# Handling SMS Notifications in Android

## Overview

SMS notifications are a sub-set of Push Notification, as such make sure to first go through the Push notifications in Android (../../) tutorials.

**Prerequisites:**

- Make sure you have read the following tutorials:
    - Notifications Overview (../../)
    - Setting up your MobileFirst development environment (../../../installation-configuration/#installing-a-development-environment)
    - Adding the MobileFirst Foundation SDK to iOS applications (../../../application-development/sdk/ios)
- MobileFirst Server to run locally, or a remotely running MobileFirst Server.
- MobileFirst CLI installed on the developer workstation

Jump to:

- Notifications API
- Using a SMS subscribe servlet
- Sample Application

## Notifications API

In SMS notifications, when registering the device, a phone number value is passed.

### Challenge Handlers

If the `push.mobileclient` scope is mapped to a **security check**, you need to make sure matching **challenge handlers** exist and are registered before using any of the Push APIs.

### Initialization

Required for the client application to connect to MFPPush service with the right application context.

- The API method should be called first before using any other MFPPush APIs.
- Registers the callback function to handle received push notifications.

```
MFPPush.getInstance().initialize(this);
```

### Register Device

Register the device to the push notifications service.

```
MFPPush.getInstance().registerDevice(new MFPPushResponseListener<String>() {
    @Override
    public void onSuccess(String s) {
        // Successfully registered
    }

    @Override
    public void onFailure(MFPPushException e) {
        // Registration failed with error
    }
}, optionObject);
```

- **optionObject**: an `JSONObject` containing the phone number to register the device with. For example:

```
JSONObject optionObject = new JSONObject();
try {
    // Obtain the inputted phone number.
    optionObject.put("phoneNumber", editPhoneText.getText().toString());
}
catch(Exception ex) {
    ex.printStackTrace();
}
```

> You can also register a device using the Push Device Registration (POST) REST API
> (http://www.ibm.com/support/knowledgecenter/en/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/rest_runtime/r_restapi_push_device_registration_post.html)

### Unregister Device

Unregister the device from push notification service instance.

```
MFPPush.getInstance().unregisterDevice(new MFPPushResponseListener<String>() {
    @Override
    public void onSuccess(String s) {
        disableButtons();
        // Unregistered successfully
    }

    @Override
    public void onFailure(MFPPushException e) {
        // Failed to unregister
    }
});
```

## Using a SMS subscribe servlet

REST APIs are used to send notifications to the registered devices. All forms of notifications can be sent: tag & broadcast notifications, and authenticated notifications

To send a notification, a request is made using POST to the REST endpoint: `imfpush/v1/apps/<application-identifier>/messages`.
Example URL:

```
https://myserver.com:443/imfpush/v1/apps/com.sample.sms/messages
```

> To review all Push Notifications REST APIs, see the REST API runtime services
> (https://www.ibm.com/support/knowledgecenter/SSHS8R_8.0.0/com.ibm.worklight.apiref.doc/rest_runtime/c_restapi_runtime.html) topic in the user
> documentation.

To send a notification, see the sending notifications (../../sending-notifications) tutorial.

## Sample application

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

**Note:** The latest version of Google Play Services is required to be installed on any Android device for the sample to run.

### Sample usage

Follow the sample's README.md file for instructions.

*Last modified on*