Handling Push Notifications in Android applications

Overview

Before Android applications are able to handle any received push notifications, they must configured with support for Google Play Services. Once an application has been configured, MobileFirst-provided Notifications API can be used in order to register & unregister devices, and subscribe & unsubscribe to tags.

In this tutorial you learn how to configure an Android application and how to use the MobileFirst-provided Notifications API.

Prerequisites:

- Make sure you have read the following tutorials:
  - Setting up your MobileFirst development environment (../../setting-up-your-development-environment/index)
  - Adding the MobileFirst Platform Foundation SDK to Cordova applications (../../adding-the-mfpf-sdk/android)
  - Push Notifications Overview (../push-notifications-overview)
- 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 Android Studio project or use an 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 Android applications (../../adding-the-mfpf-sdk/android) tutorial.

Project setup

1. In **Android → Gradle scripts**, select the **build.gradle (Project: [application-name])** file add the following line to `dependencies`:

   ```
   classpath 'com.google.gms:google-services:2.0.0-alpha3'
   ```

2. In **Android → Gradle scripts**, select the **build.gradle (Module: app)** file.
   - Add the following line to `dependencies`:

     ```
     com.google.android.gms:play-services-gcm:8.4.0
     ```

- Add the following line at the bottom:

```
apply plugin: 'com.google.gms.google-services'
```

> **Important:** This line must be placed at the bottom of the file as to not create a dependency collisions. For more details see The Google Services Gradle Plugin (https://developers.google.com/android/guides/google-services-plugin) documentation.

3. Add the **google-service.json** obtained from step 4 in Setting up Push Notifications (../push-notifications-overview/#android) to your Android project's **app**/ directory.

   Optionally from Android Studio's terminal view you can type:

   - **Mac/Linux:** `$ mv path-to-download/google-services.json app/`
   - **Windows:** `$ move path-to-download/google-services.json app/`

   remove step 4 below before going live
4. Copy ibmmobilefirstplatformfoundationpush-1.0.0.aar (from halpert Electra DevOps Latest integration build) to `<android_sdk>\extras\google\m2repository\com\ibm\mobile\foundation\ibmmobilefirstplatformfoundationpush\1.0.0\ibmmobilefirstplatformfoundationpush-1.0.0.aar`

   Remove libs folder from the aar.

   Note: This step is not required once the lib gets to maven central/jcenter. Just need to add mavenCentral()/jcenter() in app gradle.
5. Add the push required configuration in **AndroidManifest.xml**:
   1. Add the following permissions to the top the `manifest` tag:

```
<!-- Permissions -->
<uses-permission android:name="android.permission.WAKE_LOCK" />

<!-- GCM Permissions -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<permission
   android:name="your.application.package.name.permission.C2D_MESSAGE"
   android:protectionLevel="signature" />
```

   2. Add the following (`MFPPush Intent Service`, `MFPPush Instance ID Listener Service`) to the `application` tag:

```
<!-- GCM Receiver -->
<receiver
   android:name="com.google.android.gms.gcm.GcmReceiver"
   android:exported="true"
   android:permission="com.google.android.c2dm.permission.SEND">
   <intent-filter>
      <action android:name="com.google.android.c2dm.intent.RECEIVE" />
      <category android:name="your.application.package.name" />
   </intent-filter>
</receiver>

<!-- MFPPush Intent Service -->
<service
   android:name="com.ibm.mobilefirstplatform.clientsdk.android.push.api.MFPPushIntentServi
ce"
   android:exported="false">
   <intent-filter>
      <action android:name="com.google.android.c2dm.intent.RECEIVE" />
   </intent-filter>
</service>

<!-- MFPPush Instance ID Listener Service -->
<service
   android:name="com.ibm.mobilefirstplatform.clientsdk.android.push.api.MFPPushInstanceID
ListenerService"
   android:exported="false">
   <intent-filter>
      <action android:name="com.google.android.gms.iid.InstanceID" />
   </intent-filter>
</service>
```

> **Note:** Be sure to replace `your.application.package.name` with the actual package name of your application.

Notifications API

TODO: Add introduction text to the API.

API methods for tag notifications

- `MFPPush.subscribe(String[] tagNames, MFPPushResponceListener)` - Subscribes the device to the specified tag(s).
- `MFPPush.unsubscribe(String[] tagNames, MFPPushResponceListener)` - Unsubscribes the device from the specified tag(s).
- `MFPPush.getTags(MFPPushResponceListener)` - Returns a list of available tags the device can subscribe to.
- `MFPPush.getSubscriptions(MFPPushResponceListener)` - Returns a list of tag names as strings that the device is subscribed to.

API methods for tag and broadcast notifications

- `WLNotificationListener` - Defines the callback method to be notified when the notification arrives.
- `client.getPush().setWLNotificationListener(listener)` - This method sets the implementation class of the `WLNotificationListener` interface.
- `client.getPush().setOnReadyToSubscribeListener(listener)` - This method registers a listener to be used for push notifications. This listener should implement the `onReadyToSubscribe()` method.
- The `onMessage(props,payload)` method of `WLNotificationListener` is called when a push notification is received by the device.

--* *props* – A JSON block that contains the notifications properties of the platform.

--* *payload* – A JSON block that contains other data that is sent from MobileFirst Server. The JSON block also contains the tag name for tag-based or broadcast notification. The tag name appears in the “tag” element. For broadcast notification, the default tag name is `Push.ALL`.

Handling a push notification

In order to handle a push notification you will need to set up a `MFPPushNotificationListener`. This can be achieved by implementing one of the following methods.

Option One

1. In the activity in which you wish the handle push notifications, add `implements MFPPushNofiticationListener` to the class declaration.
2. Set the class to be the listener by calling `listen(this)` on an instance of `MFPPush`.
3. Then you will need to add the following required method:

```
@Override
public void onReceive(MFPSimplePushNotification mfpSimplePushNotification) {
    // Handle push notification here
}
```

4. In this method you will receive the `MFPSimplePushNotification` and can handle the notification for the desired behavior.

Option Two

Create a listener by calling `listen(new MFPPushNofiticationListener())` on an instance of `MFPPush` as outlined below:

```
push.listen(new MFPPushNotificationListener() {
    @Override
    public void onReceive(MFPSimplePushNotification mfpSimplePushNotification) {
        // Handle push notification here
    }
});
```

Note: `push` was obtained by calling `MFPPush push = MFPPush.getInstance()`

Handling a secure push notification

TODO: Add instructions on handling secure push notifications.