Handling Push Notifications in Cordova applications

The handling in client side tutorials should explain: - how to setup push notifications support in iOS Xcode project (editing the podfile?) - how to setup push notifications support in Andrid Studio project (editing the builde.gradle file?) - how to setup push notifications support in Cordova applications - how to intercept and display notifications in the client

Overview

Before Cordova applications are able to handle any received push notifications, they must be configured for the appropriate platform. Once an application has been configured, MobileFirst-provided Notifications API can be used in order to register & unregister devices, and subscribe & unsubscribe to tags.

In this tutorial you learn how to configure a Cordova application and how to use the MobileFirst-provided Notifications API.

Prequisites: * Make sure you have read the Setting up your MobileFirst development environment (../../../setting-up-your-development-environment/index) tutorial * MobileFirst Server to run locally, or a remotely running MobileFirst Server. * MobileFirst Developer CLI installed on the developer workstation * Cordova CLI installed on the developer workstation

Agenda

- Notifications Configuration for Android
- Notifications Configuration for iOS
- Notifications API
- Handling a push notification
- Handling a secure push notification

Notifications Configuration for Android

Prequisites

- Android Studio installed on the developer workstation.

Project Setup

1. Create Cordova Project using cordova create and mfp cordova template
2. `$ cordova platform add android`
3. `$ cordova platform add plugin cordova-plugin-mfp-push`
4. `$ cordova build`
5. Import the app/platforms/android into Android Studio
6. In build.gradle(module:Android), add to respositories (2x)

```
maven {
        url
"http://visustar.francelab.fr.ibm.com:8081/nexus/content/repositories/mobile-s"
```

`}`

7. In build.gradle(module:Android), add classpath `com.google.gms:google-services:2.0.0-alpha3` to dependencies (3x)
8. In build.gradle(module:Android), add jcenter() to repositories in buildscript block
9. Add compile `com.google.android.gms:play-services-gcm:8.4.0' to app/platforms/android/cordova-plugin-mfp-push/<appname>-build-extras.gradle in dependencies`
10. Add compile `com.squareup.okhttp:okhttp:2.6.0' to app/platforms/android/cordova-plugin-mfp-push/<appname>-build-extras.gradle in depedencies`
11. Add apply plugin: `com.google.gms.google-services' to app/platforms/android/cordova-plugin-mfp-push/<appname>-build-extras.gradle outside dependencies`
12. Add google-services.json configuration file to app/platforms/android folder
13. Change version to `8.0.0-Beta1-SNAPSHOT` in app/platforms/android folder
14. Add the Push SDK APIs to your application (Refer to Push API below)
15. If you want to change the notification title, then add `push_notification_tile` in strings.xml

Notifications Configuration for iOS

Prequisites

- Be using a Mac with Xcode installed

Project Setup

1. Create Cordova project without using cordova mfp template
2. `$ cordova platform add ios`
3. `$ cordova platform add plugin cordova-plugin-mfp-push`
4. `$ cordova build`
5. Open in XCode
6. Enable push notifications for your application in Project > Capabilities.
7. Use the Push SDK APIs (Refer to Push API below)

Notifications API

Client-side API for tag notifications

`MFPPush.isPushSupported()` - Returns `true` if push notifications are supported by the platform, or `false` otherwise.\

```
MFPPush.isPushSupported(function(successResponse) {
    alert("Push Supported: " + successResponse);
}, function(failureResponse) {
    alert("Failed to get push support status");
    }
);
```

`MFPPush.registerDevice(options)` - Registers devices for push notifications (?!? confirm definition ?!?)

```
MFPPush.registerDevice(settings, function(successResponse) {
    alert("Successfully registered");
    enableButtons();
}, function(failureResponse) {
    alert("Failed to register");
    }
);
```

`MFPPush.getTags()` - Gets all tags available (?!? confirm definition?!?)

```
MFPPush.getTags(function(tags) {
    alert(JSON.stringify(tags));
}, function(){
    alert("Failed to get tags");
    }
);
```

`MFPPush.subscribe(tagName,options)` - Subscribes the device to the specified tag name.

```
var tags = ['sample-tag1','sample-tag2']
MFPPush.subscribe(tags,function(tags) {
    alert("Subscribed successfully");
}, function(){
    alert("Failed to subscribe");
);
```

`MFPPush.getSubsciptions(options)` - Get tags the device is subscribed to.

```
MFPPush.getSubscriptions(function(subscriptions) {
    alert(JSON.stringify(subscriptions));
}, function(){
    alert("Failed to get subscriptions");
    }
);
```

`MFPPush.unsubscribe(tagName,options)` - Unsubscribes the device to the specified tag name.

```
var tags = ['sample-tag1','sample-tag2']
MFPPush.unsubscribe(tags, function(tags) {
    alert("Unsubscribed successfully");
}, function(){
    alert("Failed to unsubscribe");
    }
);
```

`MFPPush.unregisterDevice(options)` - Unregisters device for push notifications.

```
MFPPush.unregisterDevice(function(successResponse) {
    alert("Unregistered successfully");
}, function(){
    alert("Failed to unregister");
    }
);
```

Handling a push notification

You can handle a push notification by taking message received and creating an alert.

```
var notificationReceived = function(message) {
    alert(JSON.stringify(message));
};
```

Handling a secure push notification