Handling Push Notifications in Cordova

Overview

Before iOS, Android and Windows Cordova applications are able to receive and display push notifications, the **cordova-plugin-mfp-push** Cordova plug-in needs to be added to the Cordova project. Once an application has been configured, MobileFirst-provided Notifications API can be used in order to register & unregister devices, subscribe & unsubscribe tags and handle notifications. In this tutorial, you will learn how to handle push notification in Cordova applications.

> **Note:** In the release, authenticated notifications are **not supported** in Cordova applications due to a defect. However a workaround is provided: each `MFPPush` API call can be wrapped by `WLAuthorizationManager.obtainAccessToken("push.mobileclient").then( ... );`. The provided sample application uses this workround.

Prequisites:

- Make sure you have read the following tutorials:
  - Setting up your MobileFirst development environment (../../setting-up-your-development-environment/)
  - Adding the MobileFirst Foundation SDK to Android applications (../../adding-the-mfpf-sdk/cordova)
  - Push Notifications Overview (../)
- MobileFirst Server to run locally, or a remotely running MobileFirst Server
- MobileFirst CLI installed on the developer workstation
- Cordova CLI installed on the developer workstation

Jump to

- Notifications Configuration
- Notifications API
- Handling a push notification
- Sample application

Notifications Configuration

Create a new Cordova project or use an existing one, and add one or more of the supported platforms: iOS, Android, Windows.

> If the MobileFirst Cordova SDK is not already present in the project, follow the instructions in the Adding the MobileFirst Foundation SDK to Cordova applications (../../adding-the-mfpf-sdk/cordova) tutorial.

Adding the Push plug-in

1. From a **command-line** window, navigate to the root of the Cordova project.
2. Add the push plug-in to by running the command:

```
cordova plugin add cordova-plugin-mfp-push
```

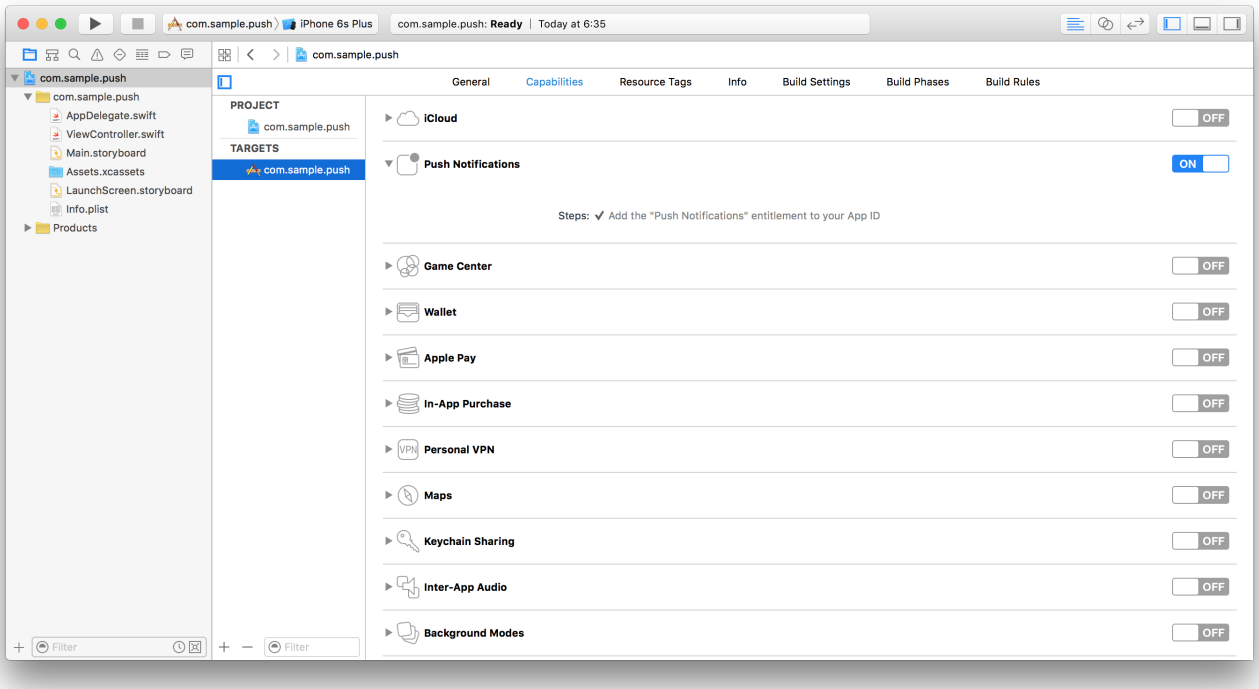
3. Build the Cordova project by running the command:

```
cordova build
```

iOS platform

The iOS platform requires an additional step.
In Xcode, enable push notifications for your application in the **Capabilities** screen.

> **Important:** the bundleId selected for the application must match the AppId that you have previously created in the Apple Developer site. See the [Push Notifications Overview] tutorial.



Notifications API

Client-side

Javascript Function	Description
`MFPPush.initialize(success, failure)`	Initialize the MFPPush instance.
`MFPPush.isPushSupported(success, failure)`	Does the device support push notifications.
`MFPPush.registerDevice(success, failure)`	Registers the device with the Push Notifications Service.
`MFPPush.getTags(success, failure)`	Retrieves all the tags available in a push notification service instance.
`MFPPush.subscribe(tag, success, failure)`	Subscribes to a particular tag.
`MFPPush.getSubsciptions(success, failure)`	Retrieves the tags device is currently subscribed to

Javascript Function	Description
`MFPPush.unsubscribe(tag, success, failure)`	Unsubscribes from a particular tag.
`MFPPush.unregisterDevice(success, failure)`	Unregisters the device from the Push Notifications Service

API implementation

Initialization

Initialize the **MFPPush** instance.

- 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.initialize (
   function(successResponse) {
      alert("Successfully intialized");
      MFPPush.registerNotificationsCallback(notificationReceived);
   },
   function(failureResponse) {
      alert("Failed to initialize");
   }
);
```

Is push supported

Check if the device supports push notifications.

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

Register device

Register the device to the push notifications service.

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

> **Note:** Due to a defect, the `options` object for **Cordova-based Android** apps must currently contain an empty value as follows: `"phoneNumber":""`. Read more about the the `available options in the user documentation.

Get tags

Retrieve all the available tags from the push notification service.

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

Subscribe

Subscribe to desired tags.

```
var tags = ['sample-tag1','sample-tag2'];

MFPPush.subscribe(
   tags,
   function(tags) {
      alert("Subscribed successfully");
   },
   function() {
      alert("Failed to subscribe");
   }
);
```

Get subscriptions

Retrieve tags the device is currently subscribed to.

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

Unsubscribe

Unsubscribe from tags.

```
var tags = ['sample-tag1','sample-tag2'];

MFPPush.unsubscribe(
   tags,
   function(tags) {
      alert("Unsubscribed successfully");
   },
   function() {
      alert("Failed to unsubscribe");
   }
);
```

Unregister

Unregister the device from push notification service instance.

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

Handling a push notification

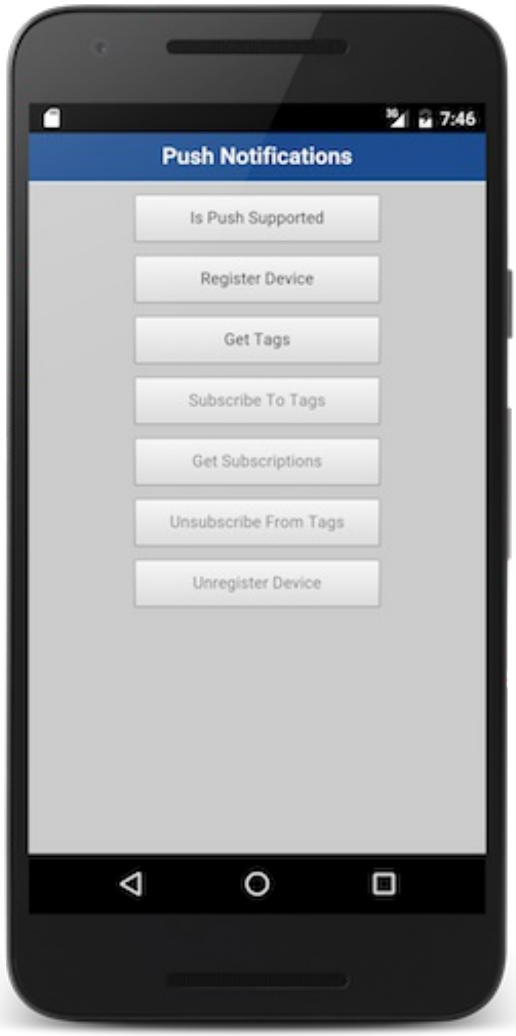
You can handle a received push notification by operating on its response object in the registered callback function.

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

Sample application

Click to download (https://github.com/MobileFirst-Platform-Developer-Center/PushNotificationsCordova/tree/release80) the Cordova project.

Note: The latest version of Google Play Services is required to be installed on any Android device for the sample to run.



Sample usage

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