Handling Push Notifications in Cordova

Overview

Agenda

- Notifications configuration
- Notifications API
- Handling a push notification
- Handling a secure push notification

Notifications Configuration

Installing necessary libraries

You should already have Node.js/npm and the Cordova package installed. If you don't, you can download and install Node from https://nodejs.org/en/download/ (https://nodejs.org/en/download/).

The Cordova library is also required to use this plugin. You can find instructions to install Cordova and set up your Cordova app at https://cordova.apache.org/#getstarted (https://cordova.apache.org/#getstarted).

Creating a Cordova application

1. Run the following commands to create a new Cordova application. Alternatively you can use an existing application as well.

   ```
   $ cordova create {appName}
   $ cd {appName}
   ```

2. Edit `config.xml` file and set the desired application name in the `<name>` element instead of a default HelloCordova.
3. Continue editing `config.xml`. Update the `<platform name="ios">` element with a deployment target declaration as shown in the code snippet below.

   ```
   <platform name="ios">
       <preference name="deployment-target" value="8.0" />
       // other properties
   </platform>
   ```

4. Continue editing `config.xml`. Update the `<platform name="android">` element with a minimum and target SDK versions as shown in the code snippet below.

   ```
   <platform name="android">
       <preference name="android-minSdkVersion" value="15" />
       <preference name="android-targetSdkVersion" value="23" />
       // other properties
   </platform>
   ```

> The minSdkVersion should be above 15.
> The targetSdkVersion should always reflect the latest Android SDK available from Google.

Adding Cordova platforms

Run the following commands according to which platform you want to add to your Cordova application

```
cordova platform add ios

cordova platform add android
```

Adding the Cordova plugin

From your Cordova application root directory, enter the following command to install the Cordova Push plugin.

```
cordova plugin add ibm-mfp-push
```

From your app root folder, verify that the Cordova Core and Push plugin were installed successfully, using the following command.

```
cordova plugin list
```

Configuration

Configuring Your iOS Development Environment

Follow the Configuring Your iOS Development Environment instructions from Bluemix Mobile Services Core SDK plugin (https://github.com/ibm-bluemix-mobile-services/bms-clientsdk-cordova-plugin-core)

Go to `Build Settings` > `Search Paths` > `Framework Search Paths` and verify that the following entry was added:

```
"[your-project-name]/Plugins/ibm-mfp-push"
```

Updating your client application to use the Push SDK

By default Cordova creates a native iOS project built with iOS therefore you will need to import an automatically generated Swift header in order to use the Push SDK. Add the following Objective-C code snippets to your application delegate class.

At the top of your AppDelegate.m:

```
#import "[your-project-name]-Swift.h"
```

If your project name has spaces or hyphens, replace them with underscores in the import statement. Example:

```
// Project name is "Test Project" or "Test-Project"
#import "Test_Project-Swift.h"
```

Add below code to your application delegate

Objective-C:

```
// Register device token with Bluemix Push Notification Service
- (void)application:(UIApplication *)application
     didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken{

     [[CDVMFPPush sharedInstance]
        didRegisterForRemoteNotifications:deviceToken];
}

// Handle error when failed to register device token with APNs
- (void)application:(UIApplication*)application
     didFailToRegisterForRemoteNotificationsWithError:(NSError*)error {

    [[CDVMFPPush sharedInstance]
        didFailToRegisterForRemoteNotifications:error];
}

// Handle receiving a remote notification
-(void)application:(UIApplication *)application
    didReceiveRemoteNotification:(NSDictionary *)userInfo
    fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler {

    [[CDVMFPPush sharedInstance] didReceiveRemoteNotification:userInfo];
}
```

Swift:

```
// Register device token with Bluemix Push Notification Service
func application(application: UIApplication,
    didRegisterForRemoteNotificationsWithDeviceToken deviceToken: NSData) {

    CDVMFPPush.sharedInstance().didRegisterForRemoteNotifications(deviceToken)
}

// Handle error when failed to register device token with APNs
func application(application: UIApplication,
    didFailToRegisterForRemoteNotificationsWithError error: NSErrorPointer) {

    CDVMFPPush.sharedInstance().didFailToRegisterForRemoteNotifications(error)
}

// Handle receiving a remote notification
func application(application: UIApplication,
    didReceiveRemoteNotification userInfo: [NSObject : AnyObject],  fetchCompletionHandler completionHan
dler: ) {

    CDVMFPPush.sharedInstance().didReceiveRemoteNotification(userInfo)
}
```

Configuring Your Android Development Environment

Android development environement does not require any additional configuration. You can open the Android Project generated by Cordova in [your-app-name]/platforms/android directory with Android Studio or use Cordova CLI to build and run it.

Notifications API

Usage

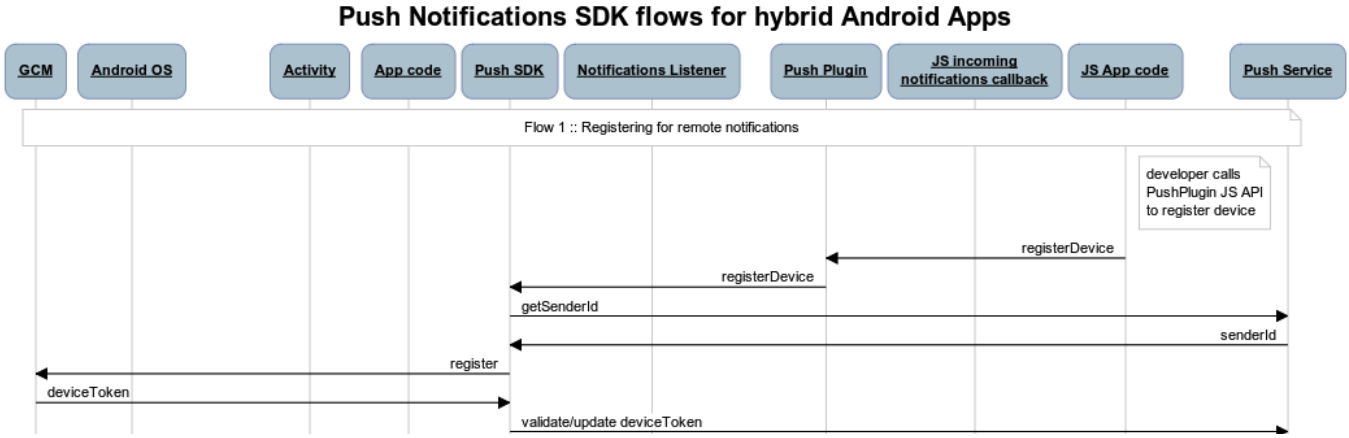
The following MFPPush Javascript functions are available:

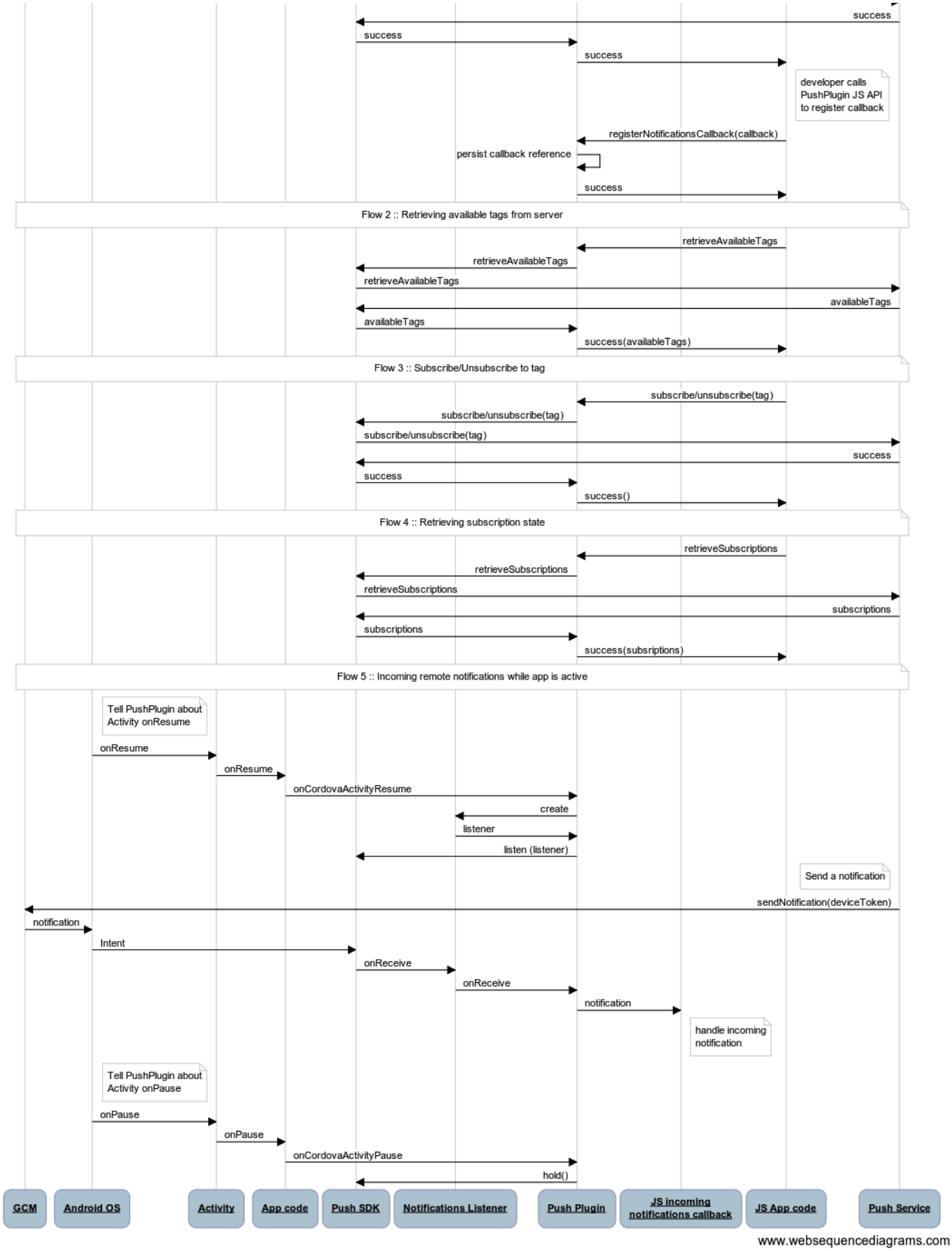
Javascript Function	Description
registerDevice(settings, success, failure)	Registers the device with the Push Notifications Service.
unregisterDevice(success, failure)	Unregisters the device from the Push Notifications Service
retrieveSubscriptions(success, failure)	Retrieves the tags device is currently subscribed to
retrieveAvailableTags(success, failure)	Retrieves all the tags available in a push notification service instance.
subscribe(tag, success, failure)	Subscribes to a particular tag.
unsubscribe(tag, success, failure)	Unsubscribes from a particular tag.
registerNotificationsCallback(callback)	Registers a callback for when a notification arrives on the device.

Android (Native) The following native Android function is available.

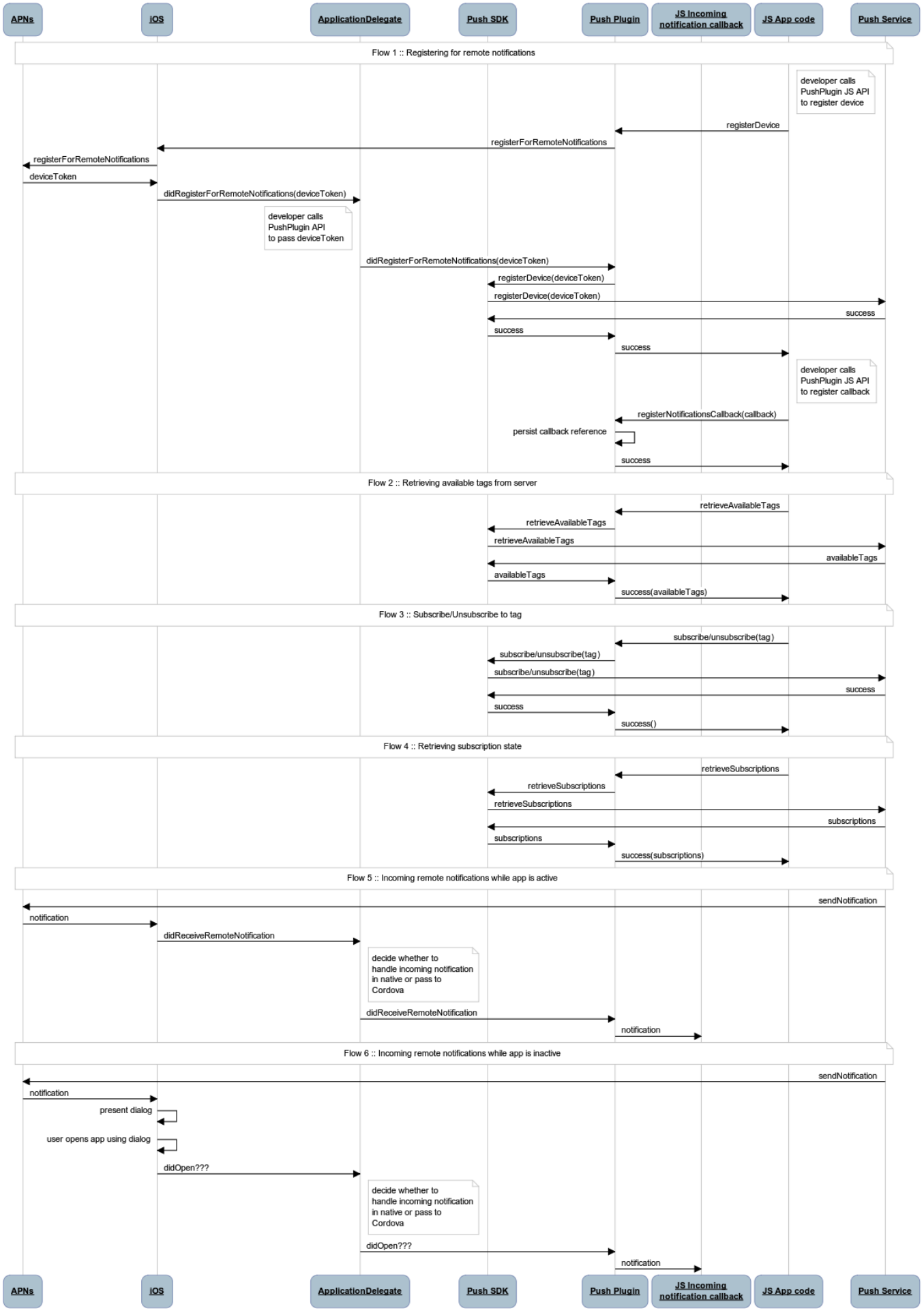
Android function	Description
CDVMFPPush. setIgnoreIncomingNotifications(boolean ignore)	By default, push notifications plugin handles all incoming Push Notification by tunnelling them to JavaScript callback. Use this method to override the plugin's default behavior in case you want to manually handle incoming push notifications in native code.

SDK Sequence Diagrams





Push Notifications SDK flows for hybrid iOS Apps



Examples

Using MFPPush

Register for Push Notifications

```
var settings = {
    ios: {
        alert: true,
        badge: true,
        sound: true
    }
}

var success = function(message) { console.log("Success: " + message); };
var failure = function(message) { console.log("Error: " + message); };

MFPPush.registerDevice(settings, success, failure);
```

The settings structure contains the settings that you want to enable for push notifications. You must use the defined structure and should only change the boolean value of each notification setting.

> Android does NOT make use of the settings parameter. If you're only building Android app pass an empty object, e.g.

```
MFPPush.registerDevice({}, success, failure);
```

To unregister for push notifications simply call the following:

```
MFPPush.unregisterDevice(success, failure);
```

Retrieving Tags

In the following examples, the function parameter is a success callback that receives an array of tags. The second parameter is a callback function called on error.

To retrieve an array of tags to which the user is currently subscribed, use the following Javascript function:

```
MFPPush.retrieveSubscriptions(function(tags) {
    alert(tags);
}, failure);
```

To retrieve an array of tags that are available to subscribe, use the following Javascript function:

```
MFPPush.retrieveAvailableTags(function(tags) {
    alert(tags);
}, failure);
```

Subscribe and Unsubscribe to/from Tags

```
var tag = "YourTag";
MFPPush.subscribe(tag, success, failure);
MFPPush.unsubscribe(tag, success, failure);
```

Handling a push notification

```
var handleNotificationCallback = function(notification) {
    // notification is a JSON object
    alert(notif);
}

MFPPush.registerNotificationsCallback(handleNotificationCallback);
```

The following table describes the properties of the notification object:

Property	Description
message	Push notification message text
payload	JSON object containing additional notification payload.
sound	The name of a sound file in the app bundle or in the Library/Sounds folder of the app's data container (iOS only).
badge	The number to display as the badge of the app icon. If this property is absent, the badge is not changed. To remove the badge, set the value of this property to 0 (iOS only).
action-loc-key	The string is used as a key to get a localized string in the current localization to use for the right button's title instead of "View" (iOS only).

Example Notification structure:

```
// iOS
notification = {
    message: "Something has happened",
    payload: {
        customProperty:12345
    },
    sound: "mysound.mp3",
    badge: 7,
    action-loc-key: "Click me"
}

// Android
notification = {
    message: "Something has happened",
    payload: {
        customProperty:12345
    },
    id: <id>,
    url: <url>
}
```

Handling a secure push notification