# LP Messaging SDK and Cordova Integration for iOS

# Basic Deployment Guide

Document Version: 2.0.1
February 2017

## Folder Content:

1. **LPSDKCordova.zip** - zipped sample app source code that demonstrate a cordova app with installed LP Messaging SDK plugin.
   To open the project in XCode navigate to the following path under the extracted folder and open workspace:

   ```
   ../LPSDKCordova/platforms/ios/LPSDKCordovaSample.xcworkspace
   ```

2. **MessagingSDKPlugin** - LP Messaging SDK plugin for cordova.
3. **MessagingSDKPlugin/src/frameworks** - LP Messaging SDK frameworks:
   LPAMS.framework
   LPInfra.framework
   LPMessagingSDK.framework
   LPMessagingSDKModels.bundle

   (Note: always use the latest one from LP public GitHub
   https://github.com/LP-Messaging/iOS-Messaging-SDK)

## Basic Installation:

If you want to integrate this plugin into your cordova project please follow these steps:

1. Create new Cordova project.
2. Copy 'MessagingSDKPlugin' folder to the same folder level you created the Cordova project.
3. Make sure to have iOS platform added to your project. If not, run the following command:

   ```
   cordova platform add ios
   ```

4. Install the MessagingSDKPlugin using the following command:

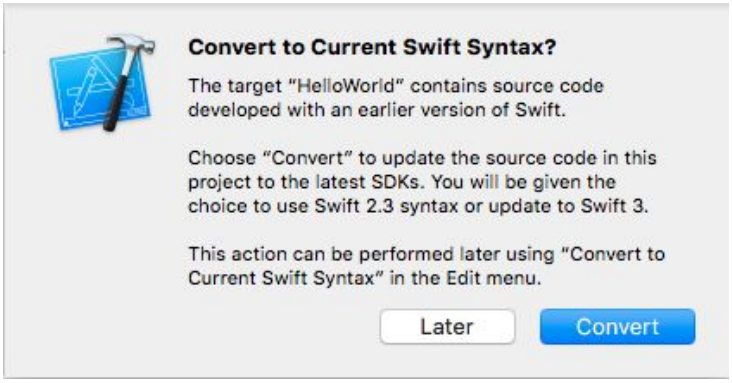
   ```
   cordova plugin add ../MessagingSDKPlugin/
   ```

5. Make a the following command to build the iOS project:

```
cordova build ios
```

**NOTE: Incase of build failures:**

1. **Error: ios-deploy was not found. Please download, build and install version 1.9.0 or greater from https://github.com/phonegap/ios-deploy into your path, or do 'npm install -g ios-deploy'**
   Fix: Open the project and change the Device to a Simulator and make sure no device is connected.
2. **** BUILD FAILED **** With a warning of older swift
   Fix: Open the xcode project (section 6) and make **Swift Convert** and then run the command again.




6. Open the XCode workspace project under:

```
../LPSDKCordova/platforms/ios/LPSDKCordovaSample.xcworkspace
```

7. In project settings, navigate to the **General** tab, and add all Framework files to the **Embedded Binaries** section.


8. Call from the JS file to the MessagingSDKPlugin as follow:

```
var success = function(message) {
        console.log("OnEvent JS: " + message)
    }

    var failure = function() {
        console.log("Error calling lp_conversation_api Plugin");
    }
    var action = lp_sdk_init; or var action = start_lp_conversation;
    var accountId = xxxx;

    lpMessagingSDK.lp_conversation_api(action, accountId, success,
failure);
```

**In the sample app source code (LPSDKCordova/platforms/android/assets/www/js/index.js) you can find example of how to use this plugin.

9. To set user profile call from the JS file to the MessagingSDKPlugin as follow:

```
  lpMessagingSDK.lp_conversation_api("set_lp_user_profile",
[accountId, "John","Doe","JohnDe",
"https://s-media-cache-ak0.pinimg.com/564x/a2/c7/ee/a2c7ee8982de3bae5
03a730fe4562cf9.jpg", "11223344"], success, failure);
```

# MessagingSDKPlugin Content:

1. LPMessagingSDKPlugin.swift - Contain the “execute” method between the JS and the iOS SDK code
2. LPMessagingSDK.js - Contain the definition of the bridge between the iOS SDK code to JS
3. Plugin.xml - Definition of the “LPMessagingSDK” obj and the required permission
4. Other FCM files are not in used for iOS - only used for Android Push handling

# Push Notification Support (iOS):

In the sample app we used the natvie UIApplication push notification methods in AppDelegate.m.

- **registerPushNotifications**:
  Register to LPMessagingSDK push notifications with the following code in AppDelegate in *didRegisterForRemoteNotificationsWithDeviceToken*:

| *func registerPushNotifications(token: Data, notificationDelegate: LPMessagingSDKNotificationDelegate? = nil, alternateBundleID: String? = nil)* | |
|---|---|
| token | A token that identifies the device to APNs. The token is an opaque data type because that is the form that the provider needs to submit to the APNs servers when it sends a notification to a device. The APNs servers require a binary format for performance reasons.<br><br>This is the exact same dictionary as received in application:didRegisterForRemoteNotificationsWithDeviceToken:<br><br>method |
| notificationDelegate | An implementer of LPMessagingSDKNotificationDelegate. |
| alternateBundleID | An optional value that can be used so that the LivePerson pusher service identifies your app with this identifier.<br><br>In debug mode, the SDK appends “-dev” string to the bundle ID. |

- **hanldePush:**
  In order to receive all incoming push notifications in a single function and handle them, add the following method in *didReceiveRemoteNotification* and *didReceiveRemoteNotification*

| *func handlePush(_ userInfo: [AnyHashable : Any])* | |
|---|---|
| userInfo | A dictionary that contains information related to the remote notification. This is the exact same dictionary as received in application(_:didReceiveRemoteNotification:fetchCompletionHandler:) method |