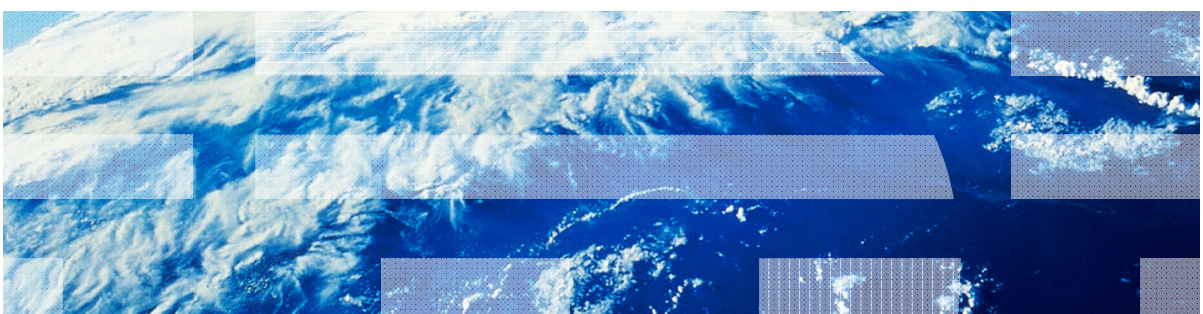
IBM Worklight V5.0.5
Getting Started

Module 25 – Custom Device Provisioning



18 January 2013

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Worklight is a trademark or registered trademark of Worklight, an IBM Company. 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

- About
- Provisioning introduction
- Custom Provisioning introduction
- Creating a Custom Provisioning
- Examining the result

About

- This module explains how to create custom provisioning:
  - You learn how to implement a custom provisioning that uses a certificate from an external service to authenticate a device
  - You learn how to implement a custom authenticator that connects to that service
- Before you follow this module, make sure that you well understand the IBM Worklight® authentication concepts
  - Make sure that you have a solid understanding of the authentication modules 20 and 23
  - For more information about Worklight authentication concepts, see the IBM Worklight Information Center

Agenda

- About
- Provisioning introduction
- Custom Provisioning introduction
- Creating a Custom Provisioning
- Examining the result

Provisioning introduction

- Provisioning: A mechanism where a digital signature is created to protect the integrity and authenticity of a device or of an application

Agenda

- About
- Provisioning introduction
- Custom Provisioning introduction
- Creating a Custom Provisioning
- Examining the result

Custom Provisioning introduction

- There are three types of provisioning processes in IBM Worklight:
  - No provisioning: the client application does not trigger the provisioning process, and the server does not verify the client certificate
  - Auto-provisioning: the Worklight Server automatically issues a certificate for the device and application data, provided by the client application
  - Custom provisioning: the Worklight Server is augmented with custom logic that controls the device and the application provisioning process.
    - This logic can involve integration with an external system that can issue the client certificate, based on data that is obtained from the app, or can instruct the Worklight Server to do so

Custom Provisioning introduction

- Whether obtained by auto-provisioning or custom provisioning process, the client app stores the certificate on the device
- The certificate is then used for signing the payload that is sent to the Worklight Server
- The Worklight server validates the client certificate, regardless of how it was obtained

Agenda

- About
- Provisioning introduction
- Custom Provisioning introduction
- Creating a Custom Provisioning
- Examining the result

Creating a Custom Provisioning

- authenticationConfig.xml

```
<securityTests>
    <customSecurityTest name="customTests">
        <test realm="wl_authenticityRealm"/>
        <test realm="wl_remoteDisableRealm"/>
        <test isInternalUserID="true" realm="wl_anonymousUserRealm"/>
        <test isInternalDeviceID="true" realm="MyCustomProvisioning"/>
    </customSecurityTest>
</securityTests>
<realms>
    <realm loginModule="MyCustomProvisioningLoginModule" name="MyCustomR
        <className>com.prov.MyProvisioningAuthenticator</className>
        <parameter name="provisioned-entity" value="group:myapps"/>
        <parameter name="pre-required-realms" value="wl_authenticityReal
    </realm>

    <realm loginModule="StrongDummy" name="SampleAppRealm">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</c
    </realm>

    <realm loginModule="requireLogin" name="WorklightConsole">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
        <onLoginUrl>/console</onLoginUrl>
    </realm>
</realms>
<loginModules>
    <loginModule name="MyCustomProvisioningLoginModule">
        <className>com.prov.MyProvisioningLoginModule</className>
    </loginModule>
    <loginModule name="StrongDummy">
        <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className>
    </loginModule>
    <loginModule name="requireLogin">
        <className>com.worklight.core.auth.ext.SingleIdentityLoginModule</className>
    </loginModule>
</loginModules>
```

A test with isInternalDeviceID="true" must exist for the realm that you are going to use for custom provisioning (MyCustomProvisioning)

Creating a Custom Provisioning

- authenticationConfig.xml

```
<securityTests>
    <customSecurityTest name="customTests">
        <test realm="wl_authenticityRealm"/>
        <test realm="wl_remoteDisableRealm"/>
        <test isInternalUserID="true" realm="wl_anonymousUserRealm"/>
        <test isInternalDeviceID="true" realm="MyCustomProvisioning"/>
    </customSecurityTest>
</securityTests>
<realms>
    <realm loginModule="MyCustomProvisioningLoginModule" name="MyCustomP
        <className>com.prov.MyProvisioningAuthenticator</className>
        <parameter name="provisioned-entity" value="group:myapps"/>
        <parameter name="pre-required-realms" value="wl_authenticityRealm,W
    </realm>

    <realm loginModule="StrongDummy" name="SampleAppRealm">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
    </realm>

    <realm loginModule="requireLogin" name="WorklightConsole">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
        <onLoginUrl>/console</onLoginUrl>
    </realm>
</realms>
<loginModules>
    <loginModule name="MyCustomProvisioningLoginModule">
        <className>com.prov.MyProvisioningLoginModule</className>
    </loginModule>
    <loginModule name="StrongDummy">
        <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className>
    </loginModule>
    <loginModule name="requireLogin">
        <className>com.worklight.core.auth.ext.SingleIdentityLoginModule</className>
    </loginModule>
</loginModules>
```

The parameter provisioned-entity group:myapps states that all apps share same provision.
For Android, you need to have a shared user ID in the application-descriptor.xml file.

Creating a Custom Provisioning

- authenticationConfig.xml

```
<securityTests>
    <customSecurityTest name="customTests">
        <test realm="wl_authenticityRealm"/>
        <test realm="wl_remoteDisableRealm"/>
        <test isInternalUserID="true" realm="wl_anonymousUserRealm"/>
        <test isInternalDeviceID="true" realm="MyCustomProvisioning"/>
    </customSecurityTest>
</securityTests>
<realms>
    <realm loginModule="MyCustomProvisioningLoginModule" name="MyCustomPr
        <className>com.prov.MyProvisioningAuthenticator</className>
        <parameter name="provisioned-entity" value="group:myapps"/>
        <parameter name="pre-required-realms" value="wl_authenticityRealm,wl_remoteDisableRealm"/>
    </realm>

    <realm loginModule="StrongDummy" name="SampleAppRealm">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
    </realm>

    <realm loginModule="requireLogin" name="WorklightConsole">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
        <onLoginUrl>/console</onLoginUrl>
    </realm>
</realms>
<loginModules>
    <loginModule name="MyCustomProvisioningLoginModule">
        <className>com.prov.MyProvisioningLoginModule</className>
    </loginModule>
    <loginModule name="StrongDummy">
        <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className>
    </loginModule>
    <loginModule name="requireLogin">
        <className>com.worklight.core.auth.ext.SingleIdentityLoginModule</className>
    </loginModule>
</loginModules>
```

The parameter pre-required-realms has a list of comma delimited realms that are pre-existing in the authenticationConfig.xml file.

Creating a Custom Provisioning

- MyProvisioningLoginModule.java

```
package com.prov;

import com.worklight.core.auth.ext.DeviceAutoProvisioningLoginModule;

public class MyProvisioningLoginModule extends DeviceAutoProvisioningLoginModule {

}
```

Extending the DeviceAutoProvisioningLoginModule class with MyProvisioningLoginModule

Creating a Custom Provisioning

- MyProvisioningAuthenticator.java

Get the options that are declared in the authenticationConfig.xml file

```
package com.prov;

import java.io.BufferedReader;

public class MyProvisioningAuthenticator extends DeviceAutoProvisioningAuthenticator {

    private static URL MY_URL;

    @Override
    public void init(Map<String, String> option) throws MissingConfigurationOptionException {
        entityString = option.remove(PROVISIONED_ENTITY_PARAM_NAME);
        preRequiredRealms = option.remove(PRE_REQUIRED_REALMS_PARAM_NAME);
        super.init(option);

        try {
            MY_URL = new URL("http://localhost:8089/");
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected AuthenticationResult checkChallangeResponse(Object challengeResponse, HttpServletResponse response) throws IOException {
        if(challengeResponse instanceof JSONObject) {
            JSONObject challengeJSON = (JSONObject) challengeResponse;

            if (challengeJSON.containsKey(CSR_PARAM_NAME)) {
                if (isProvisioningAllowed()) {
                    return handleCSR((String) challengeJSON.get(CSR_PARAM_NAME), response);
                } else {
                    return AuthenticationResult.createFailureResult(new JSONObject(), "Provisioning is not allowed at this time");
                }

            }
        }
        return super.checkChallangeResponse(challengeResponse, response);
    }

    private AuthenticationResult handleCSR(String csr, HttpServletResponse response) {
        AuthenticationResult result;
        try {
            URLConnection connection = MY_URL.openConnection();
            connection.setDoOutput(true);
            connection.setDoInput(true);
```

Creating a Custom Provisioning

- MyProvisioningAuthenticator.java

```
package com.prov;

import java.io.BufferedReader;

public class MyProvisioningAuthenticator extends DeviceAutoProvisioningAuthenticator {

    private static URL MY_URL;

    @Override
    public void init(Map<String, String> option) throws MissingConfigurationOptionExcep
        entityString = option.remove(PROVISIONED_ENTITY_PARAM_NAME);
        preRequiredRealms = option.remove(PRE_REQUIRED_REALMS_PARAM_NAME);
        super.init(option);

        try {
            MY_URL = new URL("http://localhost:8089/");
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected AuthenticationResult checkChallangeResponse(Object challengeResponse, HttpServletResponse response) throws IOException {
        if(challengeResponse instanceof JSONObject) {
            JSONObject challengeJSON = (JSONObject) challengeResponse;

            if (challengeJSON.containsKey(CSR_PARAM_NAME)) {
                if (isProvisioningAllowed()) {
                    return handleCSR((String) challengeJSON.get(CSR_PARAM_NAME), response);
                } else {
                    return AuthenticationResult.createFailureResult(new JSONObject(), "Provisioning is not allowed at this time");
                }

            }
        }
        return super.checkChallangeResponse(challengeResponse, response);
    }

    private AuthenticationResult handleCSR(String csr, HttpServletResponse response) {
        AuthenticationResult result;
        try {
            URLConnection connection = MY_URL.openConnection();
            connection.setDoOutput(true);
            connection.setDoInput(true);
```

This URL is the provisioning service URL.
In this case, the provisioning service is running on the localhost on port 8089.

Creating a Custom Provisioning

- MyProvisioningAuthenticator.java

```
package com.prov;

import java.io.BufferedReader;

public class MyProvisioningAuthenticator extends DeviceAutoProvisioningAuthenticator {

    private static URL MY_URL;

    @Override
    public void init(Map<String, String> option) throws MissingConfigurationOptionExcep
        entityString = option.remove(PROVISIONED_ENTITY_PARAM_NAME);
        preRequiredRealms = option.remove(PRE_REQUIRED_REALMS_PARAM_NAME);
        super.init(option);

        try {
            MY_URL = new URL("http://localhost:8089/");
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected AuthenticationResult checkChallangeResponse(Object challengeResponse, HttpServletResponse response) throws IOException {
        if(challengeResponse instanceof JSONObject) {
            JSONObject challengeJSON = (JSONObject) challengeResponse;

            if (challengeJSON.containsKey(CSR_PARAM_NAME)) {
                if (isProvisioningAllowed()) {
                    return handleCSR((String) challengeJSON.get(CSR_PARAM_NAME), response);
                } else {
                    return AuthenticationResult.createFailureResult(new JSONObject(), "Provisioning is not allowed at this time");
                }

            }
        }
        return super.checkChallangeResponse(challengeResponse, response);
    }

    private AuthenticationResult handleCSR(String csr, HttpServletResponse response) {
        AuthenticationResult result;
        try {
            URLConnection connection = MY_URL.openConnection();
            connection.setDoOutput(true);
            connection.setDoInput(true);
```

Overriding the AuthenticationResult checkChallengeResponse

Creating a Custom Provisioning

- MyProvisioningAuthenticator.java

```
package com.prov;

import java.io.BufferedReader;

public class MyProvisioningAuthenticator extends DeviceAutoProvisioningAuthenticator {

    private static URL MY_URL;

    @Override
    public void init(Map<String, String> option) throws MissingConfigurationOptionExcep
        entityString = option.remove(PROVISIONED_ENTITY_PARAM_NAME);
        preRequiredRealms = option.remove(PRE_REQUIRED_REALMS_PARAM_NAME);
        super.init(option);

        try {
            MY_URL = new URL("http://localhost:8089/");
        } catch (MalformedURLException e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected AuthenticationResult checkChallangeResponse(Object challengeResponse, HttpServletResponse response) throws IOException {
        if(challengeResponse instanceof JSONObject) {
            JSONObject challengeJSON = (JSONObject) challengeResponse;

            if (challengeJSON.containsKey(CSR_PARAM_NAME)) {
                if (isProvisioningAllowed()) {
                    return handleCSR((String) challengeJSON.get(CSR_PARAM_NAME), response);
                } else {
                    return AuthenticationResult.createFailureResult(new JSONObject(), "Provisioning is not allowed at this time");
                }

            }
        }
        return super.checkChallangeResponse(challengeResponse, response);
    }

    private AuthenticationResult handleCSR(String csr, HttpServletResponse response) {
        AuthenticationResult result;
        try {
            URLConnection connection = MY_URL.openConnection();
            connection.setDoOutput(true);
            connection.setDoInput(true);
```

Checking whether there is a response for custom provisioning. If so, and if provisioning is allowed then call “handleCSR”.

Creating a Custom Provisioning

- MyProvisioningAuthenticator.java

```
            if (challengeJSON.containsKey(CSR_PARAM_NAME)) {
                if (isProvisioningAllowed()) {
                    return handleCSR((String) challengeJSON.get(CSR_PARAM_NAME), response);
                } else {
                    return AuthenticationResult.createFailureResult(new JSONObject(), "Provisioning is not allowed at this time");
                }

            }
        }
        return super.checkChallangeResponse(challengeResponse, response);
    }

    private AuthenticationResult handleCSR(String csr, HttpServletResponse response) {
        AuthenticationResult result;
        try {
            URLConnection connection = MY_URL.openConnection();
            connection.setDoOutput(true);
            connection.setDoInput(true);
            connection.connect();

            OutputStreamWriter osw = new OutputStreamWriter(connection.getOutputStream());

            osw.write(csr);
            osw.close();

            BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));

            StringBuffer sb = new StringBuffer();
            String s;
            while((s = br.readLine()) != null) {
                sb.append(s);
            }

            br.close();
            result = createNewChallenge();
            result.getJson().put(CERTIFICATE, sb.toString());
        }
        catch (IOException e) {
            result = AuthenticationResult.createFailureResult(new JSONObject(), e.getMessage());
        }
        return result;
    }
}
```

The “handleCSR” method connects to the provisioning service, and pushes the CSR.

Creating a Custom Provisioning

- myProvisioningChallengeHandler.js

```
var myCustomDeviceProvisioningChallengeHandler = WL.Client
      .createProvisioningChallengeHandler("MyCustomProvisioning");

myCustomDeviceProvisioningChallengeHandler.createJsonCsr = function(provisionEntity, realm, customPayload){
    var csrPayload = {};
    if (!WLJSX.Object.isUndefined(customPayload)){
        csrPayload = customPayload;
    }

    csrPayload.deviceId = device.uuid;

    if(provisionEntity == 'application') {
        csrPayload.applicationId = WL.StaticAppProps.APP_DISPLAY_NAME;
    } else if (provisionEntity.indexOf("group:") == 0){
        csrPayload.groupId = provisionEntity.substr(6);
    }

    myCustomDeviceProvisioningChallengeHandler.onCsrDataReady(csrPayload, provisionEntity);
};
```

Creating a provisioning challenge handler.
Creating a CSR payload with the device UUID.

Agenda

- About
- Provisioning introduction
- Custom Provisioning introduction
- Creating a Custom Provisioning
- Examining the result

Examining the result

- The sample for this training module can be found in the Getting Started page of the IBM Worklight documentation website at http://www.ibm.com/mobile-docs.
- In the training module, the custom-prov.jar file exists in the provisioningService folder.
- Go into that folder, and use Java™ to run the custom-prov.jar through a command line, as follows:
  `java –jar custom-prov.jar`
  - The provisioning service will run on your localhost on port 8089.
- Build the customProvApp application a deploy it on an Android device.

Examining the result

- On the provisioning service console, you can see:

```
C:\Users\ravidor\Desktop>java -jar cutom-prov.jar
2012-12-02 14:54:47.660:INFO::Logging to StdErrLog::DEBUG=false via org.eclipse.
jetty.util.log.StdErrLog
2012-12-02 14:54:47.671:INFO::jetty-7.0.2-SNAPSHOT
2012-12-02 14:54:47.707:INFO::Started SelectChannelConnector@0.0.0.0:8089
csr: eyJqcGsiOnsiYWxnIjoiUlNBIiwiZXhwIjoiQVFBQiIsIm1vZCI6IkFQSOQ1Rmt1TjBkWGFQTm
zTUdKOVlVTHlPSmVmVlNNQl9OaHFFa25hYVE3R3U2bO85OVNMekIxajd5VFhtUm94TDFwcnRibjI2MW
yTW85ZlJncGZmazOifSwiYWxnIjoiUlMyNTYifQ==.eyJncm91cElkIjoibXlhcHBzIiwidG9rZW4iO
I2MmdqbHZzMG9pODJrazA5c2htYzdpdTMiLCJkZXZpY2VJZCI6Ijk3NzRkNTZkNjgyZTU0OWMifQ==.
5kqhBQ1K3oEPVkQIqEPT7KRWFyMuxBsdS8Vi3okcFvVAKZyHP6yqpOSKYQxc761KHiXoVo4JrRohXXK
eeuig==
Certificate sent:   [0]         Version: 3
         SerialNumber: 17512043915600993099
             IssuerDN: C=IL,ST=IL,L=Shefayim,O=IBM,OU=Worklight,CN=WL Dev
           Start Date: Sun Dec 02 14:57:33 IST 2012
           Final Date: Sat Dec 02 14:57:33 IST 2062
            SubjectDN: DC=myapps,UID=9774d56d682e549c
           Public Key: RSA Public Key
            modulus: f283e4592e37475768f365dcc189f5850bc8e25e7d548c07f361a84927
9a43b1aeea83bdf522f30758fbc935e6468c4bd69aed6e7dbad60d8ca3d7d18297df9
    public exponent: 10001

  Signature Algorithm: SHA256WithRSAEncryption
            Signature: 34597cef584dcbc42bd54d0b4b8fed6b5929004a
                       36f62e65ccc39f6b009cc4f94409d9201bf304aa
                       8d30149f155a824b6a157e084d903f5d1ad34f96
                       322e825042cee8362f63724a68d7483a962d3bc5
                       8fe29ea54ff66c5962524787787b130e8dc68326
                       747336cff511b99055e85a277fbc8625fe4239cf
                       20b7567175f1b7d3199e0c99802a9105db92a937
                       93646a641bf6eaa22a4aa4a721f68ecf82b001a0
                       ff2070dea372810cc550b1557fe921e8422ee240
                       8cee0029a519bb88811f85d93f7592b1662fc4c6
                       287a9fd573053d32a04a6f45751bd00ebe83c7dc
                       3a13ae04aa5ece21de1fde1f98686256e52f2354
                       ab25ab3cee1d265650c761e4ddb9470b
```

CSR that is received from the device

The certificate that is sent back

Examining the result

- On the LogCat you can see:

Text

```
/android/init] success: /*-secure-
{"userPrefs":{},"gadgetProps":{"directUpdate":{"availableSkins":["d
efault"],"checksum":664823034,"updateSize":279769},"ENVIRONMENT":"a
ndroid"},"userInfo":{"wl_authenticityRealm":{"userId":"wl_authentic
ityLoginModule","attributes":{},"isUserAuthenticated":1,"displayNam
e":"wl_authenticityLoginModule"},"MyCustomProvisioning":{"userId":"
device","attributes":{"mobileClientData":"com.worklight.core.auth.i
mpl.MobileClientData@2ffc32e3"},"isUserAuthenticated":1,"displayNam
e":"device"},"SampleAppRealm":{"userId":null,"attributes":{},"isUse
rAuthenticated":0,"displayName":null},"wl_remoteDisableRealm":{"use
rId":"NullLoginModule","attributes":{},"isUserAuthenticated":1,"dis
playName":"NullLoginModule"},"wl_antiXSRFRealm":{"userId":null,"att
ributes":{},"isUserAuthenticated":0,"displayName":null},"WorklightC
onsole":{"userId":null,"attributes":{},"isUserAuthenticated":0,"dis
playName":null},"wl_deviceAutoProvisioningRealm":{"userId":null,"at
tributes":{},"isUserAuthenticated":0,"displayName":null},"wl_device
NoProvisioningRealm":{"userId":null,"attributes":{},"isUserAuthenti
cated":0,"displayName":null},"myserver":{"userId":"3bd3095d-ae36-4e
8e-9053-281ec320455b","attributes":{},"isUserAuthenticated":1,"disp
layName":"3bd3095d-ae36-4e8e-9053-281ec320455b"},"wl_anonymousUserR
ealm":{"userId":"3bd3095d-ae36-4e8e-9053-281ec320455b","attributes"
:{},"isUserAuthenticated":1,"displayName":"3bd3095d-ae36-4e8e-9053-
281ec320455b"}}}*/
```

myCustomProvisioning is unauthenticated.

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.

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 forums at:
    - https://www.ibm.com/developerworks/mobile/mobileforum.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