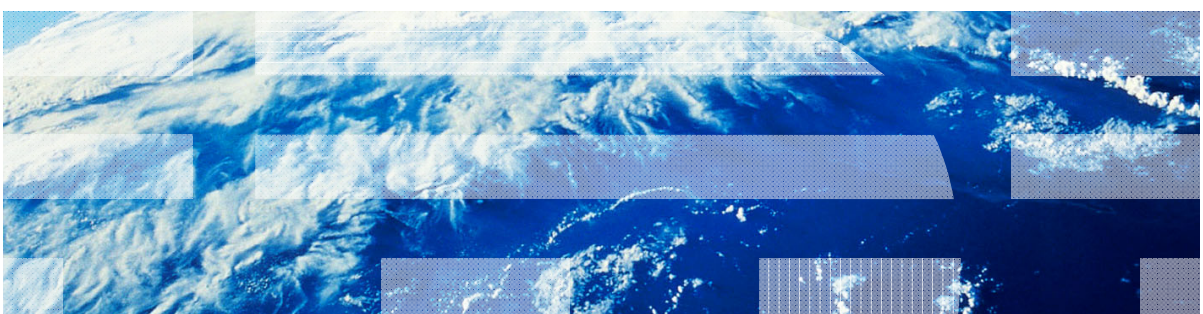
IBM Worklight V5.0.5
Getting Started

Module 22 – Adapter-Based Authentication



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

- The Adapter-based authentication introduction
- Configuring the authenticationConfig.xml
- Creating the server-side authentication components
- Creating the client-side authentication components
- Examining the result
- Exercise

The Adapter-based authentication introduction

- The Adapter-based authentication is the most flexible type of authentication to implement and contains all the benefits of the Worklight® Server authentication framework.
- When you use the adapter-based authentication, the entire authentication logic, including the credentials validation, can be implemented in an adapter by using plain JavaScript.
- Nevertheless, any login module can be used in the adapter-based authentication as an additional authentication layer.
- In this module, you implement an adapter-based authentication mechanism that relies on a user name and a password.

Agenda

- The Adapter-based authentication introduction
- Configuring the authenticationConfig.xml
- Creating the server-side authentication components
- Creating the client-side authentication components
- Examining the result
- Exercise

Configuring the authenticationConfig.xml

- Add an authentication realm to the <realms> section of the **authenticationConfig.xml** file and call it **SingleStepAuthRealm.**

```
<realm name="SingleStepAuthRealm" loginModule="SingleStepAuthLoginModule">
    <className>com.worklight.integration.auth.AdapterAuthenticator</className>
    <parameter name="login-function" value="SingleStepAuthAdapter.onAuthRequired" />
    <parameter name="logout-function" value="SingleStepAuthAdapter.onLogout" />
</realm>
```

- This realm uses the **SingleStepAuthLoginModule** login module that you will define later.

Configuring the authenticationConfig.xml

- Add an authentication realm to the <realms> section of the **authenticationConfig.xml** file and call it **SingleStepAuthRealm.**

```
<realm name="SingleStepAuthRealm" loginModule="SingleStepAuthLoginModule">
    <className>com.worklight.integration.auth.AdapterAuthenticator</className>
    <parameter name="login-function" value="SingleStepAuthAdapter.onAuthRequired" />
    <parameter name="logout-function" value="SingleStepAuthAdapter.onLogout" />
</realm>
```

- Using the **com.worklight.integration.auth.AdapterAutenticator** class means that the server-side part of the authenticator is defined in the adapter.

Configuring the authenticationConfig.xml

- Add an authentication realm to the <realms> section of the **authenticationConfig.xml** file and call it **SingleStepAuthRealm.**

```
<realm name="SingleStepAuthRealm" loginModule="SingleStepAuthLoginModule">
    <className>com.worklight.integration.auth.AdapterAuthenticator</className>
    <parameter name="login-function" value="SingleStepAuthAdapter.onAuthRequired" />
    <parameter name="logout-function" value="SingleStepAuthAdapter.onLogout" />
</realm>
```

- When the Worklight authentication framework detects an attempt to access a protected resource, an adapter function that is defined in a login-function parameter is invoked automatically.
- When logout is detected (explicit or session timeout), a logout-function is invoked automatically.
- In both cases, the parameter value syntax is the adapterName.functionName.

Configuring the authenticationConfig.xml

- Add a login module to the <loginModules> section of the **authenticationConfig.xml** file and call it **SingleStepAuthLoginModule.**

```
<loginModule name="SingleStepAuthLoginModule" canBeResourceLogin="true" isIdentityAssociationKey="false">
    <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className>
</loginModule>
```

- Using a **NonValidatingLoginModule** class name means that no additional validation is performed by the Worklight Platform, and the developer takes responsibility for the credential validation within the adapter.

Configuring the authenticationConfig.xml

- Add a security test to the <securityTests> section of the **authenticationConfig.xml** file.
- You will use this security test to protect the adapter procedure, so make it a **<customSecurityTest>.**

```
<securityTests>
    <customSecurityTest name="SingleStepAuthAdapter-securityTest">
        <test isInternalUserID="true" realm="SingleStepAuthRealm"/>
    </customSecurityTest>
</securityTests>
```

- Remember the security test name. You will use it in subsequent slides.

Agenda

- The Adapter-based authentication introduction
- Configuring the authenticationConfig.xml
- Creating the server-side authentication components
- Creating the client-side authentication components
- Examining the result
- Exercise

Creating the server-side authentication components

- The following diagram illustrates the adapter-based authentication process:

Creating the server-side authentication components

- Create an adapter that will take care of the authentication process. Name it **SingleStepAuthAdapter.**
- The **SingleStepAuthAdapter** will have the two following procedures:

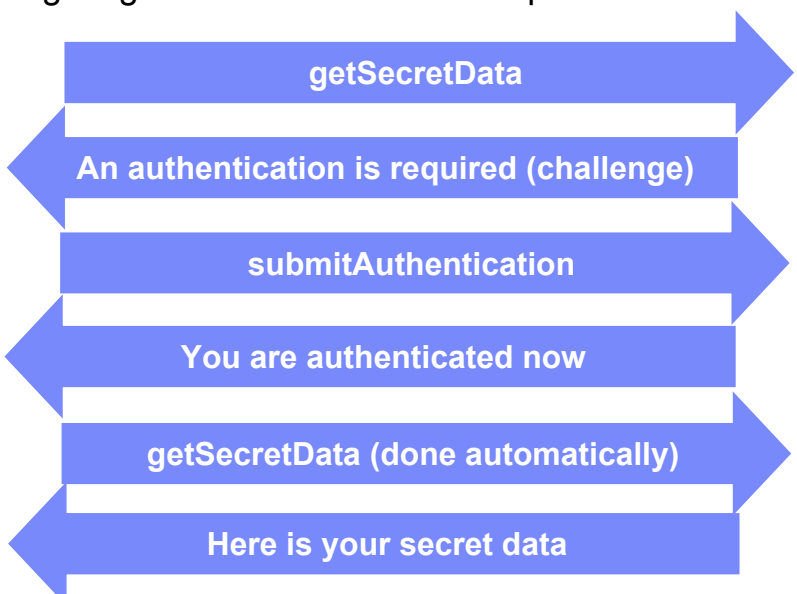
```
<procedure name="submitAuthentication"/>

<procedure name="getSecretData" securityTest="SingleStepAuthAdapter-securityTest"/>
```

- The **submitAuthentication** procedure is taking care of the authentication process and authentication is not required in order to invoke it.
- The second procedure, however, is available to authenticated users only.

Creating the server-side authentication components

- The following diagram shows the flow to implement:

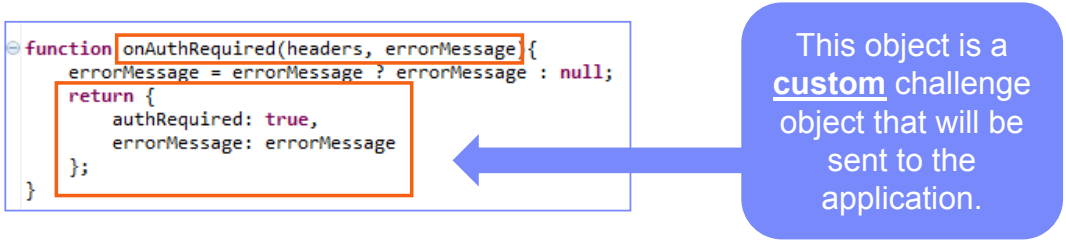
Application



Worklight Server

Creating the server-side authentication components

- Whenever the IBM Worklight framework detects an unauthenticated attempt to access a protected resource, the onAuthRequired function is invoked (as defined in the **authenticationConfig.xml**).



- This function receives the response headers and an optional **errorMessage** parameter. The object that is returned by this function is sent to the client application.
- Note the **authRequired: true** property. You will use this property in challenge handler to detect that server is requesting authentication.

Creating the server-side authentication components

- The **submitAuthentication** function is invoked by a client application to validate user name and password.

```
function submitAuthentication(username, password){
    if (username==="worklight" && password === "worklight"){

        var userIdentity = {
                userId: username,
                displayName: username,
                attributes: {
                    foo: "bar"
                }
        };

        WL.Server.setActiveUser("SingleStepAuthRealm", userIdentity);

        return {
            authRequired: false
        };
    }

    return onAuthRequired(null, "Invalid login credentials");
}
```

The user name and password are received from the application as parameters.

Creating the server-side authentication components

- The **submitAuthentication** function is invoked by a client application to validate user name and password.

```
function submitAuthentication(username, password){
    if (username==="worklight" && password === "worklight"){

        var userIdentity = {
                userId: username,
                displayName: username,
                attributes: {
                    foo: "bar"
                }
        };

        WL.Server.setActiveUser("SingleStepAuthRealm", userIdentity);

        return {
            authRequired: false
        };
    }

    return onAuthRequired(null, "Invalid login credentials");
}
```

In this sample, the credentials are validated against some hardcoded values, but any other validation can be performed, for example by using SQL or WebServices.

Creating the server-side authentication components

- The **submitAuthentication** function is invoked by a client application to validate user name and password.

```
function submitAuthentication(username, password){
    if (username==="worklight" && password === "worklight"){

        var userIdentity = {
                userId: username,
                displayName: username,
                attributes: {
                    foo: "bar"
                }
        };

        WL.Server.setActiveUser("SingleStepAuthRealm", userIdentity);

        return {
            authRequired: false
        };
    }

    return onAuthRequired(null, "Invalid login credentials");
}
```

If validation is successfully passed, a WL.Server.setActiveUser API is called to create an authenticated session for the SingleStepAuthRealm with a user data stored in a userIdentity object. Note, you can add your own custom properties to the user identity attributes.

Creating the server-side authentication components

- The **submitAuthentication** function is invoked by a client application to validate user name and password.

```
function submitAuthentication(username, password){
    if (username==="worklight" && password === "worklight"){

        var userIdentity = {
                userId: username,
                displayName: username,
                attributes: {
                    foo: "bar"
                }
        };

        WL.Server.setActiveUser("SingleStepAuthRealm", userIdentity);

        return {
            authRequired: false
        };
    }

    return onAuthRequired(null, "Invalid login credentials");
}
```

An object is sent to the application, stating that the authentication screen is no longer required.

Creating the server-side authentication components

- The **submitAuthentication** function is invoked by a client application to validate user name and password.

```
function submitAuthentication(username, password){
    if (username==="worklight" && password === "worklight"){

        var userIdentity = {
                userId: username,
                displayName: username,
                attributes: {
                    foo: "bar"
                }
        };

        WL.Server.setActiveUser("SingleStepAuthRealm", userIdentity);

        return {
            authRequired: false
        };
    }

    return onAuthRequired(null, "Invalid login credentials");
}
```

If the credentials validation fails, an object that is built by the onAuthRequired function is returned to the application with a corresponding error message.

Creating the server-side authentication components

- For training purposes, the **getSecretData** function returns a hardcoded value. Keep in mind that the **getSecretData** is protected by a security test, as defined in the adapter XML.
- The **onLogout** function is defined in the **authenticationConfig.xml** file to be invoked automatically on logout (for example to perform a cleanup).

```
function getSecretData(){
    return {
        secretData: "A very very very very secret data"
    };
}

function onLogout(){
    WL.Logger.debug("Logged out");
}
```

Agenda

- The Adapter-based authentication introduction
- Configuring the authenticationConfig.xml
- Creating the server-side authentication components
- Creating the client-side authentication components
- Examining the result
- Exercise

Creating the client-side authentication components

- Create a Worklight application.
- The application consists of two main elements:
  - The ** ** element is used to display the application content.
  - The ** ** element is used for authentication form purposes.
- When the authentication is required, the application hides the AppDiv and shows the AuthDiv. When the authentication is complete, it does the opposite.

Creating the client-side authentication components

- Start by creating an AppDiv.
- It has a basic structure and functions:

```
<div id="AppDiv">
    <div class="header">
        <h1>SingleStepAdapterBasedAuthentication</h1>
    </div>
    <input type="button" value="Get secret data" onclick="getSecretData()" />
    <input type="button" value="Logout"
           onclick="WL.Client.logout('SingleStepAuthRealm', {onSuccess:WL.Client.reloadApp})" />
    <div id="ResponseDiv"></div>
</div>
```

- The buttons are used to invoke the **getSecretData** procedure and to log out.
- The ** ** is used to display the **getSecretData** response.

Creating the client-side authentication components

- The AuthDiv contains the following elements:

```
<div id="AuthDiv" style="display:none">
    <p id="AuthInfo"></p>
    <hr />
    <input type="text" placeholder="Enter username" id="AuthUsername"/><br />
    <input type="password" placeholder="Enter password" id="AuthPassword"/><br />
    <input type="button" value="Submit" id="AuthSubmitButton" />
    <input type="button" value="Cancel" id="AuthCancelButton" />
</div>
```

  - The AuthInfo to display error messages.
  - The AuthUsername and the AuthPassword to input elements.
  - The AuthSubmitButton and the AuthCancelButton.
- The AuthDiv is styled as **display:none** because it should not be displayed before the authentication is requested by server.

Creating the client-side authentication components

- Finally, create a challenge handler.
- Use the following API to create this handler and implement its functionality.

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

Use the **WL.Client.createChallengeHandler()** to create a challenge handler object. A realm name must be supplied as a parameter.

Creating the client-side authentication components

- Finally, create a challenge handler.
- Use the following API to create this handler and implement its functionality.

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

The **isCustomResponse** function of the challenge handler will be called each time that a response is received from the server. It is used to detect whether the response contains data that is related to this challenge handler. It should return **true** or **false.**

Creating the client-side authentication components

- Finally, create a challenge handler.
- Use the following API to create this handler and implement its functionality.

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

If the isCustomResponse returns true, the framework will call the handleChallenge() function. This function is used to perform required actions, such as hide the application screen and show the login screen.

Creating the client-side authentication components

- In addition to the methods that the developer must implement, the challenge handler contains functionalities that the developer might want to use:
  - The **myChallengeHandler.submitAdapterAuthentication()** is used to send collected credentials to a specific adapter procedure. It has the same signature as the **WL.Client.invokeProcedure()** API.
  - The **myChallengeHandler.submitSuccess()** will notify the Worklight framework that the authentication successfully finished. The Worklight framework will then automatically issue the original request that triggered the authentication.
  - The **myChallengeHandler.submitFailure()** will notify the Worklight framework that the authentication completed with failure. The Worklight framework will then dispose the original request that triggered the authentication.
- You will use these functions during the implementation of the challenge handler in the next slides.

Creating the client-side authentication components

- Implement the **isCustomResponse** function. It should detect whether the server response contains the challenge object that you defined previously.
- You defined challenge object in the adapter. You now use its authRequired property.

```
return {
    authRequired: true,
    errorMessage: errorMessage
};
```

```
singleStepAuthRealmChallengeHandler.isCustomResponse = function(response) {
    if (!response || !response.responseJSON || response.responseText === null) {
        return false;
    }
    if (typeof(response.responseJSON.authRequired) !== 'undefined'){
        return true;
    } else {
        return false;
    }
};
```

- Return true if the authRequired property is found, false otherwise.

Creating the client-side authentication components

- Implement the **handleChallenge** function. It should prepare the authentication UI.
- Use the optional errorMessage property of the challenge object.

```
return {
    authRequired: true,
    errorMessage: errorMessage
};
```

```
singleStepAuthRealmChallengeHandler.handleChallenge = function(response){
    var authRequired = response.responseJSON.authRequired;

    if (authRequired == true){
        $("#AppDiv").hide();
        $("#AuthDiv").show();
        $("#AuthPassword").empty();
        $("#AuthInfo").empty();

        if (response.responseJSON.errorMessage)
            $("#AuthInfo").html(new Date() + " :: " + response.responseJSON.errorMes

    } else if (authRequired == false){
        $("#AppDiv").show();
        $("#AuthDiv").hide();
        singleStepAuthRealmChallengeHandler.submitSuccess();
    }
};
```

If the authRequires is true, it shows login screen, cleans up password field, and shows an errorMessage (if present).

Creating the client-side authentication components

- Implement the handleChallenge function. It should prepare the authentication UI.
- Use the optional errorMessage property of the challenge object.

```
return {
    authRequired: true,
    errorMessage: errorMessage
};
```

```
singleStepAuthRealmChallengeHandler.handleChallenge = function(response){
    var authRequired = response.responseJSON.authRequired;

    if (authRequired == true){
        $("#AppDiv").hide();
        $("#AuthDiv").show();
        $("#AuthPassword").empty();
        $("#AuthInfo").empty();

        if (response.responseJSON.errorMessage)
            $("#AuthInfo").html(new Date() + " :: " + response.responseJS

    } else if (authRequired == false){
        $("#AppDiv").show();
        $("#AuthDiv").hide();
        singleStepAuthRealmChallengeHandler.submitSuccess();
    }
};
```

If the authRequired is false, it shows AppDiv, it hides AuthDiv, and it notifies the Worklight framework that the authentication successfully completed.

Creating the client-side authentication components

- Clicking a login button will trigger the function that collects the user name and the password from the HTML input fields, and submits them to the adapter.
- Notice that the challenge handler the **submitAdapterAuthentication** method is used.

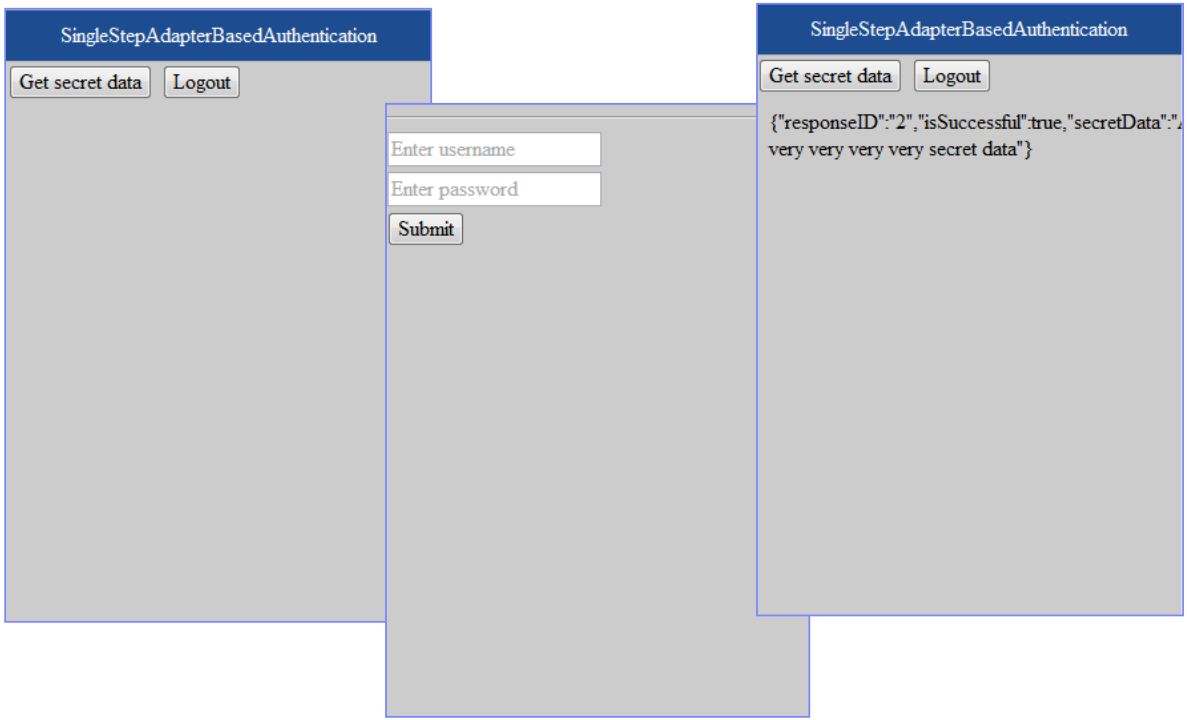
```
$("#AuthSubmitButton").bind('click', function () {
    var username = $("#AuthUsername").val();
    var password = $("#AuthPassword").val();

    var invocationData = {
        adapter : "SingleStepAuthAdapter",
        procedure : "submitAuthentication",
        parameters : [ username, password ]
    };

    singleStepAuthRealmChallengeHandler.submitAdapterAuthentication(invocationData, {});
});
```

- There is no need to specify the callbacks because the response will be checked by the IBM Worklight framework.

Agenda

- The Adapter-based authentication introduction
- Configuring the authenticationConfig.xml
- Creating the server-side authentication components
- Creating the client-side authentication components
- Examining the result
- Exercise

Examining the result



Agenda

- The Adapter-based authentication introduction
- Configuring the authenticationConfig.xml
- Creating the server-side authentication components
- Creating the client-side authentication components
- Examining the result
- Exercise

Exercise

- Examine the Single Step Adapter-based Authentication sample.
- Update your **authenticationConfig.xml** file according to the training module (included in the sample).
- Implement a two-step authentication flow according to the following rules:
  - Step 1 – The user name and the password validation as described in the training module.
  - Step 2 - After the user name and the password are verified, a secret question should be sent to the application (for example “What is your pet’s name”).
  - An answer should be collected from the user and sent to the server (the adapter).
  - Only if the answer to the secret question passes validation, an authenticated session should be established (the WL.Server.setActiveUser() API).
- 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

Check yourself questions

- When you define a realm that is using an adapter-based authentication in the authenticationConfig.xml, which two parameters are mandatory?
  - The login-function, the logout-function.
  - The adapter-name, the realm-name.
  - The adapter-name, the login-function.
  - The login-function, the login-module.
- How can a developer specify which adapter procedures will be protected by an authentication realm?
  - When the authentication realm is specified in the adapter XML file, all the adapter procedures are protected by it.
  - The developer does not need to specify it. Authentication credentials should be added on the client side when you use WL.Client.invokeProcedure for the procedure to work.
  - By adding a securityTest property to the procedure definition in the adapter XML.
  - You cannot protect the adapter procedures by an authentication realm. The protection is for applications only.
- What client side mechanism is used to detect that the server requires an authentication for the client request?
  - The challengeHandler.isAuthenticationRequired
  - The challengeHandler.isUserAuthenticated
  - The challengeHandler.analyzeServerResponse
  - The challengeHandler.isCustomResponse

Check yourself questions

- When you define a realm that is using an adapter-based authentication in the authenticationConfig.xml, which two parameters are mandatory?
  - The login-function, the logout-function.
  - The adapter-name, the realm-name.
  - The adapter-name, the login-function.
  - The login-function, the login-module.
- How can a developer specify which adapter procedures will be protected by an authentication realm?
  - When the authentication realm is specified in the adapter XML file, all the adapter procedures are protected by it.
  - The developer does not need to specify it. Authentication credentials should be added on the client side when you use WL.Client.invokeProcedure for the procedure to work.
  - By adding a securityTest property to the procedure definition in the adapter XML.
  - You cannot protect the adapter procedures by an authentication realm. The protection is for applications only.
- What client side mechanism is used to detect that the server requires an authentication for the client request?
  - The challengeHandler.isAuthenticationRequired
  - The challengeHandler.isUserAuthenticated
  - The challengeHandler.analyzeServerResponse
  - The challengeHandler.isCustomResponse

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