Push Notifications Overview

Overview

IBM MobileFirst Platform Foundation provides a unified set of API methods to send notifications to iOS, Android, Windows 8 Universal, Windows 10 UWP and Cordova (iOS, Android) applications.

This tutorial provides an introduction to push notifications and the supported notifications types, required setup steps to ready the MobileFirst Server to be able to send notificaitons, and the setup steps to ready Native and Cordova applications with support for push notifications, as well as delving into supported scenarios such as sending notifications to applications with and without authentication and available notification types.

Prerequisites: MobileFirst Server to run locally, or a remotely running MobileFirst Server.

Jump to:

What is a Push Notification

Push notifications is the ability of a mobile device to receive messages that are "pushed" from a server. Notifications are received regardless of whether the application is currently running in the foreground or background.

Notifications can take several forms:

- **Alert (iOS, Android)** - a pop-up text message
- **Badge (iOS)** - a graphical representation that allows a short text or image
- **Banner (iOS)** - a disappearing pop-up text message at the top of the device display
- **Sound (iOS, Android)** - a sound file playing when a notification is received
- **Interactive (iOS 8 and above)** - action buttons inside the banner of a received notification
- **Silent (iOS 8 and above)** - sending notifications without distrubing the user

Device support

Push notifications are supported for the following platforms in MobileFirst Platform Foundation:

- iOS 8.x, 9.x
- Android 4.x, 5.x, 6.x

Push Notification Types

Tag notifications

Tag notifications are notification messages that are targeted to all the devices that are subscribed to a particular tag.
Tags represent topics of interest to the user and provide the ability to receive notifications according to the chosen interest.

Broadcast notifications

Broadcast notifications are a form of tag push notifications that are targeted to all subscribed devices. Broadcast notifications are enabled by default for any push-enabled MobileFirst application by a subscription to a reserved `Push.all` tag (auto-created for every device). Broadcast notifications can be disabled by unsubscribing from the reserved `Push.all` tag.

User Authenticated Notifications

User Authenticated Notifications are notifications secured with OAuth.

> For more information about notifications types, see the topic about push notifications in the user documentation.

Setting up Push Notifications

The first step to enable push notifications support is to map the **push.mobileclient** scope element to the application.

1. In the MobileFirst Operations Console → **[your application] → Security → Map Scope Elements to Security Checks**, click on **Create New**
2. Write "push.mobileclient" in the **Scope element** field. Then, click **Add**.

For User Authenticated notifications the **push.mobileclient** scope element should be mapped to the security check of the application.

Android

Android devices use the Google Cloud Messaging (GCM) service for push notifications.
To setup GCM:

1. Visit Google's Services website (https://developers.google.com/mobile/add?platform=android&cntapi=gcm&cnturl=https:%2F%2Fdevelopers.google.com%2Fcloud-messaging%2Fandroid%2Fclient&cntlbl=Continue%20Adding%20GCM%20Support&%3Fconfigured%3Dtrue).
2. Provide your application name and package name.
3. Select "Cloud Messaging" and click on **Enable Google cloud messaging**.
   This step generates a `Server API Key` and a `Sender ID`.
   The generated values are used to identify the application by Google's GCM service in order to send notifications to the device.
4. Click **Generate configuration file** and download the **google-services.json** file. This file will be used later to configure the Android application (../handling-push-notifications-in-android).
5. In the MobileFirst Operations Console → **[your application] → Push → Push Settings**, add the GCM **Sender ID** and server **API Key** and click **Save**.

Notes

If your organization has a firewall that restricts the traffic to or from the Internet, you must go through the following steps:

- Configure the firewall to allow connectivity with GCM in order for your GCM client apps to receive messages.
- The ports to open are 5228, 5229, and 5230. GCM typically uses only 5228, but it sometimes uses 5229 and 5230.
- GCM does not provide specific IP, so you must allow your firewall to accept outgoing connections to all IP addresses contained in the IP blocks listed in Google’s ASN of 15169.

- Ensure that your firewall accepts outgoing connections from MobileFirst Server to android.googleapis.com on port 443.

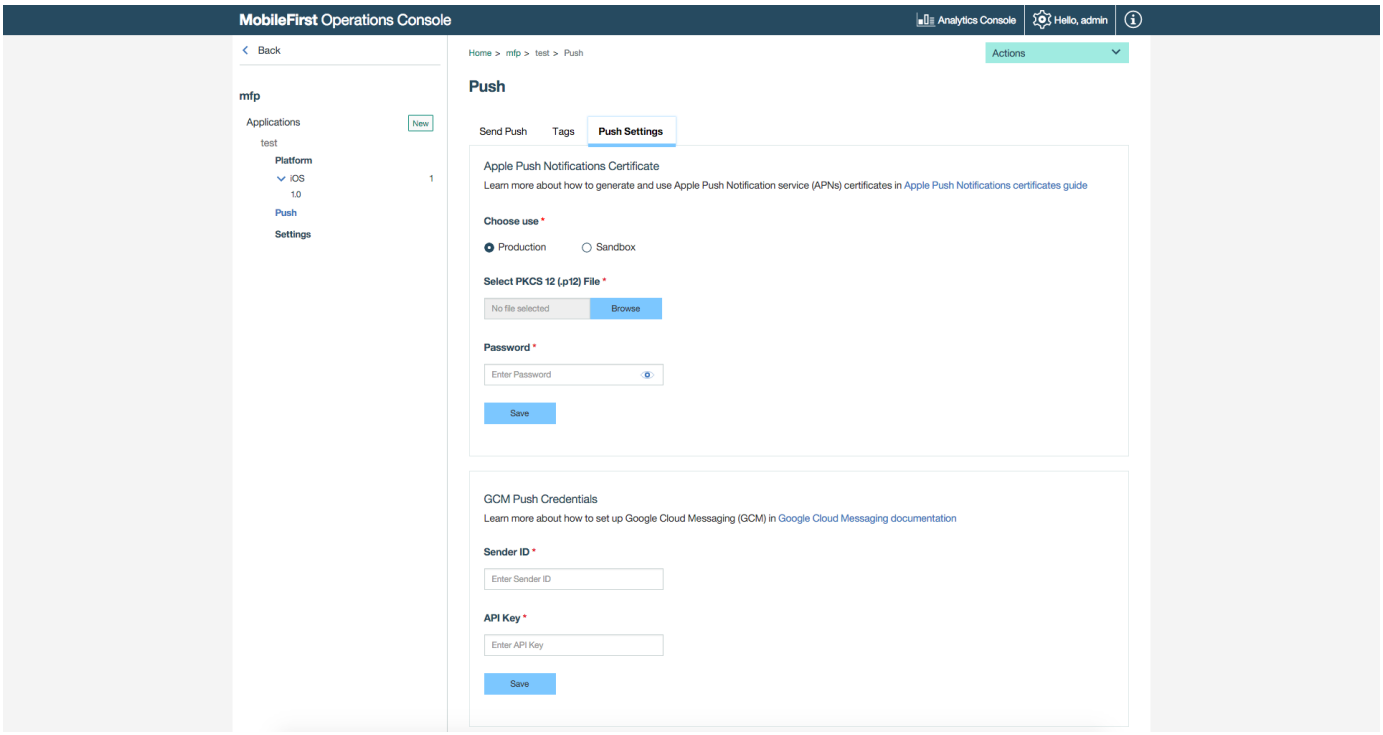
iOS

iOS devices use Apple's Push Notification Service (APNS) for push notifications.
To setup APNS:

1. Generate a push notification certificate (https://www.ibm.com/developerworks/community/blogs/worklight/entry/understanding-and-setting-up-push-notifications-in-development-evnironment?lang=en).
2. In the MobileFirst Operations Console → **[your application] → Push → Push Settings**, select the certificate type and provide the certificate's file and password. Then, click **Save**.

Notes

- For push notifications to be sent, the following servers must be accessible from a MobileFirst Server instance:
    - Sandbox servers:
        - gateway.sandbox.push.apple.com:2195
        - feedback.sandbox.push.apple.com:2196
    - Production servers:
        - gateway.push.apple.com:2195
        - Feedback.push.apple.com:2196
        - 1-courier.push.apple.com 5223
- During the development phase, use the apns-certificate-sandbox.p12 sandbox certificate file.
- During the production phase, use the apns-certificate-production.p12 production certificate file.
    - The APNS production certificate can only be tested once the application that utilizes it has been successfully submitted to the Apple App Store.



Tutorials to follow next

With the prerequisites for push notifications now accomplished, the application can be configured as well. Select a tutorial:

- Handling push notifications in Cordova applications (../handling-push-notifications-in-cordova)
- Handling push notifications in iOS applications (../handling-push-notifications-in-ios)
- Handling push notifications in Android applications (../handling-push-notifications-in-android)
- Sending push notifications (../sending-push-notifications)