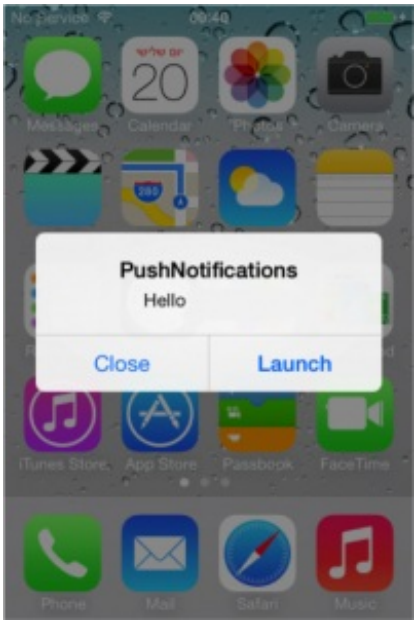
Push notifications in hybrid applications

Overview

IBM MobileFirst Platform Foundation provides a unified set of API methods to send, or push, notifications to devices where the MobileFirst application is installed in. It is possible to send a notification in 3 distinct types: event source notifications, broadcast notifications, and tag-base notifications.

This tutorial explains the concept, API, and usage of push notifications in the context of hybrid applications.
Topics:

- What is push notification?
- Notification types
- Project setup and guidelines
- Project configuration



What is 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.
Notifications can take several forms:

- **Alert (all)**: A pop-up text message
- **Badge (iOS), Tile (W8, WP8)**: A graphical representation that allows a short text or image
- **Banner (iOS), Toast (W8, WP8)**: A disappearing pop-up text message at the top of the device display
- **Sound (all)**: A sound file playing when a notification is received
- **Interactive (iOS 8)**: Action buttons inside the **Banner** of a received notification

Device support

Push notifications are supported for the following mobile platforms:

- Android 2.3.5, 4.x, 5.x
- iOS 6, 7, and 8
- Windows Phone 8.x
- Windows 8

Notification types

Event source notifications

Event source notifications are notification messages that are targeted to devices with a user subscription.

Broadcast notifications

Broadcast notifications are notification messages that are targeted to all subscribed devices.

Tag notifications

Tag notifications are notification messages that are targeted to all the devices that are subscribed to a particular tag.

For more information, select a notification type.

Project setup and guidelines

You set up push notification differently depending on the platform that you use.

Android

To send push notifications to Android devices, use the Google Cloud Messaging (GCM) service.
To register to the GCM service, you need a valid Gmail account.

For more information about how to get a GCM Project Number and API key (which you use later in the MobileFirst project), see Google Developer Console (https://console.developers.google.com).

Notes:

- When you create an API key, make sure that the type is **server key**.
- Android OS 2.3.5 devices must be synchronized with a Gmail account.
- Android OS 4.x devices do not impose Gmail account synchronization.

iOS

To send push notifications to iOS devices, use the Apple Push Notifications Service (APNS).
You must be a registered Apple iOS Developer to obtain an APNS certificate for your application. *APNS certificates must have a non-blank password.*.

- During the development phase, use the `apns-certificate-sandbox.p12` sandbox certificate file and place it in the environment root folder or in the application root folder. The environment root folder takes the highest priority.
- During a production phase, use the `apns-certificate-production.p12` production certificate file and place it in the environment root folder or in the application root folder. The environment root folder

takes the highest priority.

When the hybrid application has both iPhone and iPad environments, it requires separate certificates for push notification for each environment. In that case, place those certificates in the corresponding environment folders.

Windows Phone 8

To send push notifications to Windows Phone 8 devices, use the Microsoft Push Notifications Service (MPNS).

- Non-authenticated push notification does not require any setup from the developer. Authenticated push notification requires a Windows Phone Dev Center account.
- To use authenticated push, you must use a certificate that is issued by a Microsoft-trusted root certificate authority.
  Important: For production, consider using authenticated push notification in order to ensure that the information is not compromised.

Windows 8

To send push notifications to Windows 8 devices, use Windows Push Notification Services (WNS).
As a developer, you need to register your app with Windows Store through the Windows Dev Center by using your Microsoft account.

For more information about how to get WNS credentials (which you use later in the MobileFirst project), see http://msdn.microsoft.com/en-in/library/windows/apps/hh465407.aspx (http://msdn.microsoft.com/en-in/library/windows/apps/hh465407.aspx)

For push notification to be sent, the following servers must be accessible from a MobileFirst Server instance:

iOS
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

Android

If your organization has a firewall that restricts the traffic to or from the Internet, you must do 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 ASN of 15169. For more information, see Implementing an HTTP Connection Server (https://developers.google.com/cloud-messaging/http).

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

Windows Phone 8

No specific port needs to be open in your server configuration.
MPNS uses regular http or https requests.

Windows 8

No specific port needs to be open in your server configuration.
WNS uses regular http or https requests.

Project configuration

To set up push notifications in an application, add the following lines to the `application-descriptor.xml` file.
You can also edit these settings with the Application Descriptor Editor in Design mode.

Android

Use the values that you previously created in the GCM website:

1. Replace **GCM_Key** with the **API Key** value.
2. Replace **senderId** with the **Project Number** value.

```
<android securityTest="PushApplication-strong-mobile-securityTest" version="1.0">
<pushSender key="GCM_key" senderId="GCM_ID"/>
```

Add Google Play Services

1. From **Android SDK Manager > Extras**, add the **Google Play Services** option.
2. Import Google Play Services as a library to the Eclipse workspace:
    1. Select **File > Import**, and then select **Android > Existing Android Code into workspace**, browse to the `google-play-services_lib` project @ *`android_sdk_location`*`\extras\google\google_play_services\libproject\google-play-services_lib`
    2. After successfully importing `google-play-services_lib` into the workspace, mark it as an Android library project. To do this, right-click **imported-project > properties > Android** and select the **IsLibrary** checkbox.
3. Right-click the **generated Android project > properties > Android >** and click **Add**.
    1. In the Project Selection dialog, select the **google-play-services_lib project** and click **OK**.
    2. Click **Apply** and **OK** in the Properties window.
4. Add a reference to the google-play-services version in the *`your-app`*`\android\native\AndroidManifest.xml` file, as the first child of the element:

```
<meta-data
   android:name="com.google.android.gms.version"
   android:value="@integer/google_play_services_version" /
>
```

iOS

1. Place the Apple APNS certificate file at the root of the application folder or at the root of the environment folder.
2. Replace **certificate password** with your actual certificate password.
3. Replace `com.PushNotifications` with the `bundleId` of your application. Consult the Apple documentation about how to create a `bundleId` for Xcode projects.

```
<iphone bundleId="com.PushNotifications" version="1.0">
   <pushSender password=""/>
```

Windows Phone 8

Non-authenticated push

```
<windowsPhone8 version="1.0">
   <uuid>e446f9d1-8d04-4198-be53-9fb44ae47548</uuid
>
   <pushSender/>
```

Authenticated push

```
<windowsPhone8 version="1.0">
   <uuid>e446f9d1-8d04-4198-be53-9fb44ae47548</uuid>
   <pushSender>
   <authenticatedPush serviceName=" ... " keyAlias=" ... " keyAliasPassword=" ... "/></pushSender>
```

> For more information about using the certificate file, see the topic about setting up push notifications for Windows Phone 8, in the user documentation..

Windows 8

Use the values that you previously generated in the Windows Store Dashboard.

1. Enter the package name in `appIdentityName`.
2. Enter the subject name of the certificate in `appIdentityPublisher`.

3. Enter the package security identifier in `packageSID`.
4. Enter the secret key in `clientSecret`.

```
<windows8 version="1.0">
   <pushSender clientSecret="..." packageSID="..." appIdentityName="..." appIdentityPublisher="..."/
>
```