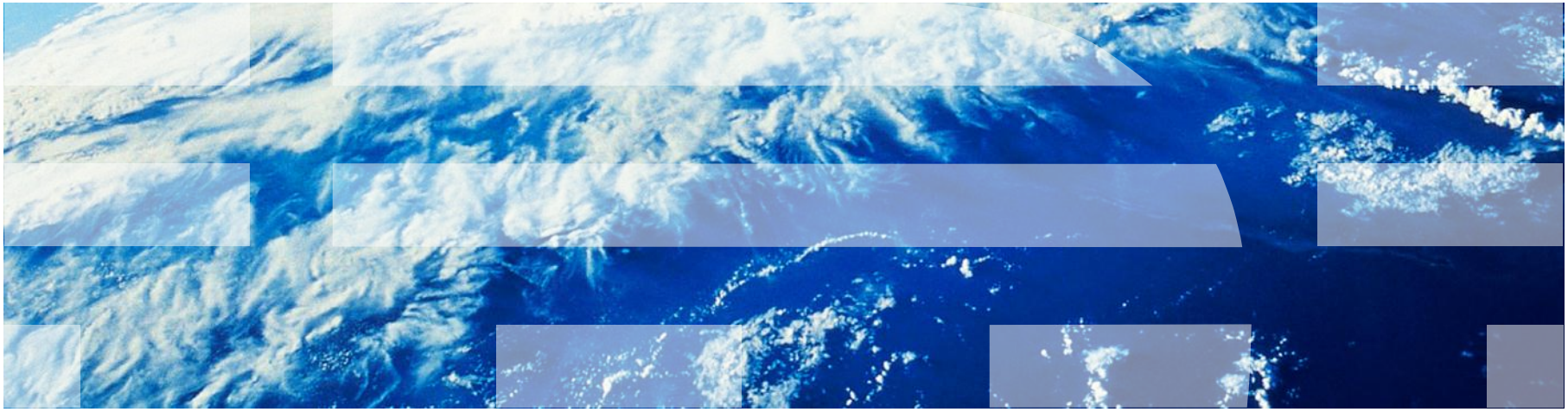
IBM Worklight Foundation V6.2.0 Getting Started

Push notifications in native Android applications



30 June 2014

Trademarks

- IBM, the IBM logo, ibm.com, and Worklight are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Other company products or service names may be trademarks or service marks of others.
- This document may not be reproduced in whole or in part without the prior written permission of IBM.

About IBM®

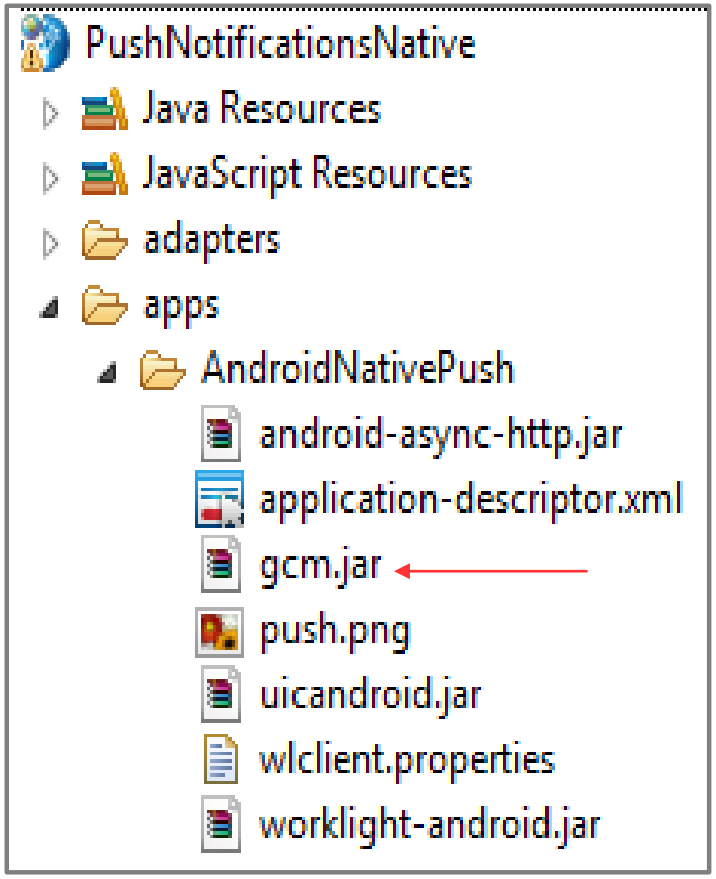
- See http://www.ibm.com/ibm/us/en/

Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

Setting up your Worklight project (1 of 3)

- IBM Worklight® Foundation provides the ability for native Android applications to subscribe, unsubscribe, and receive push notifications.
- Create a Worklight project and add a native API for Android.
- The native API includes the following files:
  - The **gcm.jar** file contains classes that are necessary for the Android application to register with Google Cloud Messaging (GCM).
  - The **push.png** is an icon file that is displayed when a push notification arrives.
- For more information about how to set up push notifications, see the IBM Worklight Foundation user documentation.



Setting up your Worklight project (2 of 3)

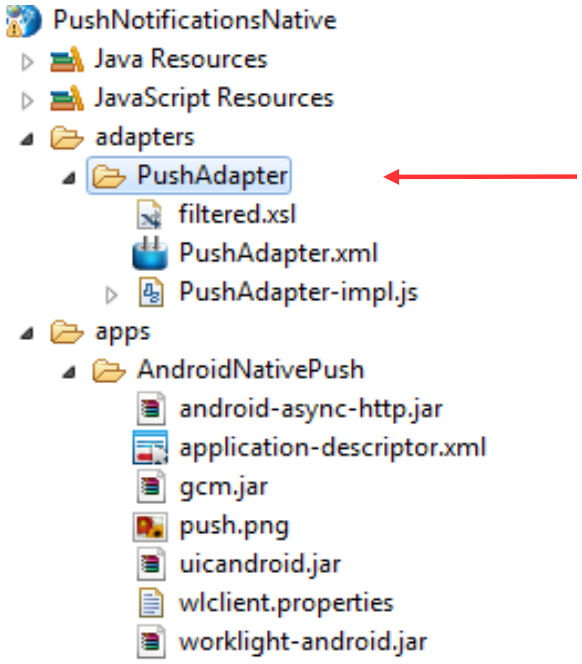
- Edit the `application-descriptor.xml` file and add your Google API server key and project ID to the `<pushSender>` tag.

```
<nativeAndroidApp xmlns="http://www.worklight.com/native-android-descriptor" id="AndroidNativePush" platformVersion="6.0.0" securityTest="MySecu
    <displayName>AndroidNativePush</displayName>
    <description>AndroidNativePush</description>
    <pushSender key="SOME-GCM-KEY" senderId="SOME-GCM-ID"/>
    <publicSigningKey>Replace this text with the public key of the certificate with which you sign the APK. For details see the Worklight Develo
</nativeAndroidApp>
```

- If you do not have an API key, get one from the following URL: https://code.google.com/apis/console

Setting up your Worklight project (3 of 3)

- Next, add an adapter with an `eventSource` in it.




```
WL.Server.createEventSource({
    name: 'PushEventSource',
    onDeviceSubscribe: 'deviceSubscribeFunc',
    onDeviceUnsubscribe: 'deviceUnsubscribeFunc'
});

function deviceSubscribeFunc(userSubscription, deviceSubscription){
    WL.Logger.debug(">> deviceSubscribeFunc");
}

function deviceUnsubscribeFunc(userSubscription, deviceSubscription){
    WL.Logger.debug(">> deviceUnsubscribeFunc");
}
```

- Deploy the native API and the adapter.

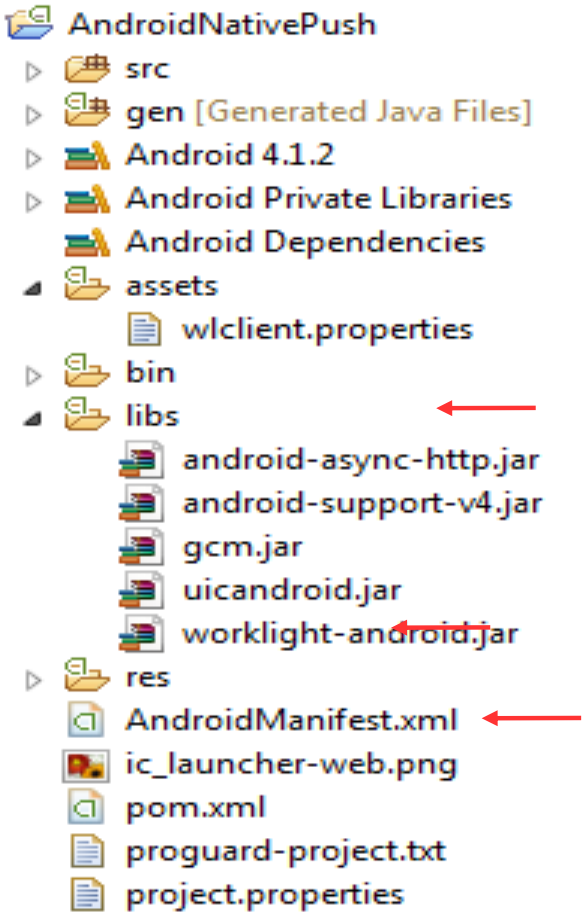
Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

Setting up your native app for push notifications (1 of 4)

- Create an Android Activity Project and copy the following files from the Worklight project to the Android project.

Worklight project	Android Activity Project
wlclient.properties	assets/wlclient.properties
gcm.jar	libs/gcm.jar
worklight-android.jar	libs/worklight-android.jar
android-async-http.jar	libs/android-async-http.jar
push.png	drawable*/push.png



Setting up your native app for push notifications (2 of 4)

- Edit the `wlclient.properties` file in your native Android project and enter appropriate values for the following fields:
  - **wlServerHost** – Hostname or IP address of the Worklight Server.
  - **wlServerPort** – Port on which the Worklight Server is listening.
  - **wlServerContext** – Context root of your Worklight Server.
  - **GcmSenderId** – The project number of your project that you obtained through Google API console.

```
wlServerProtocol = http
wlServerHost = 169.254.25.40
wlServerPort = 10080
wlServerContext = /PushNotificationsNative/
wlAppId = AndroidNativePush
wlAppVersion = 1.0
wlEnvironment = Androidnative
wlUid = wY/mbnwKTDDYQUvuQCdSgg==
wlPlatformVersion = 6.2.0.00.20140520-1620
#languagePreferences = Add locales in order of preference (e.g. en, fr, fr-CA)
#For Push Notifications,uncomment below line and assign value to it
#GcmSenderId =
```

Setting up your native app for push notifications (3 of 4)

- Add the following permissions to the `AndroidManifest.xml` file of your Android project:

```
<permission android:name="com.worklight.androidnativepush.permission.C2D_MESSAGE" android:protectionLevel="signature" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="com.worklight.androidnativepush.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

- Add the `launchMode` attribute to the main `AndroidNativePush` activity. Set its value to `singleTask`.

```
<activity
    android:name="com.worklight.androidnativepush.AndroidNativePush"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar"
    android:launchMode="singleTask">
```

Setting up your native app for push notifications (4 of 4)

- Add an `intent-filter` to the main `AndroidNativePush` activity for notifications.

```
<intent-filter>
    <action android:name="com.worklight.androidnativepush.AndroidNativePush.NOTIFICATION" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

- Add the `GCMIntentService` and add an `intent-filter` for RECEIVE and REGISTRATION of notifications.

```
<service android:name="com.worklight.wlclient.push.GCMIntentService" />

<receiver android:name="com.worklight.wlclient.push.WLBroadcastReceiver" android:permission="com.google.android.c2dm.permission.SEND">
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.worklight.androidnativepush" />
    </intent-filter>

    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.worklight.androidnativepush" />
    </intent-filter>
</receiver>
```

Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

The native push notification API

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

```
final WLClient client = WLClient.createInstance(this);
```

- You do all push notification operations from the `WLPush` class.
  - `getPush` – Use this method to retrieve an instance of the `WLPush` class from the `WLClient` instance.
    `WLPush push = WLClient.getPush();`
  - `WLOnReadyToSubscribeListener` – When connecting to a Worklight Server, the app attempts to register itself with the GCM server to receive push notifications.

```
client.getPush().setOnReadyToSubscribeListener(listener);
client.connect(listener);
```

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

```
@Override
public void onReadyToSubscribe() {
    WLClient.getInstance().getPush().registerEventSourceCallback("myAndroid", "PushAdapter","PushEventSource", this );
```

WLPush.registerEventSourceCallback

- Use the `WLPush.registerEventSourceCallback` method to register an alias on a particular event source.

```
WLClient.getInstance().getPush().registerEventSourceCallback("myAndroid", "PushAdapter","PushEventSource", this );
```

- The statement takes the following parameters:
  - `myAndroid` is the alias name.
  - `PushEventSource` is the event source on which the alias called.
  - `PushAdapter` is the adapter in which the event source is defined.
- Typically, this method is called in the `onReadyToSubscribe` callback function.

Other WLPush methods

- **`isPushSupported()`**
  - Indicates whether push notifications are supported by the device.

```
WLClient client = WLClient.getInstance();
boolean supported = client.getPush().isPushSupported();
```

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

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

Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The Native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

Override Android activity life cycle methods

- In your activity class, override the methods that define Android activity life cycle as follows:
  - **onPause()** must call the setForeground(false) method of the WLPush instance to receive the notification in the notification bar when the app is paused.

```
@Override
protected void onPause() {
    super.onPause();
    if (push != null)
        push.setForeground(false);
}
```

  - **onResume()** must call the setForeground(true) method of the WLPush instance to receive the notification in the callback of the app.

```
@Override
protected void onResume() {
    super.onResume();
    if (push != null)
        push.setForeground(true);
}
```

  - **onDestroy()** must call the unregisterReceivers method of the WLPush instance to avoid leak exceptionss from the receiver when the app exits.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (push != null)
    push.unregisterReceivers();
}
```

Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

Subscribing to push notifications

- **`WLPush.subscribe(alias, pushOptions, responseListener)`**
  - `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 `subscribe` completes.

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

- **`MyListener.java`**
  - Implements `WLResponseListener`.
  - `onSuccess` – Called when `subscribe` succeeds.
  - `onFailure` – Called when `subscribe` fails.

Unsubscribing from push notifications

- **`WLPush.unsubscribe(alias, responseListener)`**
  - `alias` – The alias to which the device has subscribed.
  - `responseListener` – An object of type `WLResponseListener`, which is called when `subscribe` completes.

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

- **`MyListener.java`**
  - Implements `WLResponseListener`.
  - `onSuccess` – Called when `unsubscribe` succeeds.
  - `onFailure` – Called when `unsubscribe` fails.

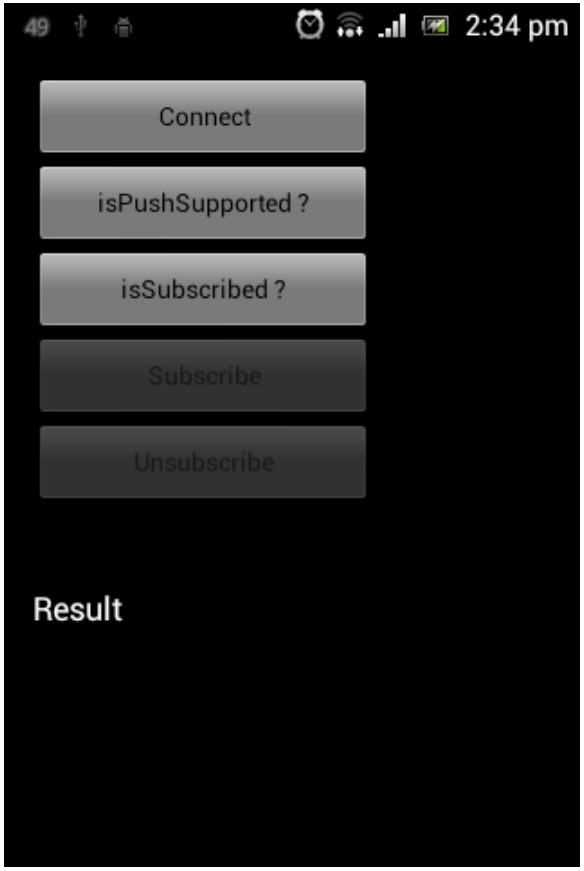
Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

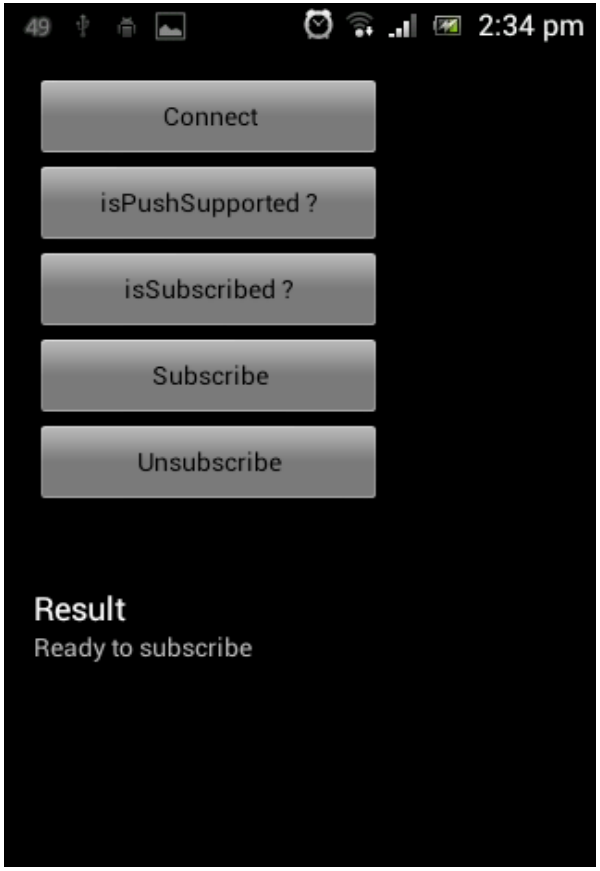
Running your Android project

- Right-click your Android project and choose **Run As > Android Application**.

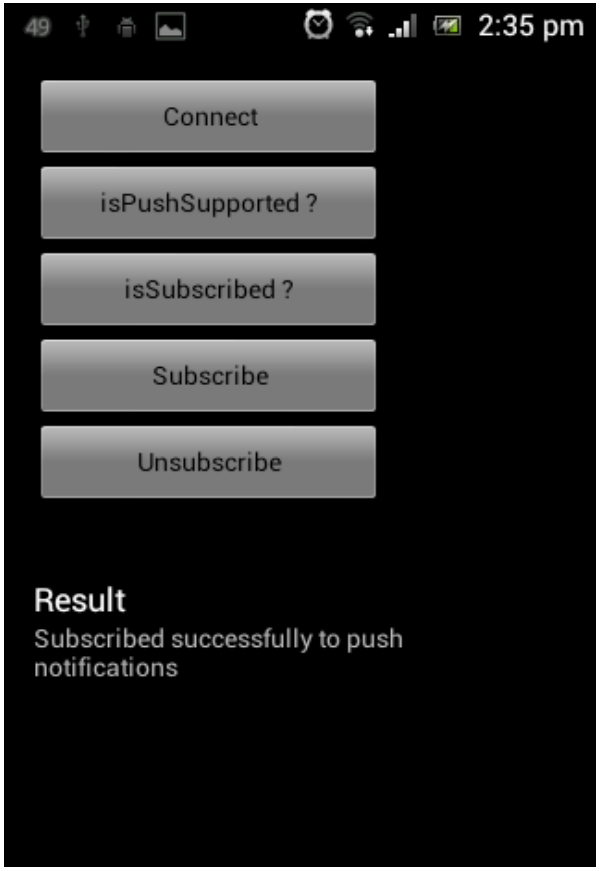
Click **Connect**.



Wait until onReadyToSubscribe listener is called.



Then click **Subscribe**.



Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

Sending a push notification (1 of 2)

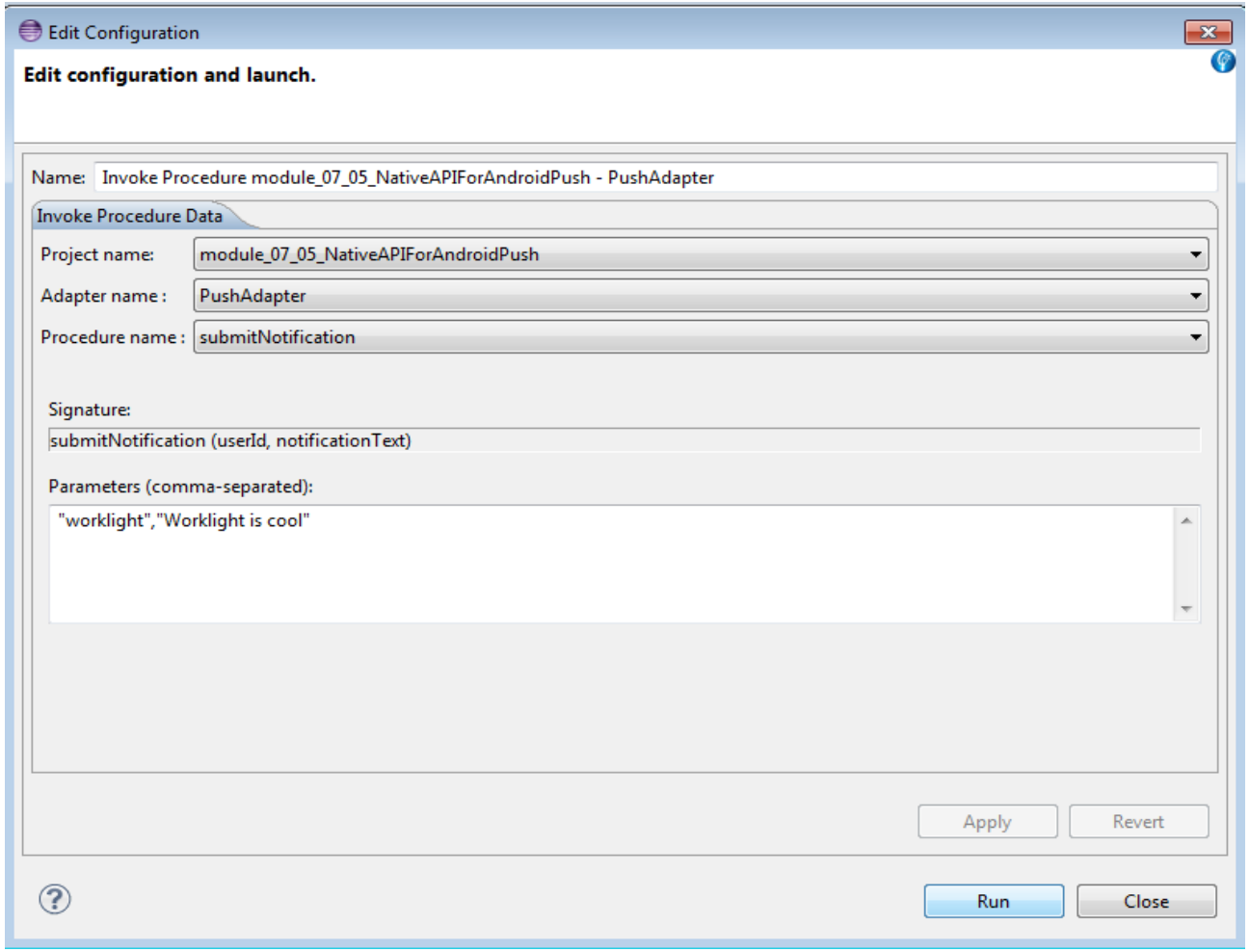
- The `PushAdapter` object has a `submitNotification` procedure.

```
function submitNotification(userId, notificationText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('PushAdapter.PushEventSource', userId);

    if (userSubscription==null){
        return { result: "No subscription found for user :: " + userId };
    }

    WL.Logger.debug("submitNotification >> userId :: " + userId + ", text :: " + notificationText);

    WL.Server.notifyAllDevices(userSubscription, {
        badge: 1,
        sound: "sound.mp3",
        activateButtonLabel: "ClickMe",
        alert: notificationText,
        payload: {
            foo : 'bar'
        }
    });

    return { result: "Notification sent to user :: " + userId };
}
```

- Right-click your adapter and select **Run As > Invoke Worklight Procedure**.

Sending a push notification (2 of 2)

- Call `submitNotification` to send a push notification.



Receiving a push notification (1 of 3)

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

```
public class MyListener implements WLOnReadyToSubscribeListener, WLResponseListener, WLEventSourceListener{
```

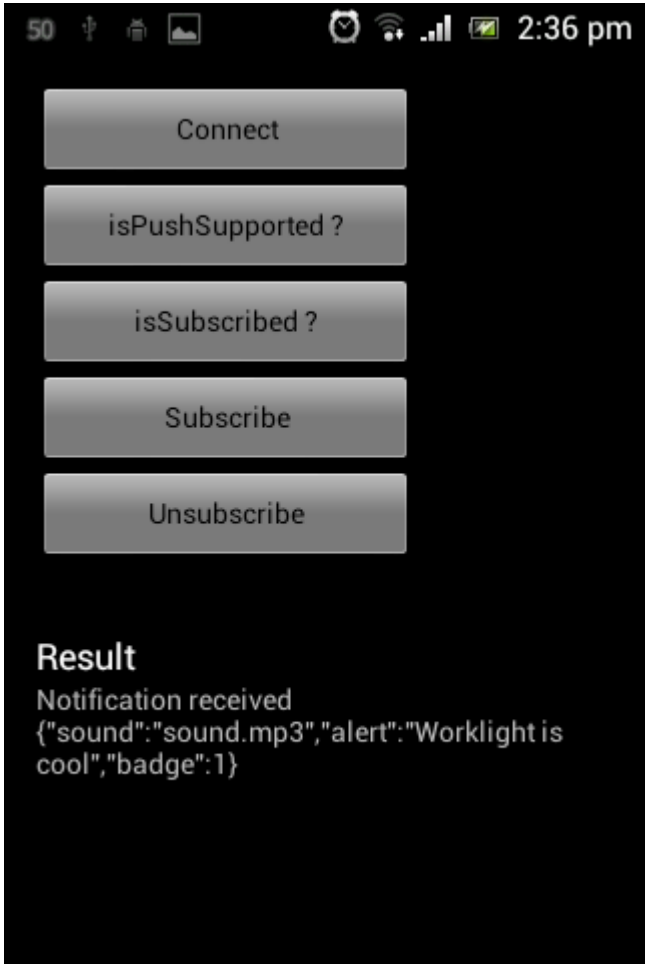
```
@Override
public void onReceive(String arg0, String arg1) {
    AndroidNativePush.updateTextView("Notification received  " + arg0);
}
```

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

```
WLClient.getInstance().getPush().registerEventSourceCallback("myAndroid", "PushAdapter","PushEventSource", this );
```

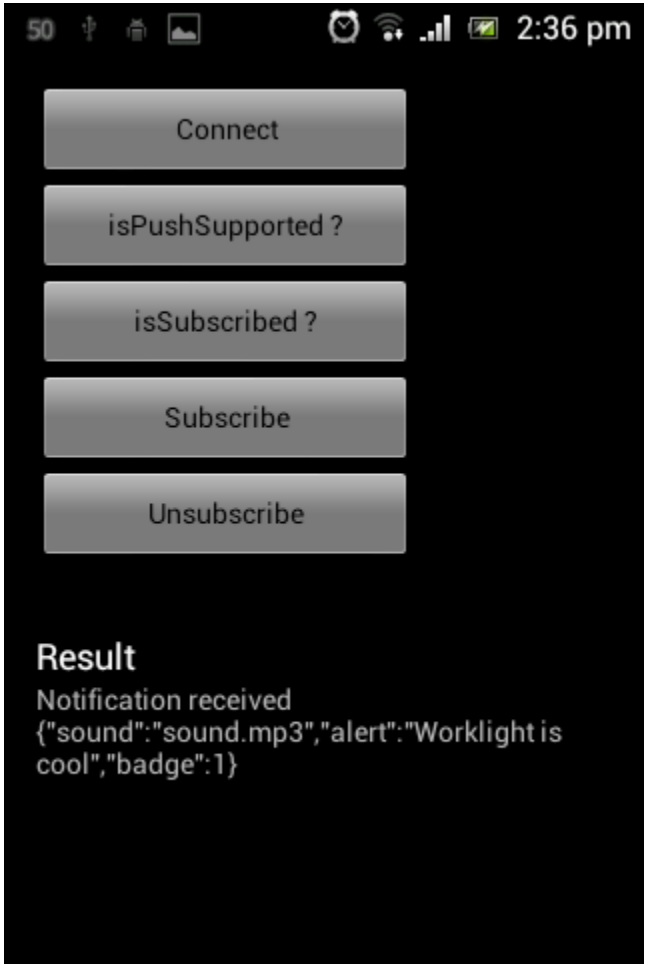
Receiving a push notification (2 of 3)

- The `onReceive` method displays the received notification on the screen.
- If the app is not running, the notification icon appears on the notification bar at the top of the screen.



Receiving a push notification (3 of 3)

- You can find the sample for this training module in the Getting Started page of the IBM Worklight documentation website at http://www.ibm.com/mobile-docs.
- The sample contains two projects:
  - **PushNotificationsNative.zip** contains a Worklight native API to be deployed to your Worklight Server.
  - **AndroidNativePush.zip** contains a native Android application that uses a Worklight native API library to communicate with Worklight Server.



Agenda

- Setting up your Worklight project
- Setting up your native app for push notifications
- The native push notification API
- Override the methods of the Android activity life cycle
- Subscribing and unsubscribing to push notifications
- Running your Android project
- Sending and receiving push notifications
- Tag-based and broadcast notification

Tag-based notification

- Tags represent topics of interest to the user and provide the ability to receive notifications according to the chosen interest.
- This notification type enables devices to send and receive messages that are filtered by tags.
- To start receiving tag-based notifications, the device must first subscribe to a push notification tag in an application.
- Tags are defined in *`application-descriptor.xml` file*:

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

- This notification is targeted to all the devices that have subscribed to a tag in an application.

Tag-based notification

- 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.

Tag-based notification

For more information about tag-based notification, see Tag-based notification in IBM Worklight Foundation user documentation.

Broadcast notification

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

Broadcast notification

For more information about broadcast notification, see Broadcast notifications in IBM Worklight Foundation user documentation.

Common APIs for tag-based and broadcast notifications (1 of 2)

- 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 the Worklight Server. This block 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`.

Common APIs for tag-based and broadcast notifications (2 of 2)

- Server-side API:
  - `WL.Server.sendMessage(applicationId,notificationOptions)`
    This method takes two mandatory parameters:
    - `applicationId` - The name of the Worklight application.
    - `notificationOptions` - A JSON block that contains message properties.

  Submits a notification that is based on the specified target parameters.
  - For a full list of message properties, see the IBM Worklight Foundation user documentation.

Notices

- Permission for the use of these publications is granted subject to these terms and conditions.
- This information was developed for products and services offered in the U.S.A.
- IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.
- IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
  - IBM Director of Licensing
    IBM Corporation
    North Castle Drive
    Armonk, NY 10504-1785
    U.S.A.
- For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
  - Intellectual Property Licensing
    Legal and Intellectual Property Law
    IBM Japan Ltd.
    1623-14, Shimotsuruma, Yamato-shi
    Kanagawa 242-8502 Japan
- **The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.
- This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.
- Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
- IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
- Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
- The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.
- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

COPYRIGHT LICENSE:

- This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.
- Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
  - © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Privacy Policy Considerations

- IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.
- Depending upon the configurations deployed, this Software Offering may use session cookies that collect session information (generated by the application server). These cookies contain no personally identifiable information and are required for session management. Additionally, persistent cookies may be randomly generated to recognize and manage anonymous users. These cookies also contain no personally identifiable information and are required.
- If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent. For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the sections entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Support and comments

- For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:
  - http://www.ibm.com/mobile-docs
- **Support**
  - Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:
    - http://www.ibm.com/software/passportadvantage
  - If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:
    - http://www.ibm.com/support/handbook
- **Comments**
  - We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.
  - For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.
  - When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.
  - Thank you for your support.
  - Submit your comments in the IBM Worklight Developer Edition support community at:
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - If you would like a response from IBM, please provide the following information:
    - Name
    - Address
    - Company or Organization
    - Phone No.
    - Email address

Thank You