for every device). Broadcast notifications can be disabled by unsubscribing from the reserved `Push.all` tag.

Prerequisites:

- Make sure you have read the following tutorials:
  - Push Notifications Overview (../push-notifications-overview)
  - Setting up your MobileFirst development environment (../../setting-up-your-development-environment/index)
  - Adding the MobileFirst Platform Foundation SDK to iOS applications (../../adding-the-mfpf-sdk/ios)
  - MobileFirst Server to run locally, or a remotely running MobileFirst Server.
- MobileFirst Developer CLI installed on the developer workstation

Jump to:

Notifications Configuration

Create a new Xcode project or use and existing one.
If the MobileFirst Native Android SDK is not already present in the project, follow the instructions in the Adding the MobileFirst Platform Foundation SDK to iOS applications (../../adding-the-mfpf-sdk/ios) tutorial.

Project setup

1. Open the projects **podfile** and add the following lines:

```
use_frameworks!
pod 'IBMMobileFirstPlatformFoundationPush'
```

2. Save and close the **podfile**
3. Open a terminal window and `cd` to the root of the **project derectory**
4. Run the command `pod install`
5. Open project using **.xcworkspace**
6. In **AppDelegat.swift**:
    - Initialize a shared instance of `MFPPush` in the `didFinishLaunchingWithOptions`

      ```
      MFPPush.sharedInstance().initialize()
      ```

    - Declare the following notification methods `didRegisterForRemoteNotificationsWithDeviceToken` & `didReceiveRemoteNotification`

      ```
      TODO:// Update var's

      func application(application: UIApplication,    didRegisterForRemoteNotificationsWithDevic
      eToken deviceToken: NSData) {
          // Registers device token with server.
          MFPPush.sharedInstance().sendDeviceToken(deviceToken)
      }

      func application(application: UIApplication, didReceiveRemoteNotification    userInfo: [NSO
      bject : AnyObject]) {
          print("Recieved Notification \(userInfo.description)")

          var alert: String = "alert"
          var alertID: String = "ID"
          var alertPayload: String = "Payload"

          //Handle notification
      }
      ```

Old tutorial

Notifications Configuration

Notifications API

API methods for tag notifications

Client-side API

- [`[WLPush sharedInstance]subscribeTag:tagName :options)]` - Subscribes the device to the specified tag name.
- `[[WLPush sharedInstance]unsubscribeTag:tagName :options)]` - Unsubscribes the device from the specified tag name.
- `[WLPush sharedInstance]isTagSubscribed:tagName]` - Returns whether the device is subscribed to a specified tag name.

Common API methods for tag and broadcast notifications

Client-side API

- `didReceiveRemoteNotification` - When a notification is received by a device, the `didReceiveRemoteNotification` method in the app delegate is called. The logic to handle the notification should be defined here.

```
-(void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo{
    NSLog(@"Received Notification %@",userInfo.description);
}
```

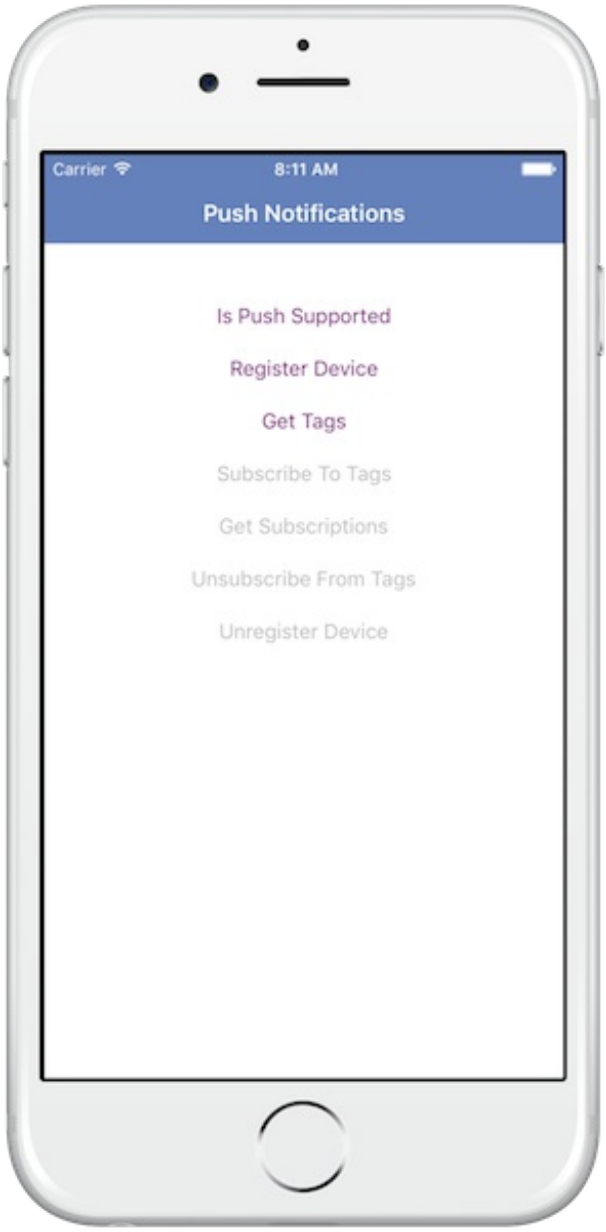
··* `userInfo` – A JSON block that contains the payload field. This field holds other data that is sent from the MobileFirst Platform server. It also contains the tag name for tag and broadcast notification. The tag name appears in the tag element. For broadcast notification, the default tag name is `Push.ALL`. * `setOnReadyToSubscribeListener` - This method registers a listener to be used for push notifications. This listener should implement the `OnReadyToSubscribe()` method. `[[WLPush sharedInstance] setOnReadyToSubscribeListener:readyToSubscribeListener];`

Handling a push notification

Handling a secure push notification

Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/PushNotificationsSwift/tree/release80) the Xcode project.



Sample usage

1. From the command line, navigate to the project's root folder.
2. Ensure the sample is registered in the MobileFirst Server by running the command: `mfpdev app register`.
3. Ensure pod's are install using the command: `pod install`
4. Import the project to Xcode using the .xcworkspace file, and run the sample by clicking the **Run** button.