Push notifications in native Windows Phone 8 applications

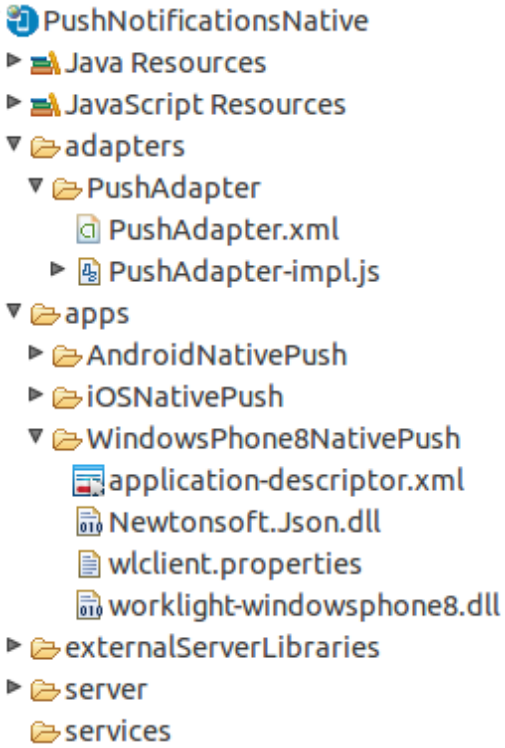
Overview

In this tutorial, the concept, API, and usage of push notifications are explained, in the context of Native Windows Phone 8 applications.

The following topics are covered:

- Setting up for push notification
- Server-side notification APIs
- Client-side notification APIs
- Tag-based notification
- Broadcast-based notification
- Sample application

Setting up your native Windows Phone 8 application for push notification



1. In MobileFirst Studio, create a MobileFirst project.
2. Add a MobileFirst Windows Phone 8 native API. The native API project provides the files that are necessary to build a Windows Phone 8 app.
3. Create an HTTP Adapter in the MobileFirst project.

In this tutorial, you set up the adapter to send and receive unauthenticated push notifications.

Edit the application-descriptor.xml file

Add the "**pushSender**" tag to the application-descriptor.xml file. -

```
<nativeWindowsPhone8App id="WindowsPhone8NativePush" platformVersion="6.3.0.00.20141111-0731"
version="1.0" xmlns="http://www.worklight.com/native-windowsphone8-descriptor" securityTest="MySecurity
Test">
   <displayName>WindowsPhone8NativePush</displayName>
   <description>WindowsPhone8NativePush</description>
   <pushSender/>
</nativeWindowsPhone8App>
```

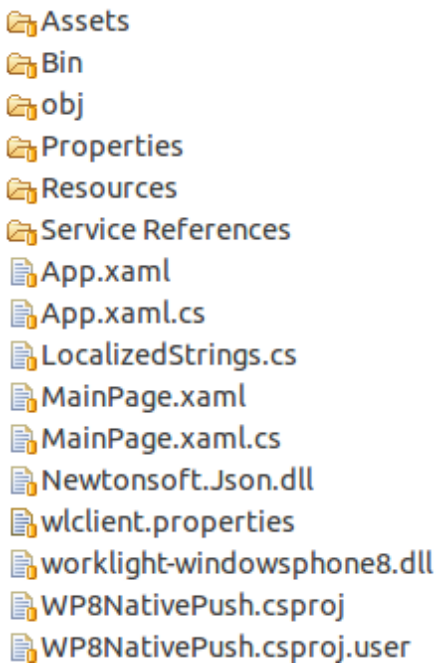
Deploy the native API and the adapter.

Copy the libraries from the MobileFirst native API to the Windows Phone 8 project.

1. Create a Visual Studio project for the Windows Phone 8 app that you are building.
2. Copy the following files from the MobileFirst project to the Windows Phone 8 project.

MobileFirst project	Windows Phone 8 project
wlclient.properties	wlclient.properties
worklight-windowsphone8.dll	worklight-windowsphone8.dll
Newtonsoft.Json.WindowsPhone.dll	Newtonsoft.Json.WindowsPhone.dll

3. After copying the libraries, add "worklight-windowsphone8.dll" and "Newtonsoft.Json.WindowsPhone.dll" as references to the Microsoft Visual Studio project. The native Windows Phone 8 project appears as shown:



Edit the wlclient.properties.

Edit the wlclient.properties file in your native Windows Phone 8 project and enter appropriate values for the following fields:

- wlServerHost - The host name or IP address of the MobileFirst Server instance.
- wlServerPort - The port on which MobileFirst Server is listening.
- wlServerContext - The context root of your MobileFirst Server instance.

```
# Licensed Materials - Property of IBM
# 5725-I43 (C) Copyright IBM Corp. 2011, 2013. All Rights Reserved.
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

wlServerProtocol = http
wlServerHost = 10.0.0.5
wlServerPort = 10080
wlServerContext = /PushNotificationsNative/
wlAppId = WindowsPhone8NativePush
wlAppVersion = 1.0
wlEnvironment = WindowsPhone8native
wlPlatformVersion = 6.3.0.0
#languagePreferences = Add locales in order of preference (e.g. fr, en, pt-BR)
#wlMPNSServiceName = Add the MPNS service name for authenticated push.
```

Modifications to the native Windows Phone 8 project

Edit the Properties\WMAppManifest.xml file and add the following capabilities:

```
<Capability Name="ID_CAP_PUSH_NOTIFICATION" />
<Capability Name="ID_CAP_IDENTITY_DEVICE" />
```

Notification API: Server side

Creating an event source.

This can be achieved by creating a notification event source in the adapter JavaScript™ code at a global level (outside any JavaScript function).

```
WL.Server.createEventSource({
   name: 'PushEventSource',
   onDeviceSubscribe: 'deviceSubscribeFunc',
   onDeviceUnsubscribe: 'deviceUnsubscribeFunc',
   securityTest:'PushApplication-strong-mobile-securityTest
'
});
```

- `name` – A name by which the event source is referenced.
- `onDeviceSubscribe` – An adapter function that is called when the request for user subscription is received.
- `onDeviceUnsubscribe` – An adapter function that is called when the request for user unsubscription is received.

- `securityTest` – A security test from the authenticationConfig.xml file, which is used to protect the event source.

Sending a notification

Notifications can be either pulled from, or pushed by, the back-end system. In this example, a `submitNotifications()` adapter function is invoked by a back-end system as an external API to send notifications. Windows Phone 8 supports Raw, Tile, and Toast notifications. To set up Toast notifications to Windows Phone 8 devices, set the appropriate JSON object to the response of `WL.Server.createDefaultNotification()` API.

```
function submitNotification(userId, notificationText) {
    var userSubscription = WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userI
d);
    if (userSubscription === null) {
        return { result: "No subscription found for user :: " + userId };
    }
    var badgeDigit = 1;
    var notification = WL.Server.createDefaultNotification(notificationText, badgeDigit, {custom:"data"});
    //Sending Toast notifications to Windows Phone 8 devices can be achieved by setting the appropriate JSO
N
 //to the response of WL.Server.createDefaultNotification() API.<br />
  /*
    notification.MPNS.toast = {
      text1: 'Meeting ',
      text2 : "Your meeting starts in 5 minutes"
    }; */
 //Send Tile notifications
    /*
    notification.MPNS.tile = {
 title : "Meeting in 5 ",
 count : 1
    };*/
    notification.MPNS.raw = {
 payload : {payload : "You have a meeting in 5 minutes"}
    };
    WL.Server.notifyAllDevices(userSubscription, notification);
    return {
        result: "Notification sent to user :: " + userId
    };
}
```

Notification API - Client side

The first step is to create an instance of the `WLClient` class:

```
WLClient client = WLClient.getInstance();
```

You derive all push notification operations from the `WLPush` class.

`getPush` - This method is used to retrieve an instance of the `WLPush` class from the `WLClient` instance.

```
WLPush push = client.getPush();
```

`onReadyToSubscribeListener` - When connecting to a MobileFirst Server instance, the application attempts to register itself with the MPNS server to receive push notifications.

```
push.onReadyToSubscribeListener = onReadyListener;
client.connect(connectListener);
```

`notificationListener` - Then the app receives the notification.

```
push.notificationListener = notificationListener;
client.connect(connectListener);
```

The `onReadyToSubscribe` method of `WLOnReadyToSubscribeListener` is called when registration is complete.

```
public void onReadyToSubscribe() { ..... }
```

`WLPush.registerEventSourceCallback`

To register an alias on a particular event source, use the `WLPush.registerEventSourceCallback` method. The API takes the following arguments:

alias - An alias name.
Adaptername - The adapter in which the event source is defined.
EventSourceName - The event source on which the alias is called.

Example:

```
WLClient.getInstance().getPush().registerEventSourceCallback("newalias", "PushAdapter", "PushEventSourc
e", this);
```

Typically, this method is called in the `onReadyToSubscribe` callback function.

```
public void onReadyToSubscribe() {
   WLClient.getInstance().getPush().registerEventSourceCallback("newalias","PushAdapter","PushEventSou
rce",this);
}
```

Subscribing to push notifications

To set up subscription to push notifications, use the `WLPush.subscribe(alias, pushOptions, responseListener)` API.

The API takes the following arguments:

`alias` - The alias to which the device must subscribe.
`pushOptions` - An object of type `WLPushOptions`
`responseListener` - An object of type `WLResponseListener`, which is called when subscription completes.

Example:

```
WLClient client = WLClient.getInstance(); client.getPush().subscribe("newalias",new WLPushOptions(), new MyListener());
```

`MyListener` implements `WLResponseListener` and provides the following callback functions:

`onSuccess` - Called when subscription succeeds.
`onFailure` - Called when subscription fails.

Unsubscribing from push notifications

To set up unsubscription from push notifications, use the `WLPush.unsubscribe(alias, responseListener)` API.
The API takes the following arguments:

`alias` - The alias to which the device has subscribed.
`responseListener` - An object of type `WLResponseListener`, which is called when unsubscription completes.

```
WLClient client = WLClient.getInstance(); client.getPush().unsubscribe("newalias",new MyListener());
```

`MyListener` implements `WLResponseListener` and provides the following callback functions:

`onSuccess` - Called when unsubscription succeeds.
`onFailure` - Called when unsubscription fails.

Additional client-side API methods:

`isSubscribed()` - Indicates whether the device is subscribed to push notifications.

```
WLClient client = WLClient.getInstance(); boolean bIsSubscribed = client.getPush().isSubscribed("newalias");
```

Receiving a push notification

When a push notification is received, the `onReceive` method is called on an `WLEventSourceListener` instance.

```
class MyListener : WLOnReadyToSubscribeListener , WLEventSourceListener
{
```

The `WLEventSourceListener` instance is registered during the `registerEventSourceCallback` callback.

```
WLClient.getInstance().getPush().registerEventSourceCallback("newalias", "PushAdapter","PushEventSourc
e", this );
```

The `onReceive` method displays the received notification on the screen.

```
public void onReceive(String props, String payload) {
  Deployment.Current.Dispatcher.BeginInvoke(() =>
  {
    MessageBox.Show("Push notification received " + payload)
;
  });
}
```

If the application is not running, the notification icon appears on the notification bar at the top of the screen.

Notification API - Tag-based notification

This notification type enables sending and receiving messages by tags. Tags represent topics of interest to the user and provide the ability to receive notifications according to the chosen interest. Tags are defined in the application-descriptor.xml file.

```
<tags>
  <tag>
    <name>PushTag1</name>
    <description>About pushTag1</description>
  </tag>
  <tag>
    <name>PushTag2</name>
    <description>About pushTag2</description>
  </tag>
</tags>
```

Client-side API:

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

Notification API - Broadcast notification

Broadcast notifications are enabled by default for any push-enabled MobileFirst application. A subscription to a reserved tag, `Push.ALL`, is created for every device. Broadcast notifications can be disabled by unsubscribing to the reserved tag `Push.ALL`.

For more information about broadcast notification, see the topic about broadcast notification in the user documentation.

Common APIs for tag-based and broadcast notifications

Client-side API:

`WLNotificationListener` defines the callback method to be notified when the notification arrives.

```
client.getPush().setWLNotificationListener(listener)
```

The `onMessage(props,payload)` method of `WLNotificationListener` is called when a push notification is received by the device.

> **props** - A JSON block that contains the notifications properties of the platform.
> **payload** - A JSON block that contains other data that is sent from MobileFirst 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`.

Server-side API:

`WL.Server.sendMessage(applicationId,notificationOptions)` method takes two mandatory parameters:

> **applicationId** - (mandatory) The name of the MobileFirst application.
> **notificationOptions** - (mandatory) A JSON block that contains message properties.

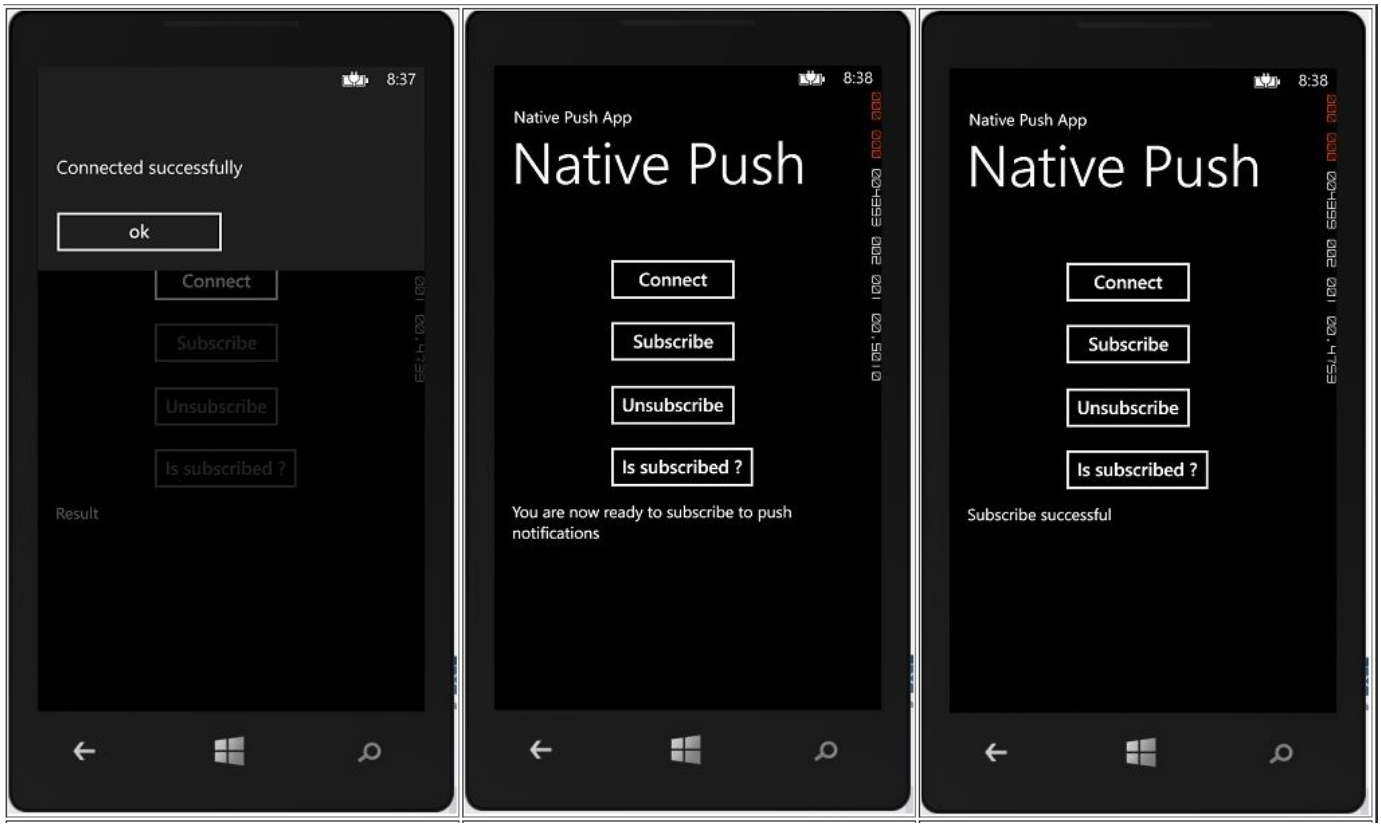
This API submits a notification that is based on the specified target parameters. For a full list of message properties, see the user documentation.

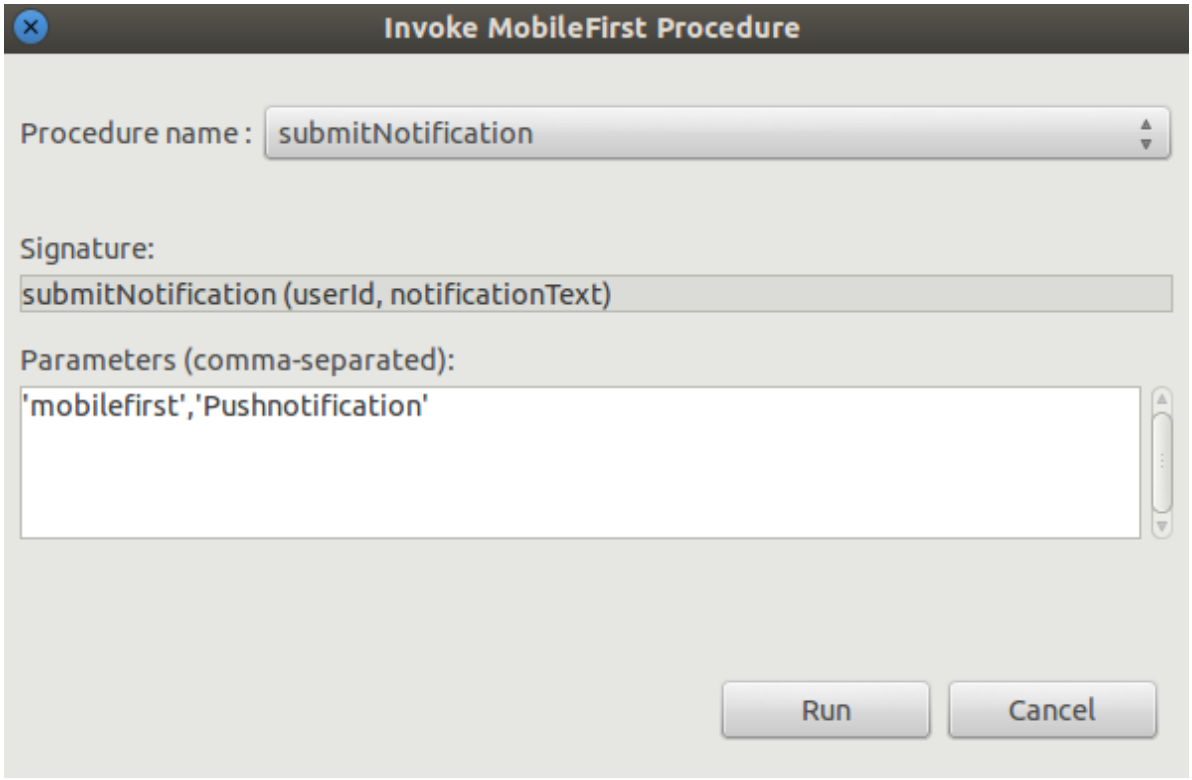
Sample application

Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/PushNotificationsNativeProject.zip) the studio project.
Click to download (http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/WP8NativePushProject.zip) the Native project.

The sample contains two projects:
-The **PushNotificationsNativeProject.zip** file contains a MobileFirst native API that you can deploy to your MobileFirst server.
-The **WP8NativePushProject.zip** file contains a native Windows Phone 8 application that uses a MobileFirst native API library to subscribe for push notifications and receive notifications from MPNS.
Make sure to update the wlclient.properties file in WP8NativePush with the relevant server settings.

To run the native Windows Phone 8 application, use the Microsoft Visual Studio IDE.


8:37
Connected successfully
ok
Connect
Subscribe
Unsubscribe
Is subscribed ?
Result
8:38
Native Push App
Native Push
Connect
Subscribe
Unsubscribe
Is subscribed ?
You are now ready to subscribe to push notifications
8:38
Native Push App
Native Push
Connect
Subscribe
Unsubscribe
Is subscribed ?
Subscribe successful


Invoke MobileFirst Procedure
Procedure name :
submitNotification
Signature:
submitNotification (userId, notificationText)
Parameters (comma-separated):
'mobilefirst','Pushnotification'
Run
Cancel

Raw notification **Tile notification** **Toast notification**