Handling SMS Notifications in iOS

Overview

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

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.sharedInstance().initialize()
```

Register Device

Register the device to the push notifications service.

```
MFPPush.sharedInstance().registerDevice(jsonOptions, completionHandler: {(response: WLResponse!, error: NSError!) -> Void in
    if error == nil {
        // Successfully registered
    } else {
        // Registration failed with error
    }
 })
```

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

```
let phoneNumber: String = self.phoneNumberTF.text!

let jsonOptions: [NSObject: AnyObject] = [
    "phoneNumber": phoneNumber
]

let isValid = NSJSONSerialization.isValidJSONObject(jsonOptions)

if isValid {
    // JSON is valid and can be sent with registerDevice request
}
```

> 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.sharedInstance().unregisterDevice({(response: WLResponse!, error: NSError!) -> Void in
    if error == nil {
        // Unregistered successfully
    } else {
        // 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:

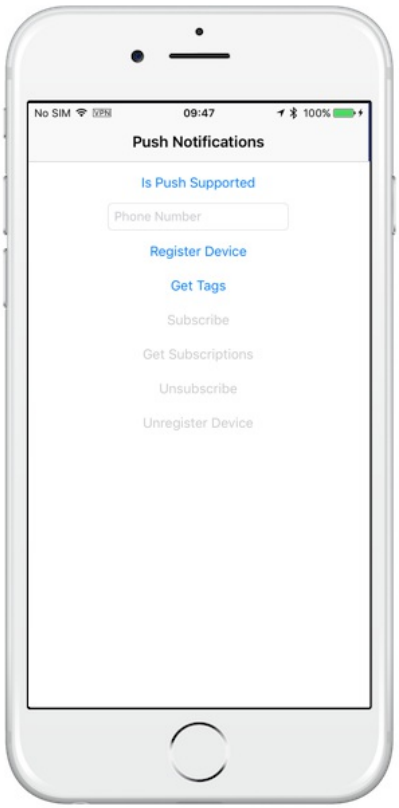
```
https://myserver.com:443/imfpush/v1/apps/com.sample.sms/messages
```

> 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/SMSNotificationsSwift/tree/release80) the Xcode project.



Sample usage

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

Last modified on