MobileFirst Platform {dev}

Tag and Broadcast Notifications in Native iOS Applications

Relevant to:
Native iOS

Overview

Prerequisite: Make sure to read the Push notifications in native iOS applications tutorial first.

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 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 by unsubscribing from the reserved `Push.all` tag.

Agenda

- Notifications configuration
- Notifications API

Notifications configuration

Tag Notifications configuration

Setting up tags

Tags are defined in the `application-descriptor.xml` file:

```
<nativeIOSApp xmlns="http://www.worklight.com/native-ios-descriptor"
 bundleId="com.REPLACE-WITH-BUNDLE-ID" id="NativeiOSTagNotifications"
 platformVersion="7.0.0.00.20150312-0731" version="1.0">
...
...
...
<tags>
    <tag>
        <name>my tag 1</name>
        <description>About my tag 1</description>
    </tag>
    <tag>
        <name>my tag 2</name>
        <description>About my tag 2</description>
    </tag>
</tags>
```

Notifications API

API methods for tag notifications

Client-side API

- `[[WLPush sharedInstance]subscribeTag:tagName :options)]`
  Subscribes the device to the specified tag name.
- `[[WLPush sharedInstance]unsubscribeTag:tagName :options)]`
  Unsubscribes the device from the specified tag name.
- `[WLPush sharedInstance]isTagSubscribed:tagName]`
  Returns whether the device is subscribed to a specified tag name.

Common API methods for tag and broadcast notifications

Client-side API:

- `didReceiveRemoteNotification`
  When a notification is received by a device, the `didReceiveRemoteNotification` method in the app delegate is called. The logic to handle the notification should be defined here.

  ```
  -(void)application:(UIApplication *)application
  didReceiveRemoteNotification:(NSDictionary *)userInfo{
      NSLog(@"Received Notification %@",userInfo.description);
  }
  ```

  - userInfo – A JSON block that contains the payload field. This field holds other data that is sent from the MobileFirst Platform server. It also contains the tag name for tag and broadcast notification. The tag name appears in the tag element. For broadcast notification, the default tag name is `Push.ALL`.
- `setOnReadyToSubscribeListener`
  This method registers a listener to be used for push notifications. This listener should implement the OnReadyToSubscribe() method.

  ```
  [[WLPush sharedInstance]
  setOnReadyToSubscribeListener:readyToSubscribeListener];
  ```

Server-side API

`WL.Server.sendMessage(applicationId,notificationOptions)`
This method submits a notification based on the specified target parameters and takes two mandatory parameters:

- `applicationId` – (mandatory) The name of the MobileFirst application
- `notificationOptions` – (mandatory) A JSON block containing message properties

> For a full list of message properties, refer to the `WL.Server.sendMessage` API in

> the API reference documentation.

Sample application

Before running the application, check the adapter’s `PushAdapter-impl.js` file and verify that the `WL.Server.sendMessage()` method use the correct application name. The correct application name can be determined from the `id` attribute in `application-descriptor.xml`.

Click to download the MobileFirst project.

Click to download the Native project.

- The `TagNotifications` project contains a MobileFirst native API that you can deploy to your MobileFirst server.
- The `TagNotificationsObjC` project contains a native iOS application that uses a MobileFirst native API library to subscribe for push notification and receive notifications from APNS.
- Make sure to update the `worklight.plist` file in the native project with the relevant server settings.