Mobile Engagement SDK Integration Guide (iOS)

Mobile Engagement SDK Integration Guide (iOS)

Requirements

- iOS 9.0 or greater
- Xcode 6 or greater

Prerequisites

- iOS Core SDK downloaded from: https://github.com/phunware/maas-core-ios-sdk
- Apps configured in MaaS Portal (See: MaaS Portal Setup Guide 1.0)
- iOS Mobile Engagement SDK downloaded from: https://github.com/phunware/maas-messaging-ios-sdk

Step 1: Installation

The recommended way to use PWEngagement is via CocoaPods.

1. Add the following pod to your `Podfile`: pod 'PWEngagement'
2. Do pod install
3. Navigate to Build Phases inside your Xcode Projects and click on Link Binary With Libraries add following frameworks:
    a. CoreLocation.framework
    b. libz.tbd
    c. libsqlite3.tbd

Step 2: p12 Push Notification Certificate

A. In your Apple Developer's Console get your p12 Certificate. You will download this certificate to MaaS Portal to enable Broadcast Campaigns.

- Apple's documentation
- Apple Push Notification services (APNs) tutorial

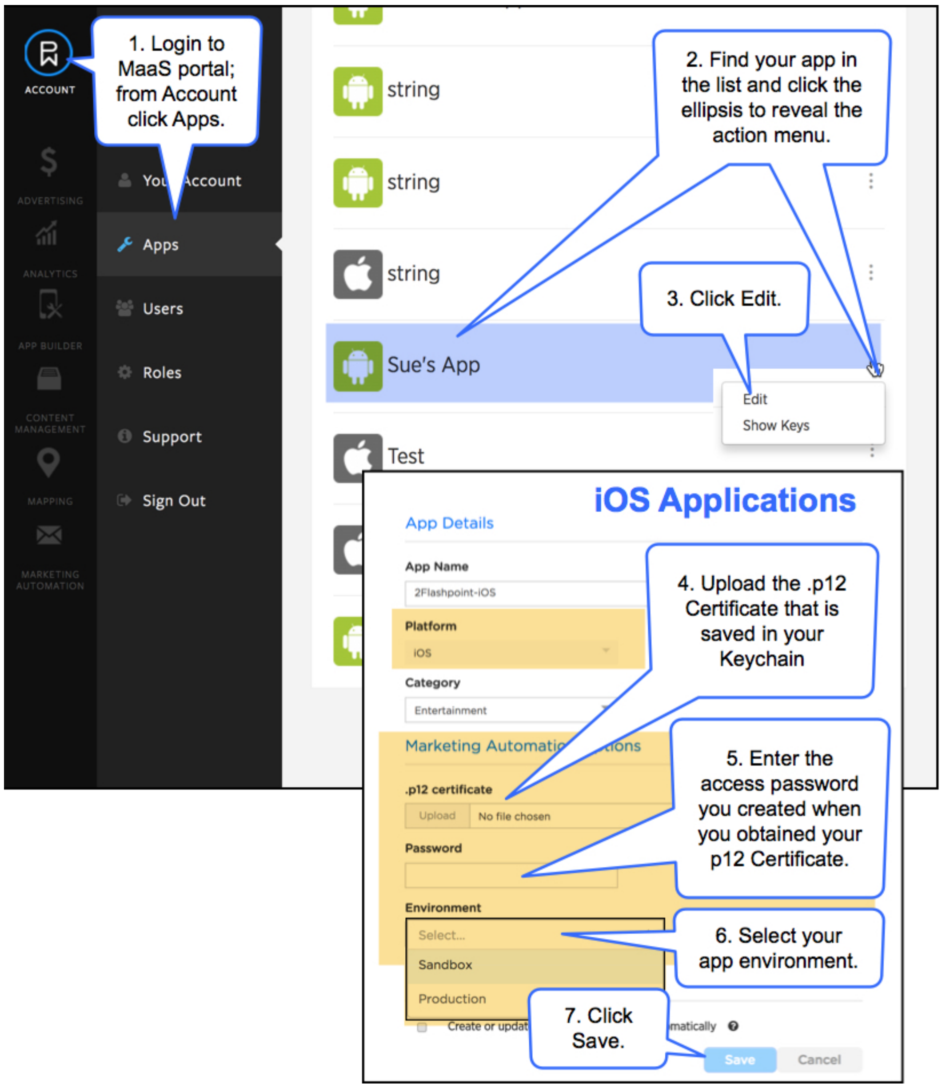
B. Once it's created, download the push production certificate and add it to Keychain Access. Then, from Keychain Access, export both the certificate and key.

(Right click to view the Export option) as a .p12 and set a password.

C. Log in to Maas portal and navigate to the app created for your application and update the following.

- Click **Apps** to go to your application list.
- Either Click **NEW** or find the App you want to send push notifications for and click **Edit** in the Action menu
- Complete the Marketing Automation section as follows:
- p12 Certificate: Upload the certificate you downloaded to your local drive
- Password: The password you setup for the push certificate.

- Environment: Select Production environment for production apps.


1. Login to MaaS portal; from Account click Apps.
2. Find your app in the list and click the ellipsis to reveal the action menu.
3. Click Edit.
4. Upload the .p12 Certificate that is saved in your Keychain
5. Enter the access password you created when you obtained your p12 Certificate.
6. Select your app environment.
7. Click Save.
ACCOUNT
ADVERTISING
ANALYTICS
APP BUILDER
CONTENT MANAGEMENT
MAPPING
MARKETING AUTOMATION
Your Account
Apps
Users
Roles
Support
Sign Out
string
string
string
Sue's App
Test
Edit
Show Keys
Save
Cancel

iOS Applications

App Details

App Name: 2Flashpoint-iOS

Platform: iOS

Category: Entertainment

Marketing Automation Options

.p12 certificate: Upload No file chosen

Password:

Environment: Select... / Sandbox / Production

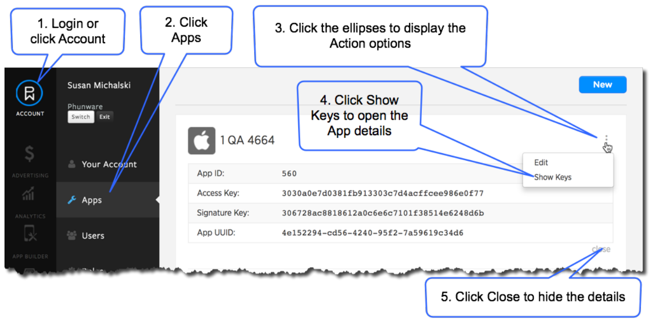
Create or update ... automatically

Step 3: Application Setup

Add the following key/values to your application's Info.plist file:

- Required background modes:
  - `App registers for location updates` (This value allows the app to keep users informed of their location, even while it is running in the background.)
  - `App downloads content in response to push notifications` (This value allows the app to regularly download remote notifications.)
  - Uses Bluetooth low energy accessories (This value allows for the app to range for vBLE .)
- MaaSAppId : The application ID matching the server choice.
- MaaSAccessKey : The accesskey for your application.
- MaaSSignatureKey : The signature key for your application.

iOS 11 Location Prompt Changes
- NSLocationWhenInUseUsageDescription: The message to display to user for user to grant the location usage only when in use.
- NSLocationAlwaysAndWhenInUseUsageDescription: The message to display to user for user to grant the location usage when in use and always.
- NSLocationAlwaysUsageDescription: The message you want to display on the prompted alert when the user grants


1. Login or click Account
2. Click Apps
3. Click the ellipses to display the Action options
4. Click Show Keys to open the App details
5. Click Close to hide the details
Susan Michalski
Phunware
Switch
Exit
ACCOUNT
ADVERTISING
ANALYTICS
APP BUILDER
Your Account
Apps
Users
New
Edit
Show Keys
close

1 QA 4664

App ID:	560
Access Key:	3030a0e7d0381fb913303c7d4acffcee986e0f77
Signature Key:	306728ac8818612a0c6e6c7101f38514e6248d6b
App UUID:	4e152294-cd56-4240-95f2-7a59619c34d6

the app permission to use the location service.
iOS 13 Bluetooth Prompt - add Key and description to application's info.plist

- Key: NSBluetoothAlwaysUsageDescription. App needs access to bluetooth for hardware measurements and to receive location based messages from bluetooth beacons
- Wifi Capability - The Mobile Engagement SDK has a dependency on the Phunware Core SDK. If you would like Core to be able to return information about the Wifi network the device is attached to via the "currentWifi" public method in the Core SDK's CoreDevice class, then add the "networking.wifi-info" key to your application by going to the capabilities section in the app's project file. Turn on the "Access Wifi Information" button. Then make sure the provisioning profile used to sign the application has the same capability.

Step 4: Application Initialization

At the top of your application delegate (.m) file, add the following:

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

Objective-C Bridging Header for Swift projects (Option)

For Swift projects, you will need an Objective-C Bridging Header that has the above import. Xcode will prompt you to create a bridging header if you attempt to add an objective-c file if your project was created as Swift, and it will prompt you if you attempt to add a Swift file to an originally objective-c project. If you already declined the option for Xcode to generate one automatically, here's how you can create one manually: http://www.learnswiftonline.com/getting-started/adding-swift-bridging-header/

Inside your application delegate, you will need to initialize PWEngagement in the application:didFinishLaunchingWithOptions: method.

```
- (BOOL)application:(UIApplication
*)application
didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
    // These values can be found for your
application in the MaaS portal.

   [PWEngagement startWithMaasAppId:<#appid#>
          accessKey:<#accessKey#>
       signatureKey:<#signatureKey#>
      completion:^(NSError *error) {
   }];
}
```

Step 5: Beacon and Geofence Registration

As of PWEngagement, the *application developer* needs to implement the following delegate in launch options. This registers for the remote notifications and notifies user when app is launched from a local notification (Geofence/ Beacon).

```
[PWEngagement
didFinishLaunchingWithOptions:launchOptions
withCompletionHandler:^BOOL(PWMELocalNotificati
on *notification) {

}]
```

Step 6: Remote Notification Handling

Apple has these primary methods for handling remote notifications. You will need to implement these in your application delegate, forwarding the results to PWEngagement:

```
- (void)application:(UIApplication *)app
didRegisterForRemoteNotificationsWithDeviceToke
n:(NSData *)devToken
{
    [PWEngagement
didRegisterForRemoteNotificationsWithDeviceToke
n:devToken];
}

- (void)application:(UIApplication *)app
didFailToRegisterForRemoteNotificationsWithErro
r:(NSError *)err
{
    [PWEngagement
didFailToRegisterForRemoteNotificationsWithErro
r:err];
}

- (void)application:(UIApplication
*)application
didReceiveRemoteNotification:(NSDictionary
*)userInfo
{
    [PWEngagement
didReceiveRemoteNotification:userInfo
withNotificationHandler:^(PWMELocalNotification
*notification) {

    }];
}

- (void)application:(UIApplication
*)application
didReceiveRemoteNotification:(NSDictionary
*)userInfo fetchCompletionHandler:(void
(^)(UIBackgroundFetchResult))completionHandler
{
    [PWEngagement
didReceiveRemoteNotification:userInfo
fetchCompletionHandler:completionHandler
withNotificationHandler:^(PWMELocalNotification
*notification) {

    }];
}
```

Step 7: Fetching/Updating Profile Attributes

PWMEAttributeManager customizes notifications to users' preferences. The PWMEAttributeManager is the central point for fetching and updating attributes associated with the device or user. You use an instance of this class to fetch and update profile and user attributes.

```
// Get Profile attribute meta data.
  [[PWMEAttributeManager sharedManager]
fetchProfileAttributeMetadataWithCompletion:^(N
SArray *attributes, NSError *error) {

  }];

// Get profile attributes.
  [[PWMEAttributeManager sharedManager]
fetchProfileAttributesWithCompletion:^(NSDictio
nary *attributes, NSError *error) {

  }];

// Update profile attributes.
  [[PWMEAttributeManager sharedManager]
updateProfileAttributes:<#userattributes#>
completion:^(NSError *error) {

  }];
```

Step 8: Listening for and Receiving Notifications

The application can listen to the following to be notified of Message Read/Modified notifications:

- PWMEReceiveMessageNotificationKey
- PWMEDeleteMessageNotificationKey
- PWMEReadMessageNotificationKey

The application can be notified of Geofence events like entry/exits. Add/Remove and Modify Geofence notifications by subscribing to the following keys.

- PWMEAddGeoZonesNotificationKey
- PWMEDeleteGeoZonesNotificationKey
- PWMEEnterGeoZoneNotificationKey
- PWMEExitGeoZoneNotificationKey

```
// Adding an Observer
[[NSNotificationCenter defaultCenter]
addObserver:self
selector:@selector(didEnterZoneNotification:)
name:PWMEEnterGeoZoneNotificationKey
object:nil];

// Removing an Observer
[[NSNotificationCenter defaultCenter]
removeObserver:self
name:PWMEEnterGeoZoneNotificationKey
object:nil];

- (void)didEnterZoneNotification:(NSNotificatio
n*)notification {
    NSString *identifier =
notification.userInfo[PWMEGeoZoneIdentifierKey]
;
    id<PWMEZone> msgZone;
    for (id<PWMEZone> zone in [PWEngagement
geozones]) {
       if ([zone.identifier
isEqualToString:identifier]) {
           msgZone = zone;
       }
    }
   UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@"Entered Zone" message:[zone
identifier] delegate:nil
cancelButtonTitle:@"OK" otherButtonTitles:nil];
   [alert show];
}
```

Deep Linking (Option)

The Message object has metadata property which may consist of multiple key-value pairs that allow for deep link to different sections of the application (Ex: Launch Image, Initial screen, app launch behavior etc.)

By default a notification launches the App and goes to the HOME page. Metadata deep linking can direct the notification to open elsewhere in the app or to web URL.

```
[PWEngagement
didReceiveRemoteNotification:userInfo
withNotificationHandler:^(PWMELocalNotification
*notification) {
        if (notification) {
            // Deep linking
   NSDictionary *metaData =
notification.message.metaData;
   NSString *className = [metaData
valueForKey:@"className"];
   Class class = NSClassFromString(className);
   //display the view controller
        }
 }];
```

Promotions Handling (Option)

The campaign can be setup to have promotions configured as html pages. The promotions are loaded in a web-view.

```
UIWebView *webView;
PWMEZoneMessage *message; //message returned
from the notification
NSString *promotionInformation =
message.promotionBody;

[webView loadHTMLString:promotionInformation
baseURL:nil];
```

API Documentation

PWEngagement API documentation is included in the Documents folder in the repository as both HTML and as a .docset. You can also find the latest documentation here: Introduction to Mobile Engagement SDK Reference (iOS)