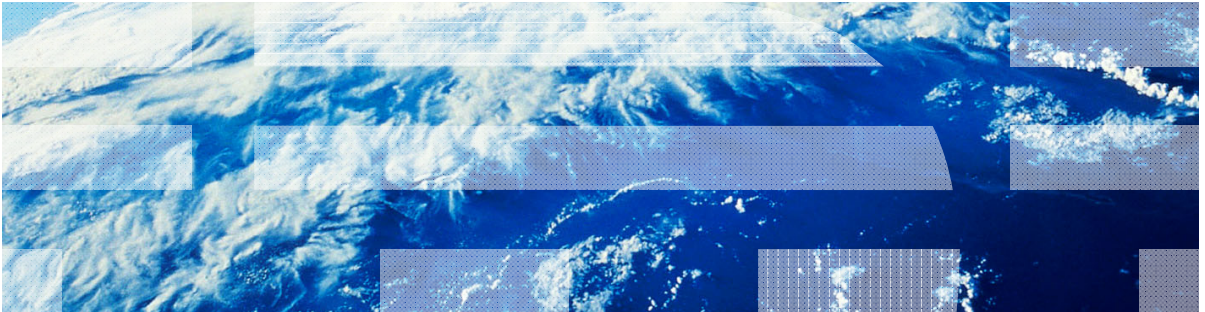


# ***IBM Worklight V5.0.5 Getting Started***

## **Module 7.11 – JSONStore with Synchronization**



## **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](http://www.ibm.com/legal/copytrade.shtml)” at [www.ibm.com/legal/copytrade.shtml](http://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

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- Sample

## ***Intermediate JSONStore Usage***

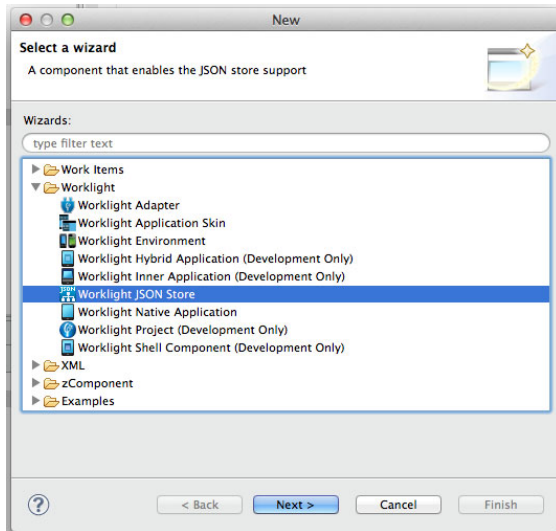
- This module covers intermediate-level tasks that can be performed on a local JSONStore collection, including the use of adapters to connect to a back-end.

## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- Sample

## Using the JSONStore wizard

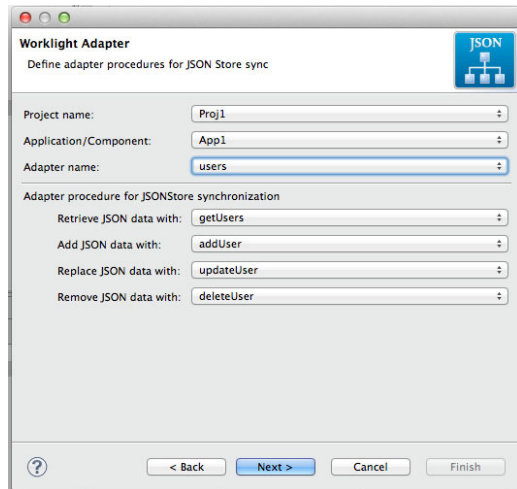
- To start the wizard, right-click the app that you want to set up and select **New > Other > Worklight > Worklight JSON Store**



## Using the JSONStore wizard – Choosing the adapter

Note: The adapter must already be deployed.

- Select the **user** adapter that is included in the sample project.
- In the adapter, the following sync operation procedures are defined
  - Retrieve (required)
  - Add
  - Replace
  - Remove



The screenshot shows a 'Worklight Adapter' configuration window. The title bar includes a 'JSON' icon. The main content area is titled 'Define adapter procedures for JSON Store sync'. It contains several input fields with dropdown menus:

- Project name: Proj1
- Application/Component: App1
- Adapter name: users

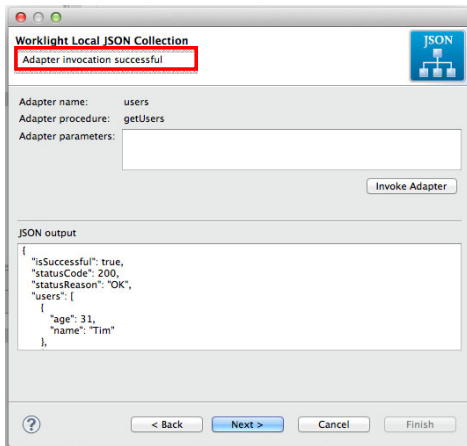
Below these fields is a section titled 'Adapter procedure for JSONStore synchronization' with four rows of configuration:

- Retrieve JSON data with: getUsers
- Add JSON data with: addUser
- Replace JSON data with: updateUser
- Remove JSON data with: deleteUser

At the bottom of the window, there are four buttons: a help icon (?), '< Back', 'Next >', 'Cancel', and 'Finish'.

## Using the JSONStore wizard – Testing the adapter

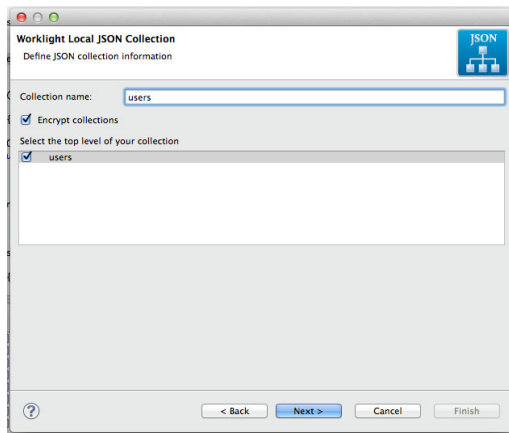
- In this step, you test adapter invocation
- After you click **Invoke Adapter**, you see the JSON output. A message is displayed that indicates whether the adapter invocation is a success or a failure.





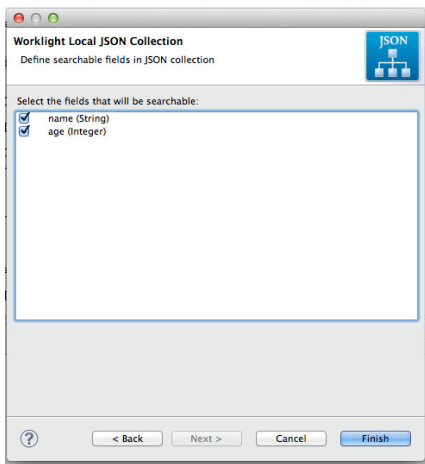
## Using the JSONStore wizard – Naming the Collection

- In this step, you define the JSON collection information
  1. Set **Collection name** to `users`
  2. Select the **Collection Encryption** check box
  3. Select the check box next to **users**



## Using the JSONStore wizard – Search fields

- Choose the fields that you want to make searchable.
- The searchable fields define the keys in JavaScript objects that are indexed, thus determining what you can query in a collection

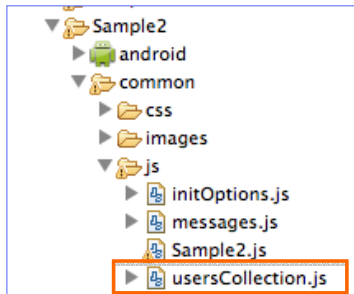


## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - **Generated Code**
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- Sample

## Generated Code

- When the wizard completes, the `usersCollection.js` file is generated in the `common/js` directory. This file contains the necessary code to initialize the local store and to connect it to the adapter.
- To link this file to your app, first add script tags to the generated file in the `app_name.html` file.



```
<div id='status'></div>
<table id='user_table' border='1'></table>

<script src="js/initOptions.js"></script>
<script src="js/Sample2.js"></script>
<script src="js/messages.js"></script>
<script src="js/usersCollection.js"></script>
</body>
</html>
```

## Generated Code

- Next, copy the last part of the generated code into the `wlCommonInit()` method inside the `app_name.js` file.
- This part of the code is called when the collection is ready for use and performs a `findAll` on the data.

usersCollection.js

```

if (WL.Client.getEnvironment() === 'iphone' ||
    WL.Client.getEnvironment() === 'ipad' ||
    WL.Client.getEnvironment() === 'android') {

    // Specify a password if you would like all collections encrypted.
    // Passwords should either be collected from the user or some other
    // secure source. The JSONStore password should not be hardcoded in
    // the application. Just 1 password per application is required,
    // not 1 password per collection.
    var pwd = prompt('Enter your password');
    WL.JSONStore.setPassword(pwd);
    pwd = null;

    WL.usersCollection.init();

    //You may want to capture the 'ready' event and do something like a 'findAll' with your collection.
    WLQ(document.body).bind('usersCollection/ready', function () {

        var onFindAllFailureCallback = function (err) {

            WL.Logger.debug('onFindAllFailureCallback: ' + err);
            //You may want to execute: WL.JSONStore.getErrorMessage(err);
        },
        onFindAllSuccessCallback = function (data) {

            //data is every document stored in this collection
            WL.Logger.debug(JSON.stringify(data));
        },

        //This is an example of how to get the collection instance and perform an operation on it (findAll).
        //Other valid operations on the instance are: push, replace, remove, add, among others.
        usersInstance = WL.usersCollection.getInstance();
        usersInstance.findAll({onFailure: onFindAllFailureCallback, onSuccess: onFindAllSuccessCallback});

    });

} //end if statement that checks for valid environments

```

app\_name.js

```

// Worklight comes with the jQuery framework
window.$ = window.jQuery = WLJQ;

```

```

function wlCommonInit(){
    // Common initialization code goes here

}

```

## Generated Code

- The following options are configured in the `usersCollection.js` file:
  - **userSearchFields** – Search fields you selected when you defined the searchable fields in the wizard.
  - **userAdapterOptions** – (Explained in further detail on following slide).
  - **onSuccess/onFailure Callbacks** – Called after the collection provisioning is attempted.
  - **Encryption** – If you chose to encrypt your data, the following code is also generated, to enable encryption. (Discussed in a subsequent module)

```
// Specify a password if you would like all collections encrypted.  
// Passwords should either be collected from the user or some other  
// secure source. The JSONStore password should not be hardcoded in  
// the application. Just 1 password per application is required,  
// not 1 password per collection.  
var pwd = prompt('Enter your password');  
WL.JSONStore.usePassword(pwd);  
pwd = null;
```

## Generated Code - userAdapterOptions

- userAdapterOptions
  - Contains the name of the adapter that is linked to this collection and the name of various procedures that perform operations when documents are pushed back to the server.
  - In addition to adapter procedures, `load` and `accept` objects are defined:
    - **Load** – determines the procedure that is used to load documents into the collection.
    - **Accept** – called every time an adapter procedure is invoked to push data to the back end service.

```
//Private Members:
var userSearchFields = {"age":"integer","name":"string"},
    userAdapterOptions = {
      name: 'user',
      replace: 'updateUser',
      remove: 'deleteUser',
      add: 'addUser',
      load: {
        procedure: 'getUsers',
        params: [],
        key: 'users'
      },
      accept: function (data) {
        return (data.status === 200);
      }
    }
},
```

## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- Sample



## Loading Data from an adapter – UI

Initialize the collection and load data from the user adapter into the local store.

For this demonstration, the previously generated code is wrapped inside a JavaScript event that will be fired when you click **init**.



To display the documents, clicking **Show All**, which performs a `findAll` on the collection.

## Loading Data from an adapter – The Code

```
usersCollection = WL.JSONStore.initCollection(  
    "users",  
    usersSearchFields,  
    {adapter: usersAdapterOptions,  
     onSuccess: initCollectionSuccessCallback,  
     onFailure: genericFailureCallback,  
     load:true});
```

In the generated code, the `initCollection` method has the `load` object set to `true`. This means that data is loaded by using the adapter that is linked to the collection.

```
var initCollectionSuccessCallback = function () {  
    logMessage('Collection has been initialized!');  
};
```

If initialization is successful, the `onSuccess` callback is fired. At this point, the collection is configured and ready.

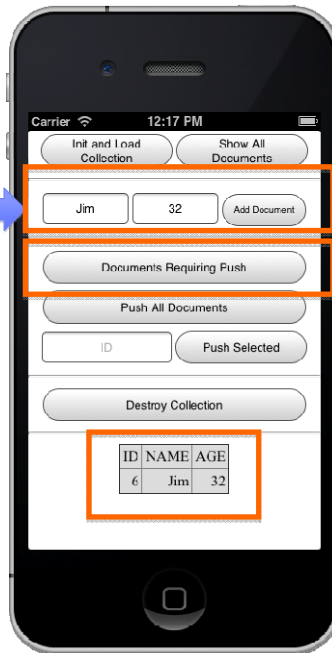
## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - **Get Documents Requiring Push**
  - Push All
  - Push Selected
- Sample

## Get Documents Requiring Push – UI

First, add a document or documents to the collection.

Any document added is initially marked as *requiring push*, which means it has not been pushed to the back-end.



Next, check what documents are currently marked as requiring push.

## Get Documents Requiring Push – The Code

```
WLJQ('button#push_required').bind('click', function () {  
    WL.Logger.debug('Called button#push_required');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var win = function (data) {  
        logMessage("Documents needing push : " + data );  
        reloadTable(data);  
    };  
  
    var options = {onSuccess: win, onFailure: genericFailureCallback};  
    usersCollection.getPushRequired(options);  
});
```

### Options

Define the `OnSuccess` callback. The variable `data` will contain an array of all documents that require push. The documents are displayed in a table.

## Get Documents Requiring Push – The Code

```
WLJQ('button#push_required').bind('click', function () {  
  
    WL.Logger.debug('Called button#push_required');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var win = function (data) {  
        logMessage("Documents needing push : " + data );  
        reloadTable(data);  
    };  
  
    var options = {onSuccess: win, onFailure: genericFailureCallback};  
    usersCollection.getPushRequired(options);  
});
```

### Get Documents Requiring Push

Finally, call the  
getPushRequired method with  
the onSuccess callback.

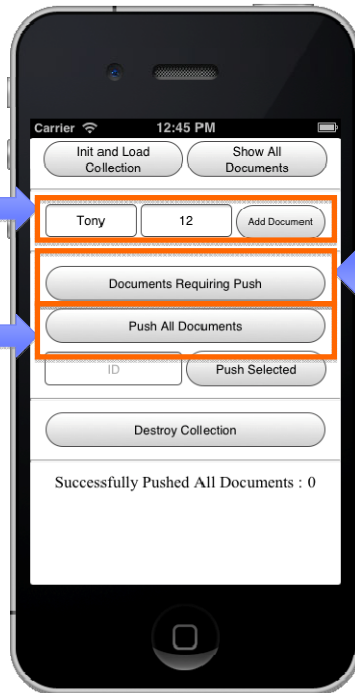
## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- Sample

## Push All – UI

First, add a new document to the collection.

Finally, push all documents that are marked as requiring push.



Then, check that it appears on the list of documents that require push.



## Push All – The Code

```
WLJQ('button#push_all').bind('click', function () {  
  
    WL.Logger.debug('Called button#push_all');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var win = function (data) {  
        logMessage("Successfully Pushed All Documents : " + data );  
    };  
  
    var options = {onSuccess: win, onFailure: genericFailureCallback};  
    usersCollection.push(options);  
  
});
```

### Options

Define the onSuccess callback. The variable **data** will contain a status code (defined in the JSONStore documentation).

## Push All – The Code

Push All Documents Requiring Push  
Finally, call the pushAll method  
with the options.

```
WLJQ('button#push_all').bind('click', function () {  
  
    WL.Logger.debug('Called button#push_all');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var win = function (data) {  
        logMessage("Successfully Pushed All Documents : " + data );  
    };  
  
    var options = {onSuccess: win, onFailure: genericFailureCallback};  
    usersCollection.push(options);  
});
```

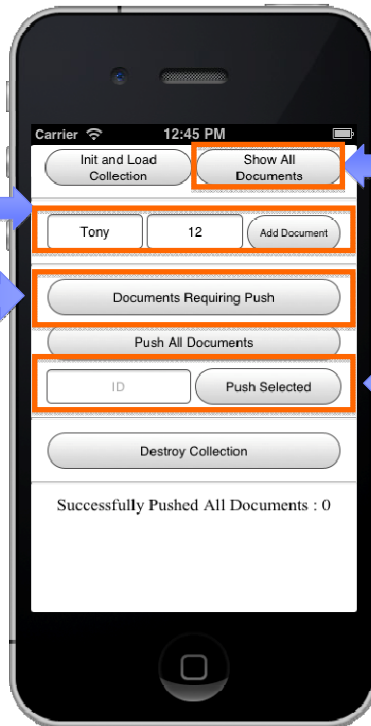
## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- Sample

## Push Selected – UI

First, add a new document to the collection.

Check that it appears on the list of documents that require push.



Next, show all documents and make note of the added document's id.

Finally, enter the document id and push that specific document only.

## Push Selected – The Code

```
WLJQ('button#push_selected').bind('click', function () {  
    WL.Logger.debug('Called button#push_selected');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var id = idTag.val();  
  
    if (id.length < 1 ) {  
        logMessage('You must provide a valid value for id');  
    } else {  
        var win = function (data) {  
            logMessage("Successfully pushed document : " + data);  
            idTag.val('');  
        };  
        var options = {onSuccess: win, onFailure: genericFailureCallback};  
  
        var doc = WL.JSONStore.documentify(parseInt(id),{});  
        usersCollection.pushSelected(doc, options);  
    }  
});
```

### Options

Define the onSuccess callback. The variable **data** will contain a status code (defined in the JSONStore docs).

## Push Selected – The Code

```
WLJQ('button#push_selected').bind('click', function () {  
    WL.Logger.debug('Called button#push_selected');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var id = idTag.val();  
  
    if (id.length < 1 ) {  
        logMessage('You must provide a valid value for id');  
    } else {  
  
        var win = function (data) {  
            logMessage("Successfully pushed document : " + data);  
            idTag.val('');  
        };  
  
        var options = {onSuccess: win, onFailure: genericFailureCallback};  
  
        var doc = WL.JSONStore.documentify(parseInt(id), {});  
        usersCollection.pushSelected(doc, options);  
    }  
});
```

### Create a new document

Create a new document object by using the documentify method (discussed in a subsequent module).

For this example, specify only an id and an empty object when you create the document.

## Push Selected – The Code

```
WLJQ('button#push_selected').bind('click', function () {  
    WL.Logger.debug('Called button#push_selected');  
    if (!checkColInit(usersCollection)) {return;}  
  
    var id = idTag.val();  
  
    if (id.length < 1 ) {  
        logMessage('You must provide a valid value for id');  
    } else {  
        var win = function (data) {  
            logMessage("Successfully pushed document : " + data);  
            idTag.val('');  
        };  
  
        var options = {onSuccess: win, onFailure: genericFailureCallback};  
  
        var doc = WL.JSONStore.documentify(parseInt(id),{});  
  
        usersCollection.pushSelected(doc, options);  
    }  
});
```

### Push the selected document

Finally we call our pushSelected method with the newly created document and options as parameters.

## Agenda

- Intermediate JSONStore Usage
  - Using the JSONStore wizard
  - Generated Code
  - Loading data from an adapter
  - Get Documents Requiring Push
  - Push All
  - Push Selected
- **Sample**



## Sample

- 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>

# 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***

