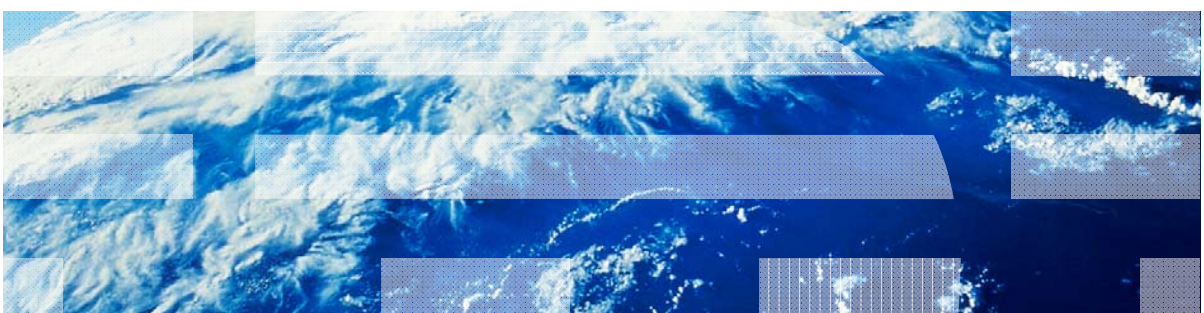
IBM Worklight V5.0.6
Getting Started

JSONStore – Encrypting sensitive data



15 March 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

- Advanced JSONStore Usage
  - Overview
  - Security
  - Multiple Users
  - Enhance
  - Documentify and Replace
  - Collection Removal
- Encryption
- Enhance
- Documentify and Replace
- Collection Removal
- Sample

Overview

- This module covers advanced tasks that you can perform on a local JSONStore collection, including encryption and management of multiple users.
- JSONStore provides a way to securely encrypt contents by using AES256 and PBKDF2.
- JSONStore supports the management of multiple users, and provides a way to limit the access to specific data.
- With JSONStore, you can add a function to a collection prototype.
- With the “documentify” method, you can create a document by using an ID and a JavaScript object.
- JSONStore supports the replacement of documents and the removal of collections.

Security

```
var options = {adapter: adapterDefinition, onSuccess: win, onFailure: fail};

//[Optional] You may assign a username to the store:
options.username = 'carlos';

//[Optional] If you want encryption you need to supply a password:
options.password = '12345';

var collection = WL.JSONStore.initCollection(name, searchFields, options);
```

- To encrypt a data store, use **WL.JSONStore.initCollection**.
  - Add a **password** field to the options object and specify its value.
  - This password is used to generate keys to encrypt data that is stored locally on the device.
- The store is always encrypted.

Multiple Users

```
var options = {adapter: adapterDefinition, onSuccess: win, onFailure: fail};

//[Optional] You may assign a username to the store:
options.username = 'carlos';

//[Optional] If you want encryption you need to supply a password:
options.password = '12345';

var collection = WL.JSONStore.initCollection(name, searchFields, options);
```

- To add a user name, use **WL.JSONStore.initCollection** method.
  - Add a **username** field to the **options** object, and specify its alphanumeric value.
  - This user name is used to create multiple stores that are stored locally on the device.
- After this initialization with a specific user name, you must call **WL.JSONStore.closeAll** before you can select another user (by calling **initCollection** again).

Enhance

```
enhance ( name , func , [options] ) Integer
```

- Use **enhance** to add a function to a collection prototype.
- You can create custom functions for tasks that might be performed frequently. You call these functions from your collection instance.
- For example, in the sample app, a custom function is created to find documents by using a user name.

Documentify and Replace

```
documentify ( id , data ) Document static

replace ( doc , [options] ) OnSuccess
```

- Use **documentify** to create JSONStore document objects.
  - Generally, you create the JSONStore document object with **documentify** by combining an ID field that is stored in the DOM with some raw object data. Then, you use this JSONStore document object as a parameter of the **replace** method.
- Use **replace** to update an entire document in a collection.

Collection Removal

```
removeCollection ( [options] ) OnSuccess
```

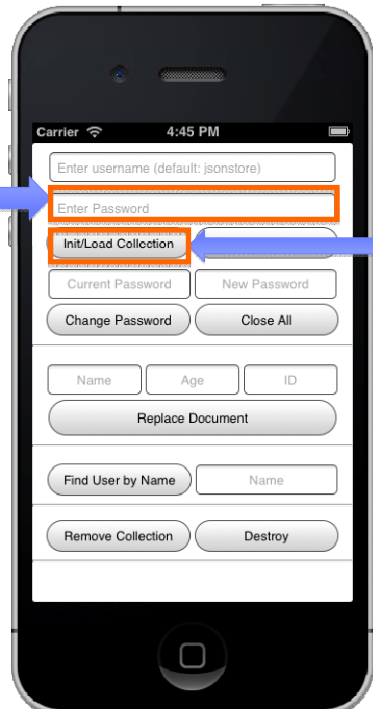
- To remove all data in a collection instance, use **removeCollection**.
- To remove all data in JSONStore, use **destroy**.
- To reuse a collection with the same name, you must reinitialize the collection.

Agenda

- Advanced JSONStore Usage
  - Overview
  - Security
  - Multiple Users
  - Enhance
  - Documentify and Replace
  - Collection Removal
- Encryption
- Enhance
- Documentify and Replace
- Collection Removal
- Sample

Encryption – Initialization

To enable encryption in the sample, specify a password.



Next, initialize, and load the collection.

Encryption – Initialization

```
var options = {adapter: usersAdapterOptions,
        onSuccess: initCollectionSuccessCallback,
        onFailure: genericFailureCallback,
        load:true};

if (pwd.length > 0) {

    options.password = pwd;
    WL.Logger.debug('Using Password');
    passwordTag.val('');

} else {

    WL.Logger.debug('Using NO Password');
}

if (usr.length > 0) {
    WL.Logger.debug('Using Username: ' + usr);
    options.username = usr;
}

usersCollection = WL.JSONStore.initCollection(
    "users",
    usersSearchFields, options),
```

Assigning a password

If a password was given, add it to the options object by using "**password**" as the key.

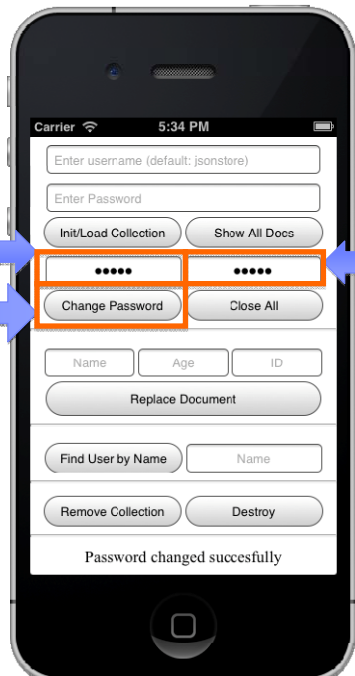
Encryption – Initialization

```
var options = {adapter: usersAdapterOptions,
        onSuccess: initCollectionSuccessCallback,
        onFailure: genericFailureCallback,
        load:true};


if (pwd.length > 0) {

    options.password = pwd;
    WL.Logger.debug('Using Password');
    passwordTag.val('');

} else {

    WL.Logger.debug('Using NO Password');
}

if (usr.length > 0) {
    WL.Logger.debug('Using Username: ' + usr);
    options.username = usr;
}

usersCollection = WL.JSONStore.initCollection(
    "users",
    usersSearchFields, options),
```

Assigning a user name

If a user name was given, add it to the options object by using "**username**" as the key.

Encryption – Changing a Password



Input your current password.

Specify a new password.

Change the password.

Encryption – Changing a Password

Process the input

Obtain the original password and the new one as specified in the inputs, and assign them to variables.

```
WLJQ('button#change_password').bind('click', function () {

    WL.Logger.debug('Called button#change_password');

    var password_old = passwordOldTag.val(),
        password_new = passwordNewTag.val();

    if (password_old.length < 1 || password_new.length < 1 ||
        checkUndefOrNull(JSONStoreInstance)) {
        logMessage('You must enter your old password as well as the new one.');
    } else {

        var win =   function (data) {
            logMessage('Password was successfully changed : ' + data);
            passwordOldTag.val('');
            passwordNewTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        WL.JSONStore.changePassword(password_old,password_new,options);
    }
});//END Change Password
```

Encryption – Changing a Password

```
WLJQ('button#change_password').bind('click', function ()

    WL.Logger.debug('Called button#change_password');

    var password_old = passwordOldTag.val(),
        password_new = passwordNewTag.val();

    if (password_old.length < 1 || password_new.length <
        checkUndefOrNull(JSONStoreInstance)) {
        logMessage('You must enter your old password as well as the new one.');
    } else {

        var win =   function (data) {
            logMessage('Password was successfully changed : ' + data);
            passwordOldTag.val('');
            passwordNewTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        WL.JSONStore.changePassword(password_old,password_new,options);
    }
});//END Change Password
```

Options

Define the **OnSuccess** callback.
The **data** variable will contain a status code.
The value of this code is defined in the JSONStore documentation.

Encryption – Changing a Password

```
WLJQ('button#change_password').bind('click', function () {

    WL.Logger.debug('Called button#change_password');

    var password_old = passwordOldTag.val(),
        password_new = passwordNewTag.val();

    if (password_old.length < 1 || password_new.length < 1
        checkUndefOrNull(JSONStoreInstance)) {
        logMessage('You must enter your old password as we
    } else {

        var win =   function (data) {
            logMessage('Password was successfully changed
            passwordOldTag.val('');
            passwordNewTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        WL.JSONStore.changePassword(password_old,password_new,options);
    }
});//END Change Password
```

Change the password

Finally, call **changePassword** with the old and new passwords as parameters. On success, the password is changed.

Agenda

- Advanced JSONStore Usage
  - Overview
  - Security
  - Multiple Users
  - Enhance
  - Documentify and Replace
  - Collection Removal
- Encryption
- Enhance
- Documentify and Replace
- Collection Removal
- Sample

Enhance – UI

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

Create a custom function

Define a custom function that creates a query and searches the `users` collection by user name. This function accepts a user name and an **onSuccess** callback as parameters.

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

Add the function to the collection prototype

Call **enhance** with the following parameters:

- A name for the custom function
- The custom function itself

The function is added to the collection prototype and is ready for use.

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

onSuccess Callback

Define an **onSuccess** callback for you to verify that the custom function successfully executes.

Enhance – The Code

```
WLJQ('button#enhance').bind('click', function () {

    WL.Logger.debug('Called button#enhance');

    var name = enhanceTag.val();

    if ( name.length < 1 ) {
        logMessage('You must provide a valid search value');
        return;
    }

    //Create a custom function
    var custFunction = function (user_name,onSuccess) {

        var query = {name: user_name};

        var win =  function (data) {
            onSuccess(data);
        };

        var fail =  function (data) {
            genericFailureCallback(data);
        };

        var options = {onSuccess: win, onFailure: fail};
        usersCollection.find(query, options);
    };

    usersCollection.enhance('getUserByName', custFunction);

    var onSuccess = function (data) {
        reloadTable(data);
        enhanceTag.val('');
    };

    usersCollection.getUserByName(name, onSuccess);
});//END Enhance
```

Calling the custom function

Finally, use the JSONStore instance to call **getUserByName**.

The parameters of **getUserByName** are:

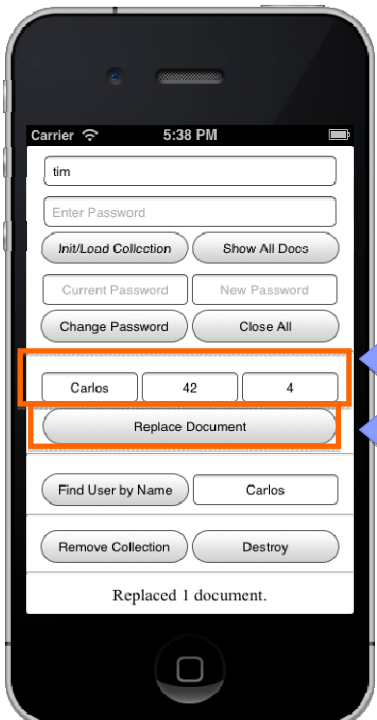
- The **name** to search for
- The **onSuccess** callback

On success, you receive an array of matching documents that are then displayed in a table.

Agenda

- Advanced JSONStore Usage
  - Overview
  - Security
  - Multiple Users
  - Enhance
  - Documentify and Replace
  - Collection Removal
- Encryption
- Enhance
- Documentify and Replace
- Collection Removal
- Sample

Documentify and Replace – UI



Specify a new name and age, and the ID of the user in the collection that you want to replace.

Users who match the specified ID in the local store are all replaced.

Documentify and Replace – The Code

```
WLJQ('button#replace').bind('click', function () {

    WL.Logger.debug('Called button#replace');
    if (!checkColInit(usersCollection)) {return;}

    var name = nameTag.val(),
        id = idTag.val(),
        age = ageTag.val();

    if (id.length < 1 ) {
        logMessage('You must provide a valid value for ID');
    } else {
        var doc = WL.JSONStore.documentify(id, {name: name, age: age});

        var win =   function (data) {
            logMessage('Replaced ' + data + ' document.');
            //Clear Text Boxes
            nameTag.val('');
            ageTag.val('');
            idTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        usersCollection.replace(doc,options);
    }
});
```

Documentify

To call **replace**, you require a document object.
You can construct a document object from the user input by using **documentify**:

- Use an ID and JSON data as the parameters.
- The result is a JSONStore document object.

Documentify and Replace – The Code

```
WLJQ('button#replace').bind('click', function () {

    WL.Logger.debug('Called button#replace');
    if (!checkColInit(usersCollection)) {return;}

    var name = nameTag.val(),
        id = idTag.val(),
        age = ageTag.val();

    if (id.length < 1 ) {
        logMessage('You must provide a valid value for ID')
    } else {

        var doc = WL.JSONStore.documentify(id, {name: name,  age});

        var win =   function (data) {
            logMessage('Replaced ' + data + ' document.');
            //Clear Text Boxes
            nameTag.val('');
            ageTag.val('');
            idTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        usersCollection.replace(doc,options);
    }
});
```

Options

Define the **OnSuccess** callback of the **replace** method. The **data** variable contains the number of replaced documents.

Documentify and Replace – The Code

```
WLJQ('button#replace').bind('click', function () {

    WL.Logger.debug('Called button#replace');
    if (!checkColInit(usersCollection)) {return;}

    var name = nameTag.val(),
        id = idTag.val(),
        age = ageTag.val();

    if (id.length < 1 ) {
        logMessage('You must provide a valid value for ID');
    } else {

        var doc = WL.JSONStore.documentify(id, {name: name, age: age});

        var win =   function (data) {
            logMessage('Replaced ' + data + ' document.');
            //Clear Text Boxes
            nameTag.val('');
            ageTag.val('');
            idTag.val('');
        };

        var options = {onSuccess: win, onFailure: genericFailureCallback};

        usersCollection.replace(doc,options);
    }
});
```

Replace the Document

Finally, call **replace** by using the newly created document object and the options as parameters.

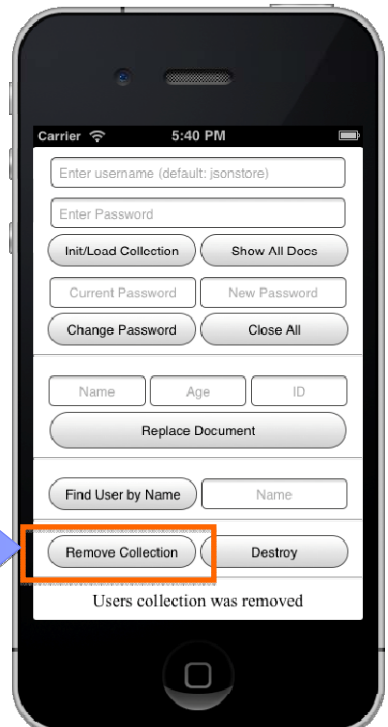
To verify that the document is replaced, display all the documents in the collection by using the **Show all documents** button.

Agenda

- Advanced JSONStore Usage
  - Overview
  - Security
  - Multiple Users
  - Enhance
  - Documentify and Replace
  - Collection Removal
- Encryption
- Enhance
- Documentify and Replace
- Collection Removal
- Sample

Collection Removal – UI

Remove the entire `users` collection. The local store still exists, and contains all the collections that were not removed yet.



Collection Removal – The Code

Options

Define the **onSuccess** callback. The **data** variable contains a `boolean` value that indicates whether the `users` collection was removed.

```
WLJQ('button#remove_collection').bind('click', function () {

    WL.Logger.debug('Called button#remove_collection');

    var win =  function (data) {
        if (data === 0) {
            logMessage('Users collection was removed');
        }else{
            logMessage('Users Collection was not removed');
        }
    };

    var options = {onSuccess: win, onFailure: genericFailureCallback};

    usersCollection.removeCollection(options);
});
```

Collection Removal – The Code

Remove the Collection

Finally, call **removeCollection** by using your options as the parameter.

```
WLJQ('button#remove_collection').bind('click', function

    WL.Logger.debug('Called button#remove_collection');

    var win =  function (data) {
        if (data === 0) {
            logMessage('Users collection was removed');
        }else{
            logMessage('Users Collection was not removed');
        }
    };

    var options = {onSuccess: win, onFailure: genericFailureCallback};

    usersCollection.removeCollection(options);
});
```

Agenda

- Advanced JSONStore Usage
  - Overview
  - Security
  - Multiple Users
  - Enhance
  - Documentify and Replace
  - Collection Removal
- Encryption
- Enhance
- Documentify and Replace
- Collection Removal
- 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.

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/en/us sections entitled "Cookies, Web Beacons and Other Technologies" and "Software Products and Software-as-a-Service".

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