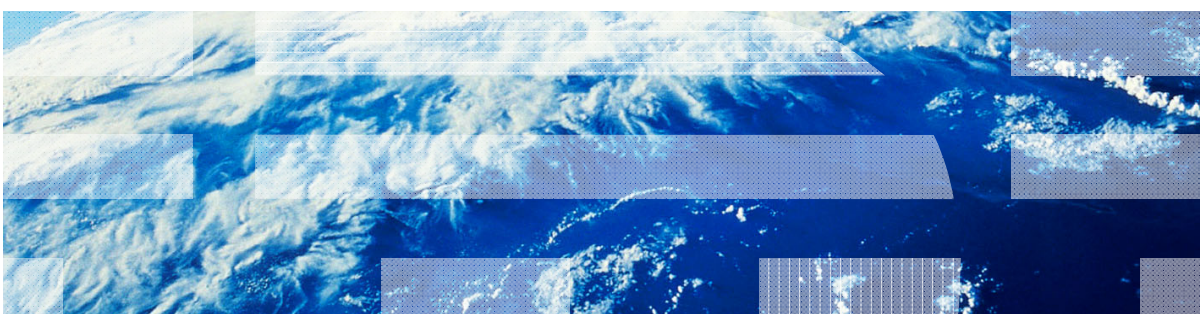
IBM Worklight V5.0.5
Getting Started

Module 30 – Team Development using RTC



18 January 2013

Trademarks

- IBM, the IBM logo, ibm.com, Jazz, Rational, and Rational Team Concert 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

- Building Worklight® projects with the Rational Team Concert™ build system
  - Building and deploying hybrid resources
  - Building the native binary for iOS
  - Building the native binary for Android

Building Worklight projects with the Rational Team Concert build system (1 of 2)

- You can deliver the code contained within your Worklight® projects to Rational Team Concert™.
- You can use the Rational Team Concert (RTC) build system to build Worklight projects by using either of the following methods:
  - Request personal builds based on the private contents of your personal workspaces.
  - Schedule regular builds based on contents delivered in a stream.
- The Consumer and Enterprise editions of IBM® Worklight contain Ant tasks. These Ant tasks build the web code within your Worklight projects and deploy the build results to the Worklight Server.
- Mobile operating system native SDKs contain Ant-based command-line interfaces, such as the Android SDK and Xcode. You can use these command-line interfaces to build the native executables (such as .apk files for Android and .ipa files for iOS) to run on the device.
- *Tip:* If you are using the Application Center that is available in the Enterprise Edition of IBM Worklight, you can use the provided Ant tasks to deploy the native executables to the application.

Building Worklight projects with the Rational Team Concert build system (2 of 2)

- The following build and deploy tasks are supported in the Rational Team Concert build system:
  - **Building and deploying hybrid resources**: You can create a build definition to build hybrid web code that is contained within Worklight projects.

    The build definition uses the Ant tasks from the `worklight-ant.jar` file to build and deploy your hybrid code.
  - **Building the native binary for iOS**: You can create a build definition to build iPhone and iPad native executables, by using the command-line interfaces provided with Xcode.
  - **Deploying the native binary for iOS to the Application Center**: For details on deploying the native binary for iOS to the Application Center, see *Command-line tool for uploading an application* in the IBM Worklight Information Center.
  - **Building the native binary for Android**: You can create a build definition to build Android native executables, by using the command-line interfaces provided with the Android SDK.
  - **Deploying the native binary for Android to the Application Center**: For details on deploying the native binary for Android to the Application Center, see *Command-line tool for uploading an application* in the IBM Worklight Information Center.

Agenda

- Building Worklight projects with the Rational Team Concert build system
  - Building and deploying hybrid resources
  - Building the native binary for iOS
  - Building the native binary for Android

Building and deploying hybrid resources (1 of 8)

- You can create a build definition to build hybrid web code that is contained within Worklight projects.
- The build definition uses the Ant tasks from the `worklight-ant.jar` file to build and deploy your hybrid code.
- To prepare the environment on the workstation that runs the Jazz™ build engine for your hybrid builds, perform the steps given in the following slides.

Building and deploying hybrid resources (2 of 8)

1. Create a build folder, such as:

   `~/Users/username/Documents/wl.build`

2. Within the build folder, complete the following steps:
   a. Create a folder named `load`: Use this folder to store the source code that Rational Team Concert™ checks out.
   b. Create a folder named `output`: Use this folder to store the build output.
   c. Create a file named `hybrid.build.xml` with the content of the next slide.

Building and deploying hybrid resources (3 of 8)

The text in italic depends on your project:

1. *RTCcomponent*
2. *projectname*
3. *appname*
4. *adaptername*

- Content of the file `hybrid.build.xml`:

```
<project name="build" basedir="../" >
   <description> build worklight project </description>
   <property name="root.dir" value="/Users/username/Documents/wl.build"/>
   <property name="project.path" value="${root.dir}/load/RTCcomponent/projectname"/>
   <property name="output.path" value="${root.dir}/output"/>
   <taskdef resource="com/worklight/ant/defaults.properties">
   </taskdef>
   <target name="build_wl_apps">
      <echo> build_wl_apps ${project.path}</echo>
      <delete dir="${output.path}/projectname"/>
      <record name="${output.path}/build.log" loglevel="verbose" append="false"/>
      <app-builder nativeProjectPrefix="projectname" applicationFolder="${project.path}/apps/appname"
         outputFolder="${output.path}/projectname"> </app-builder>
   </target>
   <target name="build_wl_adapter">
      <echo> build_wl_adapter ${project.path}</echo>
      <record name="${output.path}/build.log" loglevel="verbose" append="true"/>
      <adapter-builder folder="${project.path}/adapters/adaptername"
         destinationFolder="${output.path}/projectname">
      </adapter-builder>
   </target>
   <target name="deploy_test_wl_apps">
      <echo> deploy_test_wl_apps ${output.path}
      </echo>
      <adapter-deployer deployable="${output.path}/projectname/adaptername.adapter"
         worklightServerHost="http://..."/>
      <app-deployer deployable="${output.path}/projectname/appname.wlapp"
         worklightServerHost="http://..."/>
   </target>
</project>
```

1 2 3 4

Building and deploying hybrid resources (4 of 8)

d. Create an empty log file, named `build.log`: Log messages from the Ant tasks are appended to this file.

e. Add the `worklight-ant.jar` file: This file contains the following Ant tasks that are used in the `hybrid.build.xml` file:

- `app-builder`
- `adapter-builder`
- `app-deployer`
- `adapter-deployer`

- The resulting directory structure looks like the following example:

```
wl.build
    load
    output
    hybrid.build.xml
    build.log
    worklight-ant.jar
```

Building and deploying hybrid resources (5 of 8)

3. Update the following parts of the `hybrid.build.xml` file to correspond with your Worklight project:

Option	Description
RTCcomponent (1)	The Rational Team Concert source code component. If you do not create a folder for this component in the build definition, you can remove this segment from the build path.
projectname (2)	Specify the name of the project.
appname (3)	Specify the name of the application.
adaptername (4)	Specify the name of the adapter.

- If you use the Dojo Toolkit in the application, add the following attribute to the *app-builder* element:
  - `skinBuildExtensions="build-dojo.xml"`
  - This attribute is required to run the Ant tasks necessary to copy the required Dojo resources to the application output folder.
- To share the build script among multiple build definitions, use variables instead of hard-coded values for these settings.
  - For example, use a variable, *${projectname}*, and pass in the value by defining the property in the Rational Team Concert build definition.

Building and deploying hybrid resources (6 of 8)

4. Create the build definition in Rational Team Concert:

   Tip: You must use the Eclipse Rational Team Concert client to complete this step.

   a. Open the Rational Team Concert Eclipse client and connect to the project area.
   b. In the Team Artifacts view, ensure that there is a connection to the target Rational Team Concert server.
   c. Connect to the project area that contains the Worklight projects.
   d. Within this project area, right-click **Builds** and click **New Build Definition**.
   e. Ensure that the target project area is selected and click **Next**.
   f. Select the Ant-Jazz Build Engine template and click **Next**.
   g. Select the **Jazz Source Control** check box for Pre-Build.
   h. Ensure that all check boxes for Post-Build actions are cleared.
   i. Click **Finish**.
   – The editor for the new build definition opens.

Building and deploying hybrid resources (7 of 8)

5. Update the following values in the new build definition:

Option	Description
Supporting Build Engines	Specify the id of the build engine that is set up to run the hybrid build.
Build File	Specify the path to the `hybrid.build.xml` file.
Build targets	Specify **build_wl_apps**.
Ant argument	Specify **–lib** <path to the `worklight-ant.jar` file> Specify **-verbose**.

Building and deploying hybrid resources (8 of 8)

6. Save the updated build definition.
7. Test the build:
   a. Ensure that the build engines on the workstations configured for hybrid builds are started, running, and waiting to receive build requests.
      - For more information about setting up Jazz build engines, see the Rational Team Concert Information Center.
   b. In the Eclipse Rational Team Concert client, right-click the new build definition and click **Request Build**.
      - You can monitor the status of the build in the Builds view.

Agenda

- Building Worklight projects with the Rational Team Concert build system
  - Building and deploying hybrid resources
  - Building the native binary for iOS
  - Building the native binary for Android

Building the native binary for iOS (1 of 10)

- Before you build the native artifacts, perform a hybrid build (as described in the previous slides). Doing so ensures that the artifacts required by the iOS native build are properly generated and updated.
- You can create a build definition to build iPhone and iPad native executables, by using the command-line interfaces provided with X Code.
- Before you begin, ensure that the appropriate version of Xcode is installed on your Mac workstation.
- To prepare the environment on the Mac workstation that runs the Jazz™ build engine for your iOS builds, perform the steps given in the following slides.

Building the native binary for iOS (2 of 10)

1. Use one of the following methods to set up the Mac workstation with developer certificates and provisioning profiles:
   A. Provisioning portal
   B. Manual setup

Building the native binary for iOS (3 of 10)

A. **Provisioning portal**: Use this method if you can dedicate the Mac to a particular developer account.
   a. Log on to the iOS Developer Program Member Center.
   b. In the iOS Provisioning Portal, click **Launch Assistant** within the Development Provisioning Assistant section.
   c. Follow the Development Provisioning Assistant to set up the workstation.

Building the native binary for iOS (4 of 10)

B. **Manual setup**: Use this method if you cannot configure the Mac as the primary development workstation for a development account.

   a. Open Xcode and select **Window > Organizer > Library > Provisioning Profiles**.

   b. Beside the **Automatic Device Provisioning** option, click **Refresh**.

   c. When prompted to log on to the Provisioning Portal, use the iOS Enterprise Developer Program account to log on.

   d. Verify that the provisioning profiles are successfully imported.

   e. Various warnings are displayed for the provisioning profiles to indicate that the profiles cannot be validated with matching private keys. Complete the following steps to resolve this issue:

      i. On the Mac workstation that is set up using the Provisioning Portal, open **Keychain Access**.

      ii. Locate the certificate that is installed from the iOS program, such as login or System.

      iii. Expand the certification to view the private key that is associated with the certificate.

      iv. Right-click the private key and select **Export to a .p12 file**.

      v. Transfer the file to the build workstation and import it into the Keychain.

      vi. Verify that the provisioning profiles are now recognized.

Building the native binary for iOS (5 of 10)

2. Create a build folder, such as `~/Users/`*`username`*`/Documents/wl.build`
3. Within the build folder, complete the following steps:
   a. Create a folder named `load`: Use this folder to store the source code that Rational Team Concert™ checks out.
   b. Create a folder named `output`: Use this folder to store the build output.
   c. Create a file named `ios.build.xml`.

(continued on next slide)

Building the native binary for iOS (6 of 10)

The text in italic depends on your project:

1. *${wlProject}*
2. *${wlApp}*
3. *${mobilePlatform}*
4. *${configuration}*
5. *${certificate}*
6. *${provisioning.profile}*

d. Copy the following content into the `ios.build.xml` file:

```
<project name="build for iOS environments">
   <property name="rootDir" value="/Users/username/Documents/wl.build"/>
   <target name="build" >      <property name="loadDir" value="${rootDir}/load" />
      <property name="destdir" value="${loadDir}/build" />
      <property name="wlappdir" value="${loadDir}/${wlProject}/apps/${wlApp}" />
      <property name="xcodebuildCMD" value="/usr/bin/xcodebuild" />
      <property name="iosPackageCMD" value="xcrun" />
      <property name="xcodebuildCMDlog" value="${destdir}/xcodebuildCMD.log" />
      <property name="iosPackageCMDlog" value="${destdir}/iosPackageCMD.log" />
      <property name="local.ipaPath" value="${destdir}/distro" />
      <property name="configuration" value="Ad Hoc" />      <mkdir dir="${local.ipaPath}" />
      <!-- Build the iPhone native app -->
      <exec
         dir="${wlappdir}/${mobilePlatform}/native"
         executable="${xcodebuildCMD}"
         failonerror="false"
         output="${xcodebuildCMDlog}"
         resultproperty="xcodebuildCMDResult" >

        <arg line=" -configuration '${configuration}' -sdk iphoneos5.1" />
      </exec>

      <!-- Package into IPA -->
      <exec
         dir="${wlappdir}/${mobilePlatform}/native"
         executable="${iosPackageCMD}"
         failonerror="false"
         output="${iosPackageCMDlog}"
         resultproperty="iosPackageCMDResult" >

         <arg value="-sdk" />
         <arg value="iphoneos" />
         <arg value="PackageApplication" />
         <arg value="${wlappdir}/${mobilePlatform}/native/build/Release-iphoneos/${wlApp}.app" />
         <arg value="-o" />
         <arg value="${local.ipaPath}/${wlApp}.ipa" />
         <arg value="--sign" />
         <arg value="${certificate}" />
         <arg value="--embed" />
         <arg value="${provisioning.profile}" />
         <arg value="-verbose" />
      </exec>
   </target>
</project>
```

- The resulting directory structure looks like the following example:

```
wl.build
     load
     output
     ios.build.xml
```

Building the native binary for iOS (7 of 10)

4. Update the following parts of the `ios.build.xml` file to correspond with the target project:

Option	Description
${wlProject} (1)	Specify the name of the project.
${wlApp} (2)	Specify the name of the application.
${mobilePlatform} (3)	Specify **iphone** or **ipad**.
${configuration} (4)	Specify the configuration setting. – Default is **Ad Hoc** so that the resulting ipa can be installed over the air for QA purposes. – Can be overridden in the build definition with **Debug** or **Release**, depending on project need. – See the iOS development guide for details.
${certificate} (5)	Specify the name of the certificate that is configured in step 3 or 4. This name generally starts with “iPhone Developer:” or “iPhone Distribution:”.
${provisioning.profile} (6)	Specify the path to the .mobileprovision file which corresponds to the *${configuration}* configuration setting.

Building the native binary for iOS (8 of 10)

5. Create the build definition in Rational Team Concert:

   Tip: You must use the Eclipse Rational Team Concert client to complete this step.

   a. Open the Rational Team Concert Eclipse client and connect to the project area.
   b. In the Team Artifacts view, ensure that there is a connection to the target Rational Team Concert server.
   c. Connect to the project area that contains the Worklight projects.
   d. Within this project area, right-click **Builds** and click **New Build Definition**.
   e. Ensure that the target project area is selected and click **Next**.
   f. Select the Command Line-Jazz Build Engine template and click **Next**.
   g. Select the **Jazz Source Control** check box for Pre-Build.
   h. Ensure that all check boxes for Post-Build actions are cleared.
   i. Click **Finish**.
   – The editor for the new build definition opens.

Building the native binary for iOS (9 of 10)

6. Update the following values in the new build definition:

Option	Description
Supporting Build Engines	Specify the id of the build engine that is set up to run the iOS build.
Build File	Specify the path to the `ios.build.xml` file.
Build Target	Specify **build**.

Building the native binary for iOS (10 of 10)

7. Save the updated build definition.
8. Test the build:
   a. Ensure that the build engines on the workstations configured for iOS native builds are started, running, and waiting to receive build requests.
      - For more information about setting up Jazz build engines, see the Rational Team Concert Information Center.
   b. In the Eclipse Rational Team Concert client, right-click the new build definition and click **Request Build**.
      - You can monitor the status of the build in the Builds view.

Agenda

- Building Worklight projects with the Rational Team Concert build system
  - Building and deploying hybrid resources
  - Building the native binary for iOS
  - Building the native binary for Android

Building the native binary for Android (1 of 8)

- Before you build the native artifacts, perform a hybrid build (as described in the previous slides). Doing so ensures that the artifacts required by the Android native build are properly generated and updated.
- You can create a build definition to build Android native executables, by using the command-line interfaces provided with the Android SDK.
- Before you begin, ensure that the following software is installed on your build workstation:
  - Oracle Java™ Development Kit (JDK)
  - Android SDK (available from http://developer.android.com/sdk)
  - Ant (available from Apache.org)
  - **Important**: Ensure that you add the `bin` directory to the PATH system variable.
- To prepare the environment on the workstation that runs the Jazz™ build engine for your Android builds, perform the steps given in the following slides.

Building the native binary for Android (2 of 8)

1. Create a build folder, such as `~/Users/username/Documents/wl.build`
2. Within the build folder, complete the following steps:
   a. Create a folder named `load`: Use this folder to store the source code that Rational Team Concert™ checks out.
   b. Create a folder named `output`: Use this folder to store the build output.
   c. Create a file named `android.build.xml`.

(continued on next slide)

Building the native binary for Android (3 of 8)

The text in italic depends on your project:

1. *${wlProject}*
2. *${wlApp}*
3. *${androidSDKPath}*

d. Copy the following content into the `android.build.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
     <project name="build for Android environment">
        <property name="rootDir" value="/Users/username/Documents/wl.build"/>
        <!-- Worklight Build for Android  -->
        <target name="build">
           <property name="wlappdir" value = "${rootDir}/load/${wlProject}/apps/${wlApp}"/>
           <property name="androidCMD" value="${androidSDKPath}/tools/android.bat"/>
           <property name="destdir" value="${rootDir}/build" />
           <property name="androidCMDlog" value="${destdir}/androidCMD.log" />

           <!-- Generate the native build.xml -->
           <exec dir="${wlappdir}/android/native" executable="${androidCMD}
    " failonerror="true" resultproperty="androidCMDResult" output="${androidCMDlog}">
                 <arg line=" update project -p ${wlappdir}/android/native --target android-15" />
           </exec>

           <!-- Build the app -->
           <subant target="debug" buildpath="${wlappdir}/android/native" >
           </subant>
        </target>
     </project>
```

1 2 3

- The resulting directory structure looks like the following example:

```
wl.build
     load
     output
     android.build.xml
```

Building the native binary for Android (4 of 8)

3. Update the following parts of the `android.build.xml` file to correspond with the target project:

Option	Description
${wlProject} (1)	Specify the name of the project.
${wlApp} (2)	Specify the name of the application.
${androidSDKPath} (3)	Specify the path of the Android SDK.

Building the native binary for Android (5 of 8)

4. Configure the environment to sign the Android executable (.apk) files:
   a. Refer to the procedures available at: http://developer.android.com/guide/publishing/app-signing.html.
   b. Use the **keytool** command to generate a valid key pair.

Building the native binary for Android (6 of 8)

5. Create the build definition in Rational Team Concert:

   Tip: You must use the Eclipse Rational Team Concert client to complete this step.

   a. Open the Rational Team Concert Eclipse client and connect to the project area.
   b. In the Team Artifacts view, ensure that there is a connection to the target Rational Team Concert server.
   c. Connect to the project area that contains the Worklight projects.
   d. Within this project area, right-click **Builds** and click **New Build Definition**.
   e. Ensure that the target project area is selected and click **Next**.
   f. Select the Ant-Jazz Build Engine template and click **Next**.
   g. Select the **Jazz Source Control** check box for Pre-Build.
   h. Ensure that all check boxes for Post-Build actions are cleared.
   i. Click **Finish**.
   – The editor for the new build definition opens.

Building the native binary for Android (7 of 8)

6. Update the following values in the new build definition:

Option	Description
Supporting Build Engines	Specify the id of the build engine that is set up to run the Android build.
Build File	Specify the path to the `android.build.xml` file.
Build targets	Specify **build**.
Ant home	Specify the path to Apache Ant.
Ant argument	Specify **-verbose**.

Building the native binary for Android (8 of 8)

7. Save the updated build definition.
8. Test the build:
   a. Ensure that the build engines on the workstations configured for Android native builds are started, running, and waiting to receive build requests.
      - For more information about setting up Jazz build engines, see the Rational Team Concert Information Center.
      - Before the build engine is started, ensure that the environment variable JAVA_HOME exists and identifies the Oracle JDK installation location.
   b. In the Eclipse Rational Team Concert client, right-click the new build definition and click **Request Build**.
      - You can monitor the status of the build in the Builds view.

Agenda

- Building Worklight projects with the Rational Team Concert build system
  - Building and deploying hybrid resources
  - Building the native binary for iOS
  - Building the native binary for Android

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