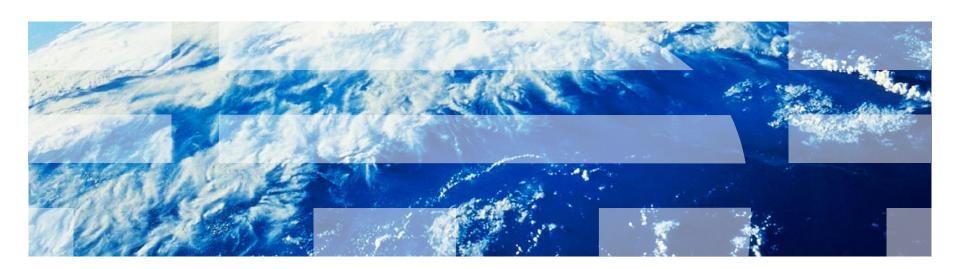# IBM Worklight Foundation V6.2.0
Getting Started

SMS notifications

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.

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

- 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

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

What are SMS notifications?

- SMS notification is the ability of a mobile device to receive notifications as SMS messages that are *pushed* from a server.

- Notifications are received regardless of whether the application is running.
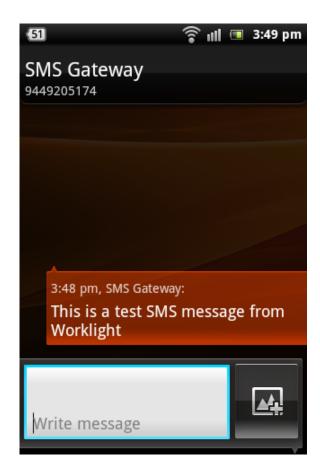
Agenda

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

Device Support

- IBM® Worklight Foundation ® supports SMS notifications on iOS, Android, Windows Phone, and BlackBerry devices that support SMS functions.

Agenda

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

Notification architecture: Terminology

- **Event source**: A notification channel to which mobile applications can subscribe.

  - An event source is defined within a Worklight adapter.

- **User ID**: A unique identifier for a Worklight user

  - A user ID is obtained through authentication or through another unique identifier such as a persistent cookie.

- **Application ID**: Worklight application ID

  - This ID identifies a specific Worklight application on the mobile market.

Notification architecture: Subscription

- To start receiving SMS notifications, an application must first subscribe to an SMS notification *event source*.

- An event source is declared in the Worklight adapter that is used by the application for SMS notification services.

- To subscribe to SMS notifications, the user supplies a mobile phone number and approves the notification subscription.

- A subscription request is sent to the Worklight Server upon receipt of the user approval.

Notification architecture: Sending notifications
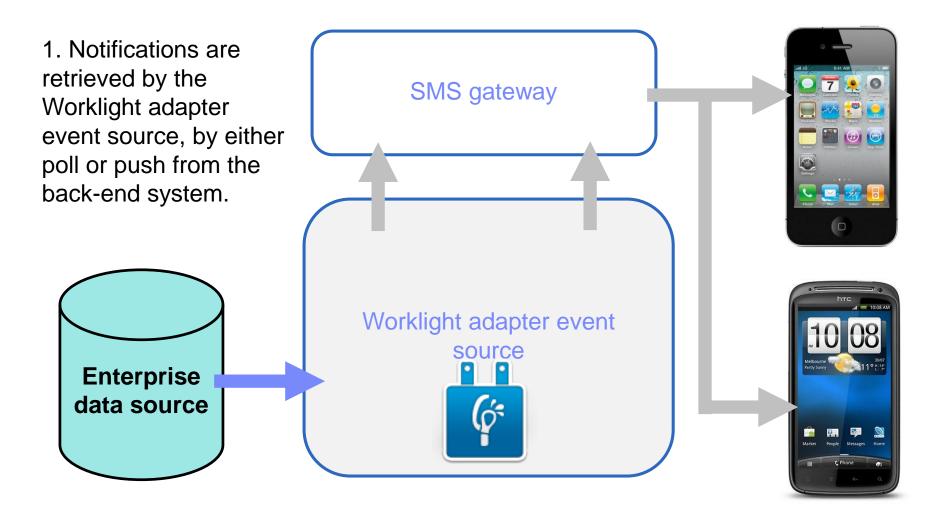
- Worklight provides a unified push notification API.

- With the adapter API, you can:

  - Manage subscriptions

  - Push and poll notifications from back-end systems

  - Send push notifications to devices

- With the application API, you can subscribe to, and unsubscribe from, push notification event sources.

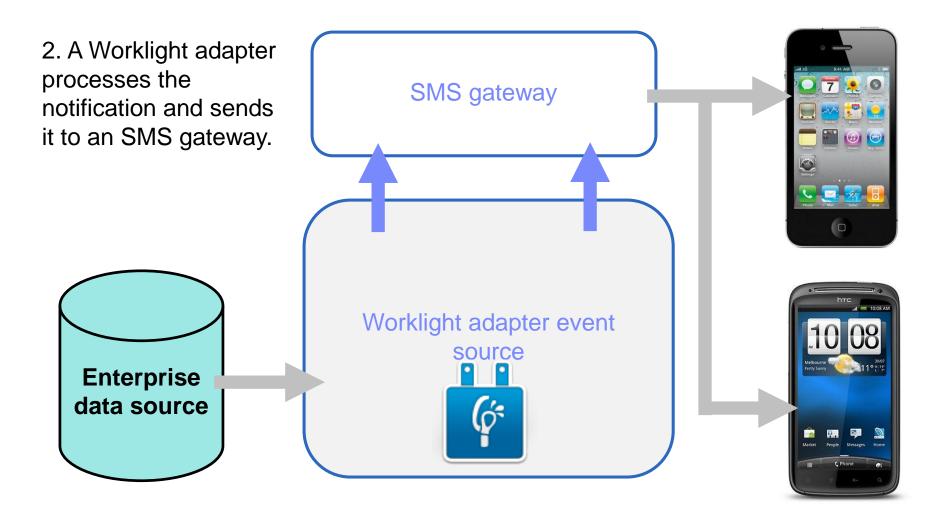# *Notification architecture: Sending notifications*

- To send a notification, you must first retrieve it from the back-end system.

- An event source can either poll notifications from the back-end system, or wait for the back-end system to explicitly push a new notification.

- When a notification is retrieved by the adapter, it is processed and sent through a preconfigured SMS gateway.

- Extra custom logic can be added to the adapter to preprocess notifications.

- The SMS gateway receives the notification and sends it to a device.

Notification architecture: Sending notifications

1. Notifications are retrieved by the Worklight adapter event source, by either poll or push from the back-end system.

SMS gateway

Worklight adapter event source

Enterprise data source

Notification architecture: Sending notifications

2. A Worklight adapter processes the notification and sends it to an SMS gateway.

SMS gateway

Worklight adapter event source

Enterprise data source

Notification architecture: Sending notifications

3. The SMS gateway sends a push notification to the device.

SMS gateway

Worklight adapter event source

Enterprise data source

Notification architecture: Sending notifications

4. The device processes the push notification as an SMS message.

SMS gateway

Worklight adapter event source

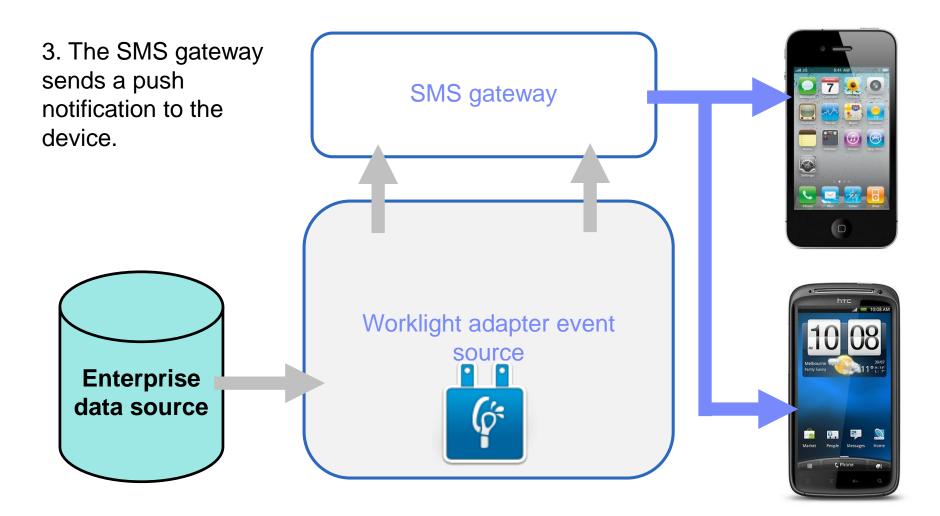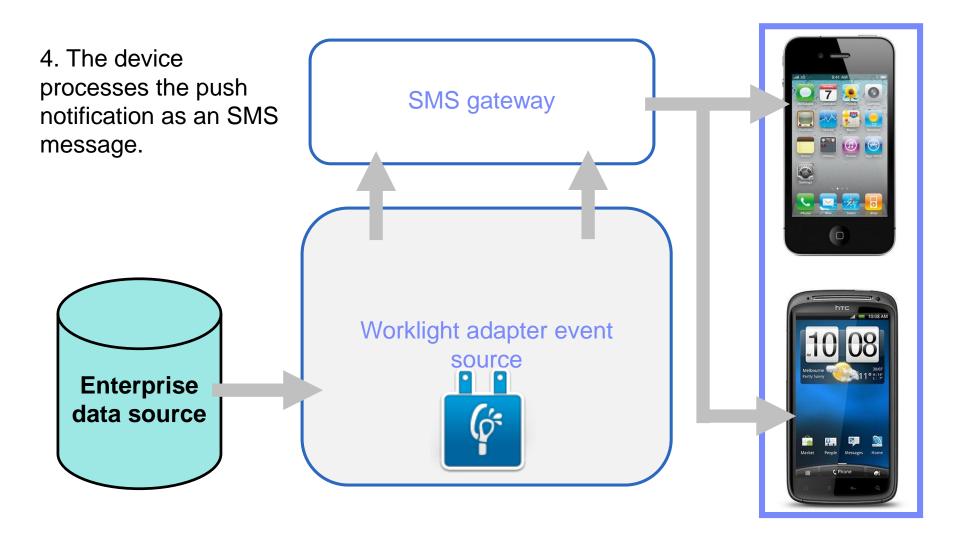**Enterprise data source**

Agenda

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

Subscription management: User subscription

- **User subscription**
  - An entity that contains a user ID, device ID, and event source ID
  - It represents the intent of the user to receive notifications from a specific event source.

- **Creation**
  - The user subscription for an event source is created when the user subscribes to that event source for the first time from any device.

- **Deletion**
  - A user subscription is deleted when the user unsubscribes from that event source from all of their devices.

- **Notification**
  - While the user subscription exists, the Worklight Server can produce notifications for the subscribed user.
  - These notifications can be delivered by the adapter code to some or all of the subscribed devices.

Subscription management: Device subscription

- A device subscription belongs to a user subscription, and exists in the scope of a specific user and event source. A user subscription can have several device subscriptions.

- The device subscription is created when the application on a device calls **WL.Client.Push.subscribeSMS**. The mobile phone number is an input parameter.

- The device subscription is deleted either when an application calls **WL.Client.Push.unsubscribeSMS**, or when the administrator unsubscribes the user through the Worklight Console.

Agenda

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

Notification API: Server side

- Start by creating an event source
  - Declare a notification event source in the adapter JavaScript code at a global level (outside any JavaScript function).

Notifications are pushed by the back-end system

```
WL.Server.createEventSource({
    name: 'SMSEventSource',
    onDeviceSubscribe: 'onDeviceSubscribeCallback',
    onDeviceUnsubscribe: 'onDeviceUnsubscribeCallback',
    securityTest:'SMSRealm-mobile-securityTest'
});
```

Notifications are polled from the back-end system

```
WL.Server.createEventSource({
    name: 'SMSEventSource',
    onDeviceSubscribe: 'onDeviceSubscribeCallback',
    onDeviceUnsubscribe: 'onDeviceUnsubscribeCallback',
    securityTest:'SMSRealm-mobile-securityTest',
    poll:{
        interval: 3,
        onPoll: getNotificationsFromBackend
    }
});
```

- **name** – name by which the event source is referenced
- **onDeviceSubscribe** – adapter function that is called when the user subscription request is received
- **onDeviceUnsubscribe** – adapter function that is called when the user unsubscription request is received
- **securityTest** – security test from authenticationConfig.xml that is used to protect the event source

Notification API: Server side

- Start by creating an event source

  - Declare a notification event source in the adapter JavaScript code at a global level (outside any JavaScript function).

Notifications are pushed by the back-end system

```
WL.Server.createEventSource({
    name: 'SMSEventSource',
    onDeviceSubscribe: 'onDeviceSubscribeCallback',
    onDeviceUnsubscribe: 'onDeviceUnsubscribeCallback',
    securityTest:'SMSRealm-mobile-securityTest'
});
```

Notifications are polled from the back-end system

```
WL.Server.createEventSource({
    name: 'SMSEventSource',
    onDeviceSubscribe: 'onDeviceSubscribeCallback',
    onDeviceUnsubscribe: 'onDeviceUnsubscribeCallback',
    securityTest:'SMSRealm-mobile-securityTest',
    poll:{
        interval: 3,
        onPoll: getNotificationsFromBackend
    }
});
```

  - **poll** – method that is used for notification retrieval, with the following required parameters:

    - **Interval**: polling interval in seconds
    - **onPoll**: polling implementation: an adapter function to be invoked at specified intervals

Notification API: Server side

- Send a notification:

```
function sendSMS(userId, smsText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('SMSAdapter.SMSEventSource', userId);

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

    var badgeDigit = 1;

    var notification = WL.Server.createDefaultNotificat

    WL.Logger.debug("sendSMS >> userId :: " + userId +

    WL.Server.notifyAllDevices(userSubscription, notifi

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

Notifications can be either polled from or pushed by the back-end system.

In this example, a **sendSMS()** adapter function is called by a back-end system as an external API to send notifications.

Notification API: Server side

- Send a notification:

    - Obtain notification data

```
function sendSMS(userId, smsText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('SMSAdapter.SMSEventSource', userId);

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

    var badgeDigit = 1;

    var notification = WL.Server.createDefaultNotificati

    WL.Logger.debug("sendSMS >> userId :: " + userId +

    WL.Server.notifyAllDevices(userSubscription, notifi

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

The **sendSMS** function takes the user ID to send the notification to, and **the** SMS text as arguments.

These arguments are provided by the back-end system that calls this function.

Notification API: Server side

- Send a notification:
  - Retrieve the active user and use it to get the user subscription data

```
function sendSMS(userId, smsText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('SMSAdapter.SMSEventSource', userId);

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

    var badgeDigit = 1;

    var notification = WL.Server.cre...rver.cre

    WL.Logger.debug("sendSMS >> user...

    WL.Server.notifyAllDevices(userS...

    return {
        result: "Notification sent t...
    };
}
```

A user subscription object contains the information about all the user subscriptions. Each user subscription can have several device subscriptions.

If the user has no subscriptions for the specified event source, a **null** object is returned.

Notification API: Server side

- Send a notification:

  – Retrieve the user subscription data

```
[
 {
    "alias": "myPushSMS",
    "applicationId": "SMSPushApp-windowsphone-1.0",
    "device": "4811C50B6D106DCA354D52BBFFFFFFFF",
    "eventSourceId": "SMSPushAdapter.SMSPushEventSource",
    "options": {
    },
    "platform": "SMS",
    "token": "3656886544",
    "userAgent": "SMS"
 }
]
```

> By using the **getDeviceSubscriptions** API, you can obtain separate subscription data for each user device.
>
> The result is an array of objects with the structure shown here.

Notification API: Server side

- Sending a notification

  – Send notification to the user devices

```
function sendSMS(userId, smsText){
    var userSubscription =
        WL.Server.getUserNotificationSubscription('SMSAdapter.SMSEventSource', userId);

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

    var badgeDigit = 1;

    var notification = WL.Server.createDefaultNotification(smsText, badgeDigit, {});

    WL.Logger.debug("sendSMS >> userId :: " + userId + ", text :: " + smsText);

    WL.Server.notifyAllDevices(userSubscription, notification);

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

WL.Server.notifyAllDevices API sends notification to all the subscribed user devices.

Notification API: Server side

- To send notifications, the following APIs are used:

  - **WL.Server.notifyAllDevices**(`userSubscription`, `options`): to send notification to all user devices (see previous slide)

  - **WL.Server.notifyDevice**(`userSubscription`, `device`, `options`): to send notification to a specific device for a specific user subscription

  - **WL.Server.notifyDeviceSubscription**(`deviceSubscription`, `options`): to send notification to a specific device

Notification API: Client side

- **Event Source** – subscribing and unsubscribing
  - To subscribe, a user must first be authenticated.
  - Use the following API to subscribe to the event source:

```
function subscribeSMSButtonClicked() {
    WL.Client.Push.subscribeSMS("myPushSMS", "SMSPushAdapter",
            "SMSPushEventSource", "919844634XXX", {
        onSuccess: subscribeSMSOnSuccess,
        onFailure: subscribeSMSOnFailure
    });
}
function subscribeSMSOnFailure(response) {
    alert("subscribeSMS OnFailure");
}
function unsubscribeSMSOnSuccess(response) {
    alert("unsubscribeSMS OnSuccess");
}
```

  - **WL.Client.Push.subscribeSMS()** has the following parameters:
    - An alias to identify the subscription
    - An adapter name where the event source is defined
    - An event source name to which the client is subscribing
    - A user mobile phone number where notifications are sent
      - Mobile phone numbers can contain digits (0-9), plus sign (+), minus sign (-), and space (␣) characters only.
    - Optional **onSuccess** callback
    - Optional **onFailure** callback

  - Callbacks receive a response object if one is required.

Notification API: Client side

- **Event Source** – subscribing and unsubscribing
  - Use the following API to unsubscribe from the event source:

```
function unsubscribeSMSButtonClicked(){
    WL.Client.Push.unsubscribeSMS("myPushSMS", {
        onSuccess: unsubscribeSMSOnSuccess,
        onFailure: unsubscribeSMSOnFailure
    });
}

function unsubscribeSMSOnSuccess(response) {
    alert("unsubscribeSMS OnSuccess");
}

function unsubscribeSMSOnFailure(response) {
    alert("unsubscribeSMS OnFailure");
}
```

  - **WL.Client.Push.unsubscribeSMS** has the following parameters:
    - An alias - the same as that defined in WL.Client.Push.subscribeSMS
    - Optional **onSuccess** callback
    - Optional **onFailure** callback

  - Callbacks receive a response object if one is required.

Notification API: Client side

- You can also unsubscribe by using the IBM Worklight Console to specify the mobile phone numbers to be unsubscribed (comma-separate multiple phone numbers)

- Additional client-side APIs:
  - **WL.Client.Push.isPushSMSSupported()** – returns **true** if SMS notifications are supported by the platform, **false** otherwise.
  - **WL.Client.Push.isSMSSubscribed(alias)** – returns **true** if the currently logged-in user is subscribed to a specified event source alias, **false** otherwise.

Agenda

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

Subscribe SMS servlet

- Subscribe to SMS notifications by using a Subscribe SMS servlet.

- Users can subscribe to SMS notifications without an application on the device by making HTTP GET calls to the servlet.

- The Subscribe SMS servlet supports both subscribe and unsubscribe options.

- Access the Subscribe SMS servlet with the URL:

  – http://<hostname>:<port>/<context>/subscribeSMS?<parameters>

- An application and event source within an adapter must be deployed to the Worklight Server before making the servlet call.

Subscribe SMS servlet : URL Parameters

- The following parameters can be passed to the Subscribe SMS servlet:

URL parameter	Usage
option	Optional. The default value is `subscribe`. For any non-blank string other than `subscribe`, the unsubscribe action is performed.
event Source	Mandatory. `AdapterName.EventSourceName`.
alias	Optional. A short ID defining the event source name.
phone number	Mandatory. The number to which the SMS notifications are sent.
user name	Optional. If no value is provided, the phone number is used as the user name.
appID	Mandatory. The ID of the application that contains SMS gateway definition. For exampe: SMSPushApp-android-1.0.

Subscribe SMS servlet: Subscribe URL and response

- Subscribe URL:

  http://<host>:<port>/<context>/subscribeSMS?alias=<aliasName>&appId=<appId>&event
  Source=<evenSourceName>&username=<username>&phoneNumber=<number>

- Sample response from the Worklight Server:

·{"result":"FWLSE00146I: User 'vk' with phone number '0987654321' successfully subscribed to SMS notifications","isSuccessful":"true"}

Subscribe SMS servlet: Unsubscribe URL and response

- Unsubscribe URL:

  http://<host>:<port>/<context>/subscribeSMS?option=unsubscribe&eventSource=SMSAdapter.SMSEventSource&userName=<username>&phoneNumber=<phonenumber>

- Sample response from the Worklight Server:

```
{"result":"FWLSE00147I: Successfully unsubscribed from SMS notifications for user 'vk'","isSuccessful":"true"}
```

Subscribe SMS servlet: Features

- Subscriptions that are made by using the Subscribe SMS servlet are independent of subscriptions that are created from a device.

- Unsubscribing by using SMS servlet does not remove subscriptions that are made from device.

- Security

  - By default, the Subscribe SMS servlet is protected as a static resource.

  - By default, the authenticationConfig.xml file is configured to reject all requests to the Subscribe SMS servlet by using a rejecting login module.

  - To allow access to the Subscribe SMS servlet, an administrator must modify the authenticationConfig.xml file with appropriate authenticator and login modules.

Subscribe SMS servlet: Security

- By default, authenticationConfig.xml is configured to reject all requests to the Subscribe SMS servlet:

```xml
<resource id="subscribeServlet" securityTest="SubscribeServlet">
    <urlPatterns>/subscribeSMS*</urlPatterns>
</resource>

<customSecurityTest name="SubscribeServlet">
    <test realm="SubscribeServlet" isInternalUserID="true" />
</customSecurityTest>

<realm name="SubscribeServlet" loginModule="rejectAll">
    <className>com.worklight.core.auth.ext.HeaderAuthenticator</className>
</realm>

<loginModule name="rejectAll">
    <className>com.worklight.core.auth.ext.RejectingLoginModule</className>
</loginModule>
```

- When the servlet is invoked with default authentication configuration:

```
/*-secure-
{"WL-Authentication-Failure":{"SubscribeServlet":{"reason":"illegal state"}}}*/
```

Subscribe SMS servlet: Security

- An administrator can change authenticationConfig.xml to allow access to the Subscribe SMS servlet:

```xml
<resource id="subscribeServlet" securityTest="SubscribeServlet">
    <urlPatterns>/subscribeSMS*</urlPatterns>
</resource>

<customSecurityTest name="SubscribeServlet">
    <test realm="SMSRealm" isInternalUserID="true" />
</customSecurityTest>

<realm name="SMSRealm" loginModule="smsHeader">
    <className>com.worklight.core.auth.ext.HeaderAuthenticator</className>

<loginModule name="smsHeader">
    <className>com.worklight.core.auth.ext.HeaderLoginModule</className>
    <parameter name="user-name-header" value="vk"></parameter>
</loginModule>
```

- When the servlet is invoked with custom authentication configuration:

{"result":"FWLSE00146I: User 'vk' with phone number '0987654321' successfully subscribed to SMS notifications","isSuccessful":"true"}

Agenda

- What are SMS notifications?

- Device support

- Notification architecture

- Subscription management

- SMS notification API

- Subscribe SMS servlet

- Setup

Setup

- **SMSConfig.xml**
  - To send SMS notifications, define in the SMSConfig.xml file the third-party gateway services that are supported by the Worklight Server.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sms:config xmlns:sms="http://www.worklight.com/sms/config"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <gateway id="myGateway" hostname="yourhostname.com" port="80" programName="backendProgram"
      toParamName="to" textParamName="text">
    <!-- <parameter name="" value=""/> -->
  </gateway>
</sms:config>
```

- **application-descriptor.xml**
  - For an application to use SMS notifications, specify the SMS gateway to use by adding an `<smsGateway id="…"/>` element to the application-descriptor.xml file.
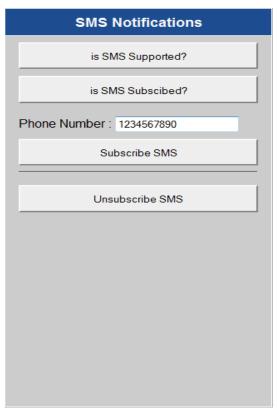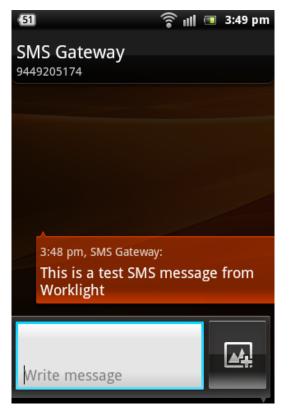
```xml
<smsGateway id="myGateway"/>
```

Setup

- The gateway ID in the SMSConfig.xml file must correspond to the SMS gateway ID in the application-descriptor.xml file.

```xml
<gateway id="myGateway" hostname="yourhostname.com" port="80" programName="backendProgram"
        toParamName="to" textParamName="text">
    <parameter name="param1name" value="param1value"/>
    <parameter name="param2name" value="param2value"/>
</gateway>
```

```xml
        ...h>320</width>
    <mainFile>SMSReceiverApp.html</mainFile>
    <thumbnailImage>common/images/thumbnail.
    <smsGateway id="myGateway"/>
    <android version="1.0" securityTest="SMS
        <worklightSettings include="true"/>
        <security>
            <encryptWebResources enabled="fa
            <testWebResourcesChecksum enable
```

The result

- The sample for this training module can be found in the Getting Started page of the IBM Worklight Foundation documentation at http://www.ibm.com/mobile-docs.

- Notice that you need a SMS Gateway subscription to activate push notifications.

Back-end emulator

- The sample for this training module comes with a back-end system emulator that you can use to simulate SMS notification submission by a back-end system.

- To run the back-end system emulator:

  1. Open a command prompt.

  2. Go to the `SMSBackendEmulator` folder of the sample project.

  3. Enter the following command to run the supplied **SMSBackendEmulator.jar** file:

java –jar SMSBackendEmulator.jar <userId> <notificationText> <serverPort>

- where **<userId>** is the user name that you used to log in to the sample application

Example:

java –jar SMSBackendEmulator.jar user1 "hello from sms" 8080

Back-end emulator

- The back-end system emulator tries to connect to a Worklight Server and call the **sendSMS()** adapter procedure. It then produces the invocation result to a command prompt console.

- Success

```
C:\worklight\workspace\test\module_42_0_SMSNotifications\SMSBackendEmulator>java
 -jar SMSBackendEmulator.jar user1 "hello from sms" 8080
SMSBackendEmulator
User Id: user1
Notification text: hello from sms
Server URL: http://localhost:8080
sending notification
Server response :: {   "isSuccessful": true,    "result": "Notification sent to u
ser :: user1"}
```

- Failure

```
C:\worklight\workspace\test\module_42_0_SMSNotifications\SMSBackendEmulator>java
 -jar SMSBackendEmulator.jar user1 "hello from sms" 8080
SMSBackendEmulator
User Id: user1
Notification text: hello from sms
Server URL: http://localhost:8080
sending notification
Server response :: {   "isSuccessful": true,    "result": "No subscription found
for user :: user1"}
```

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