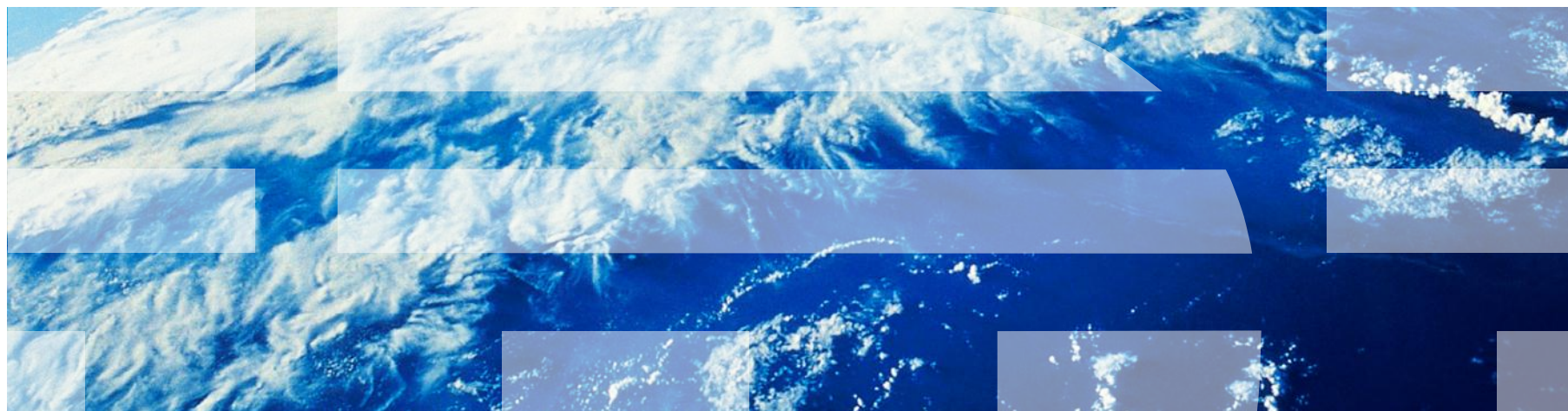


IBM Worklight Foundation V6.2.0

Getting Started

Using Worklight Server to authenticate external resources



Trademarks

- IBM, the IBM logo, ibm.com, and Worklight are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. 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.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Node.js is an official trademark of Joyent. This documentation is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.
- 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

- Feature overview
- Worklight authentication by using an access token over OAuth 2.0
- Worklight project configuration
- External service configuration
- Use of the client-side API
- Exercise

Feature overview (1 of 2)

- By using Worklight Server to authenticate external resources, you can use Single Sign-On (SSO) between IBM Worklight Foundation and external services.
- This feature protects those services through the Worklight Security Framework.
- Worklight Server acts as an authorization server and issues an access token that can be validated by the external service.
- Client applications request the access token from Worklight via token endpoint and send it to the external services.

Feature overview (2 of 2)

- The scope of the access token is a security test that is defined inside a Worklight project.
- Each scope has a timeout property that determines the lifetime of the token. For example, if the token timeout is configured to be 30 seconds, the token remains valid for 30 seconds after Worklight Server has issued it. After that time, the token is rejected by the external service and a new token must be requested and issued.

Agenda

- Feature overview
- Worklight authentication by using an access token over OAuth 2.0
- Worklight project configuration
- External service configuration
- Use of the client-side API
- Exercise

Worklight authentication by using an access token over OAuth 2.0 - Overview

- The OAuth 2.0 authorization framework enables an application from independent software vendors (also called “third-party application”) to obtain limited access to an HTTP service.
- The implementation uses three roles of the OAuth protocol:
 - **Resource Server:** **third-party server**
The server that hosts the protected resources. It can accept, and respond to, protected resource requests by using access tokens.
 - **Client:** **app**
An application that requests protected resources.
 - **Authorization Server:** **Worklight Server**
The server that issues access tokens to the client after it has successfully authenticated the resource owner and obtained authorization.

Quick flow:

- The client requests a token from the authorization server, receives it, and with that token, accesses the protected resource on the resource server.

Worklight authentication by using an access token over OAuth 2.0 (1 of 8)

■ Overview of the Resource Server component

The **Resource Server** is an external server that hosts the available resource.

One use-case is that of services that are deployed on a cloud, such as **MbaaS**. But this flow is not restricted to that case and works with any third-party server.

The **Token lib** library is provided for use with the **Resource Server**.

The public key that is necessary to verify the token must be configured for this component.

Java™ and `node.js` libraries are provided for offline validation.



Worklight authentication by using an access token over OAuth 2.0 (2 of 8)

- Overview of the client component



The **External SDK** is used to access the resource server. It must be able to attach a header to the request.

The **WL SDK** exposes the following API for both hybrid and native code:

1. Request the access token from **Worklight Server**.
2. Get the last access token (local).
3. Analyze **Resource server** error response to obtain the required scope.



Worklight authentication by using an access token over OAuth 2.0 (3 of 8)

- Overview of the Worklight Server component



The **Validation Endpoint** is a REST API exposed under the path **/oauth/validation.s**

It can be used by **Resource Server** or by the reverse proxy for online validation.

The **Token Endpoint** is a REST API exposed under the path **/oauth/token**.



Worklight Server uses the authentication infrastructure to issue an access token for the requested scope (WL security test).



Worklight authentication by using an access token over OAuth 2.0 (4 of 8)

■ Component overview of token format

```
{
  version: 1.0 (of token)
  scope: <the security test that the token authenticated>
  expiration: <time in msec since epoch>
  data: {
    user_id: <authenticated user, for example, shachor@il.ibm.com>,
    device_id: <device id as known by worklight server>,
    application_id: <identity of the app>
  }
}
```

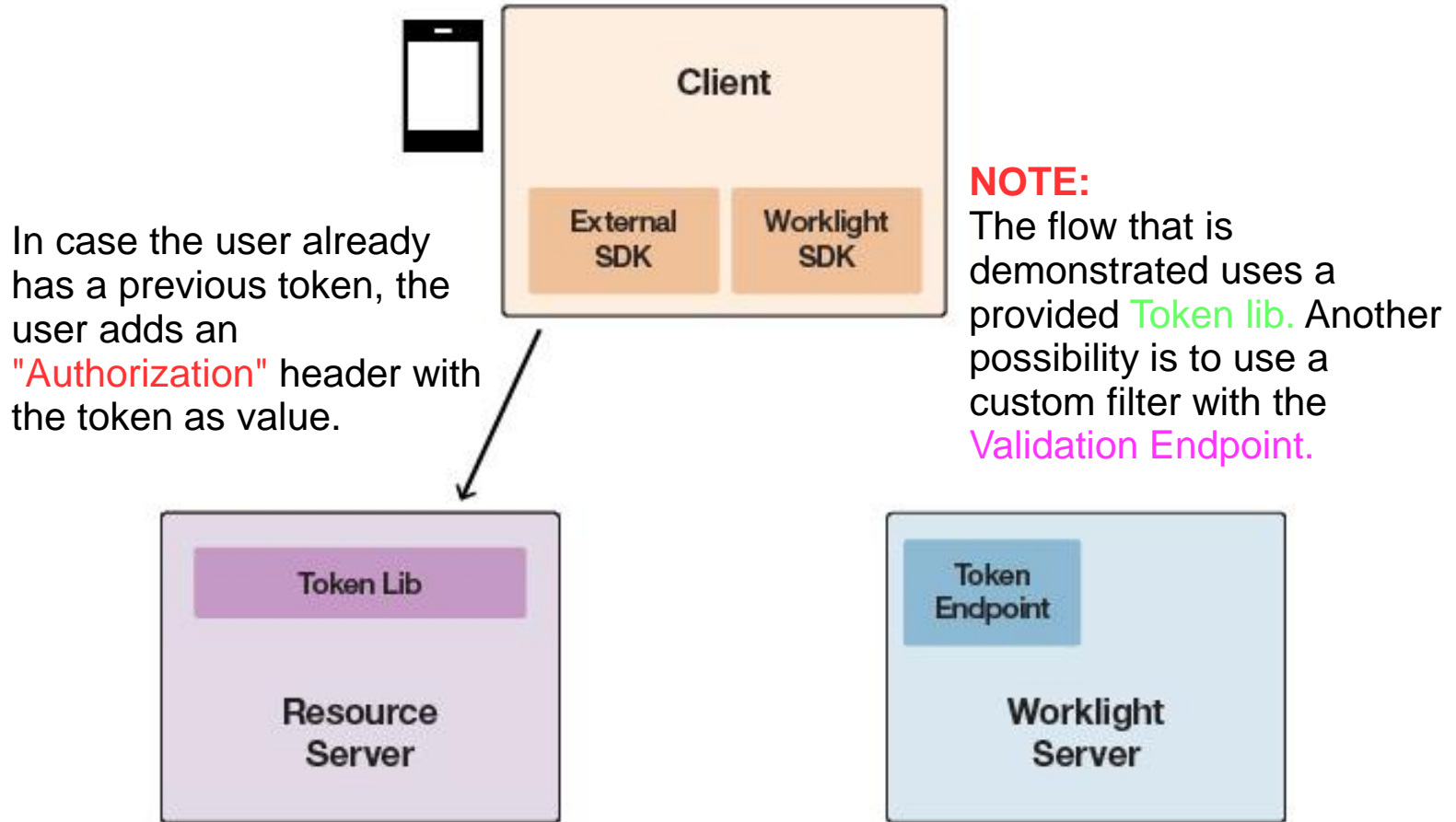
The token is signed by the Worklight Server instance.

Clarification for the `data` field:

- The `user_id` field is added only if the security test has a user realm.
- The `device_id` field is added only if the security test has a device realm.
- The `application_id` field is **always** added.

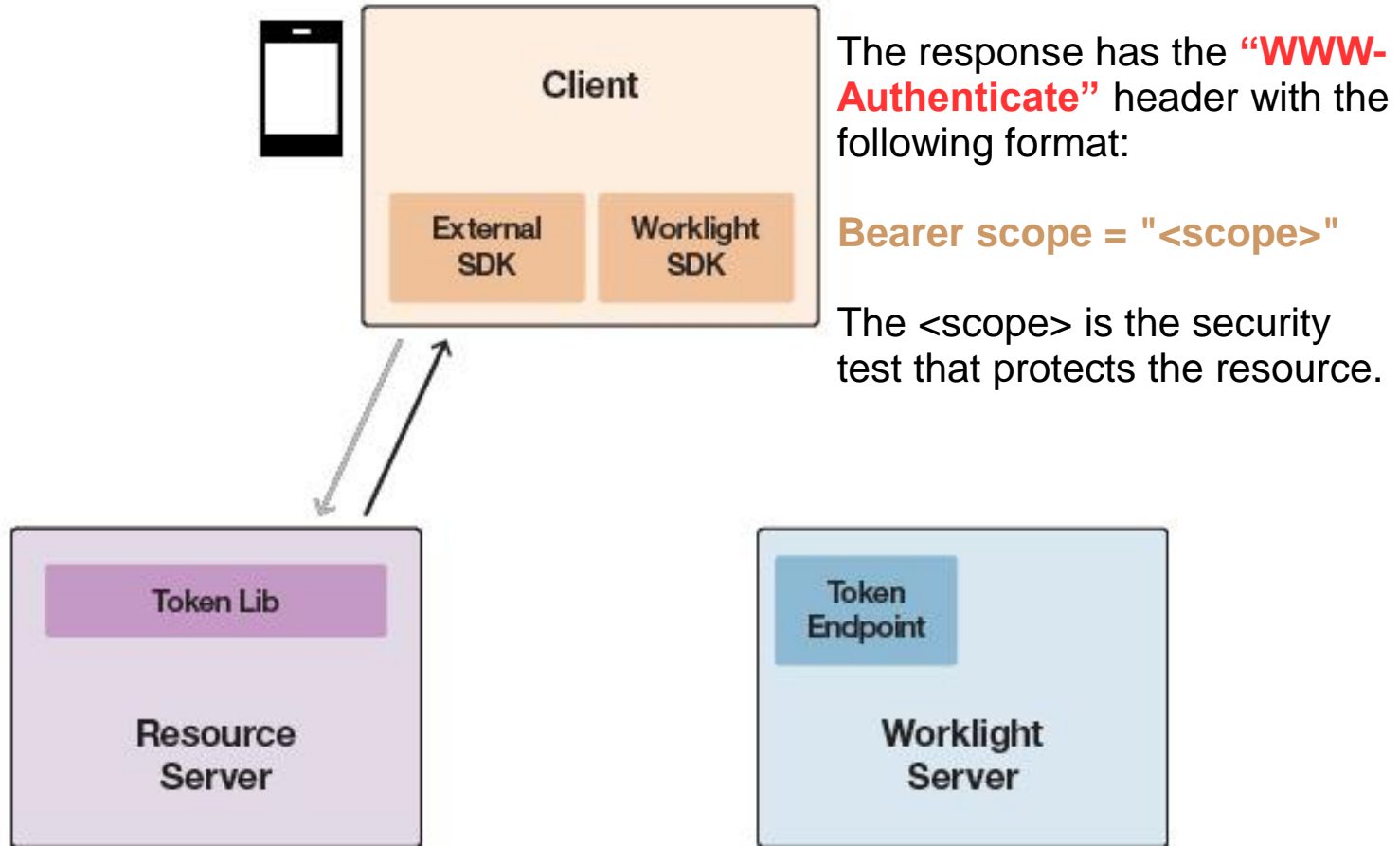
Worklight authentication by using an access token over OAuth 2.0 (5 of 8)

- Flow Step 1 – The application developer attempts to access a protected resource on a remote server.



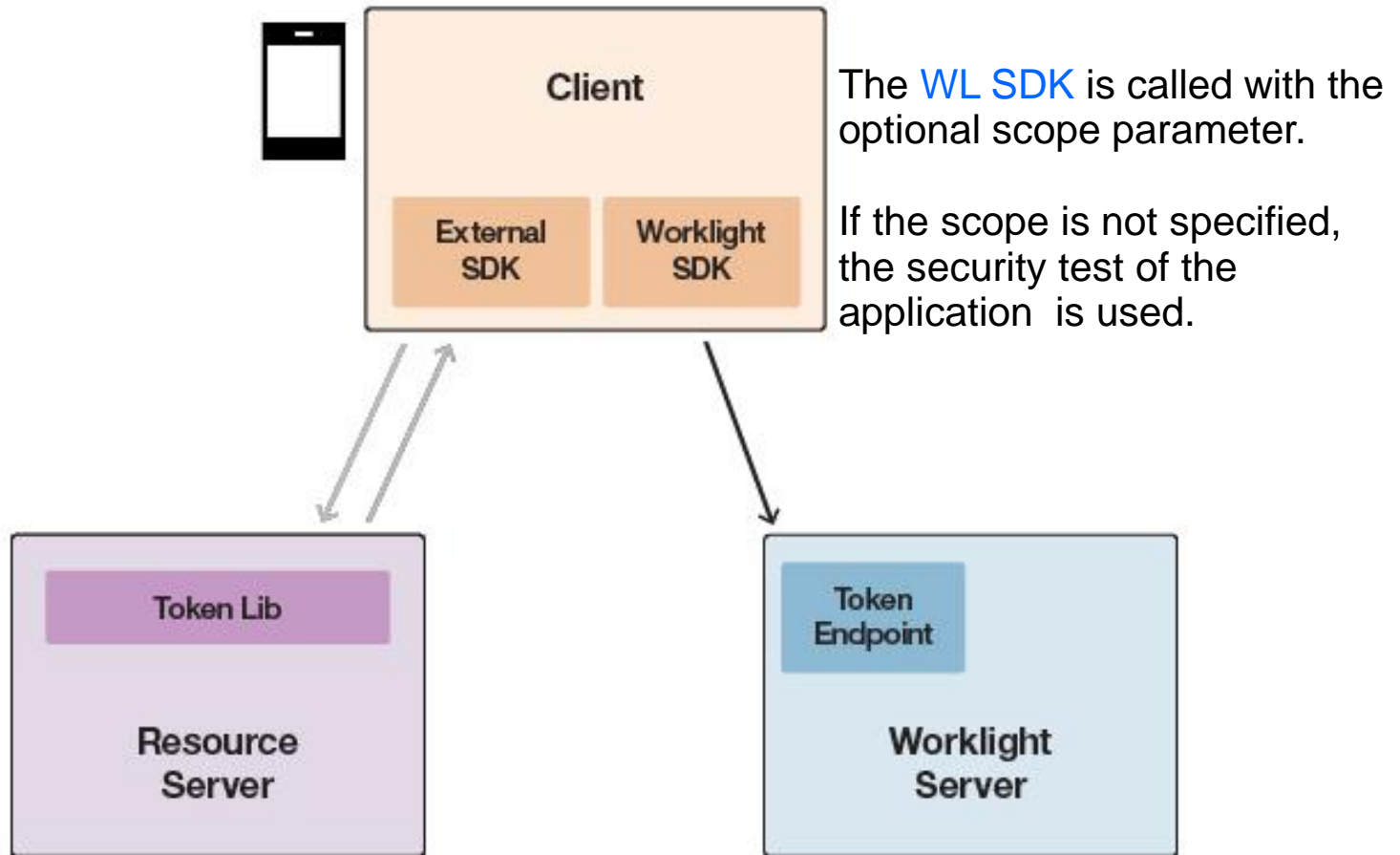
Worklight authentication by using an access token over OAuth 2.0 (6 of 8)

- Flow Step 2 - If the request does not have a token, or the token is not valid, Resource Server returns 401/403 via the supplied lib.



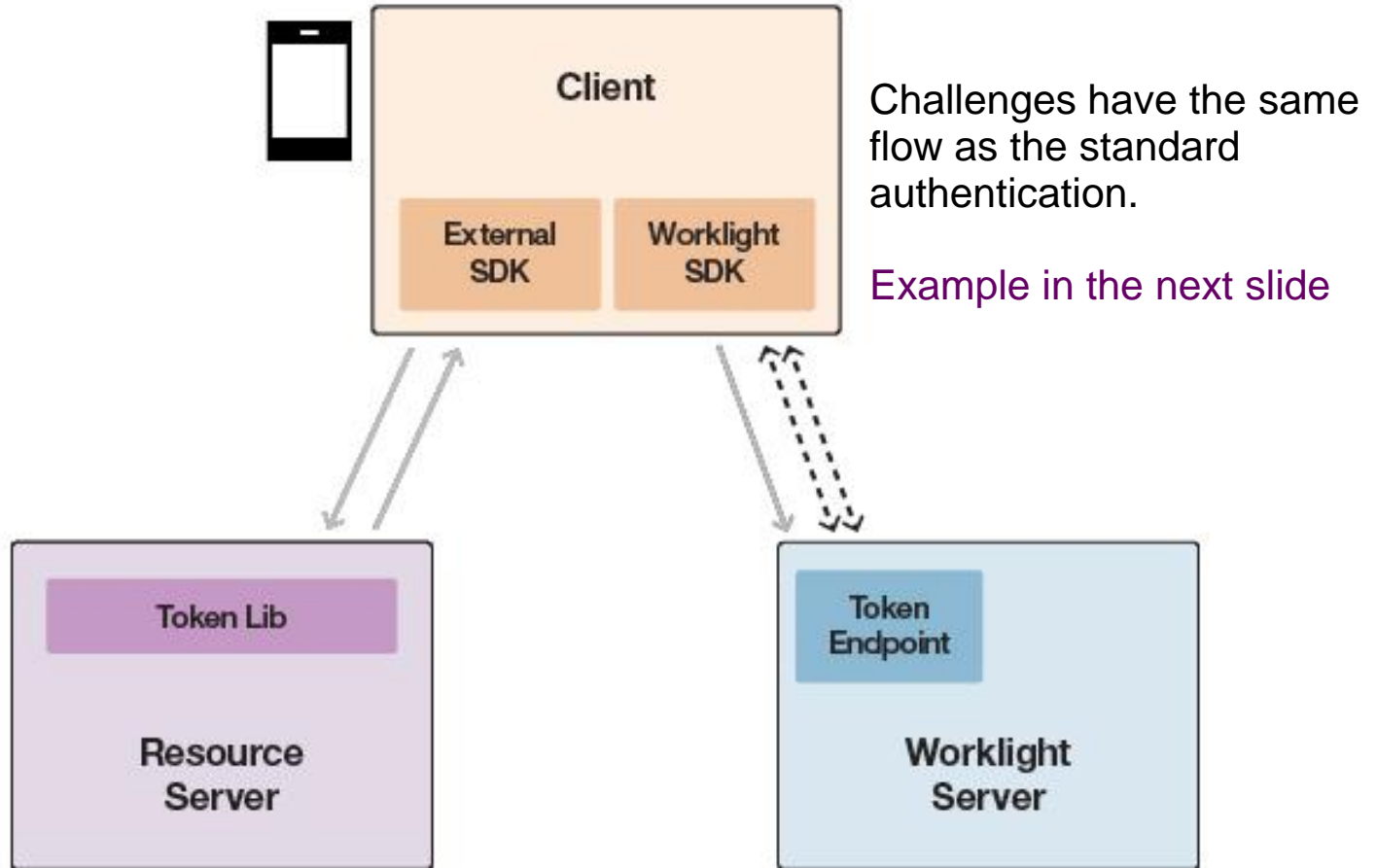
Worklight authentication by using an access token over OAuth 2.0 (7 of 8)

- Flow Step 3 - The application developer parses the response and obtains the access token by using Worklight SDK.

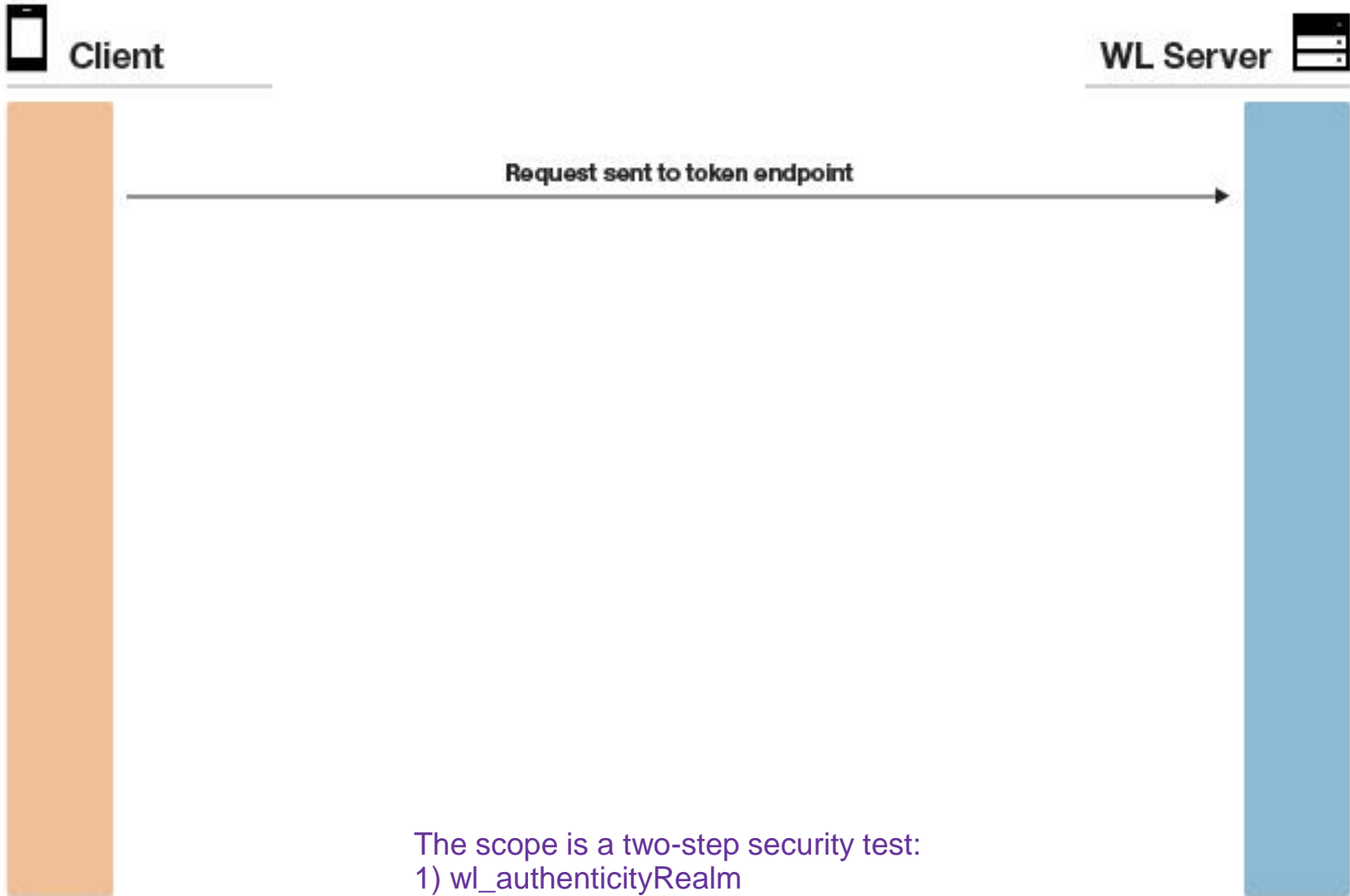


Worklight authentication by using an access token over OAuth 2.0 (8 of 8)

- Flow Step 4 - If some realms are not authenticated yet, Worklight Server sends challenges back to the client and the client responds.

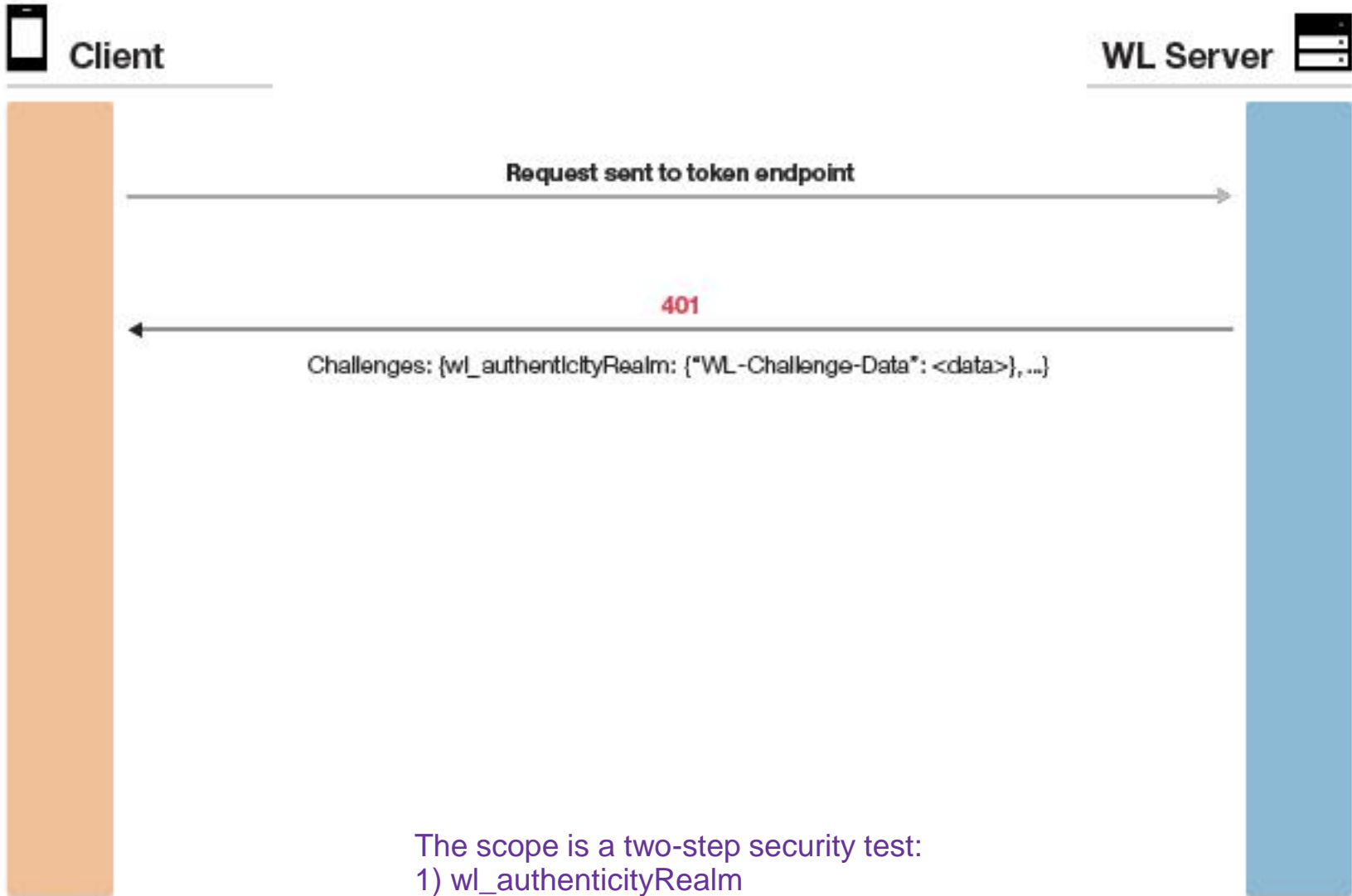


Worklight authentication by using an access token over OAuth 2.0 - Challenge flow example (1 of 6)



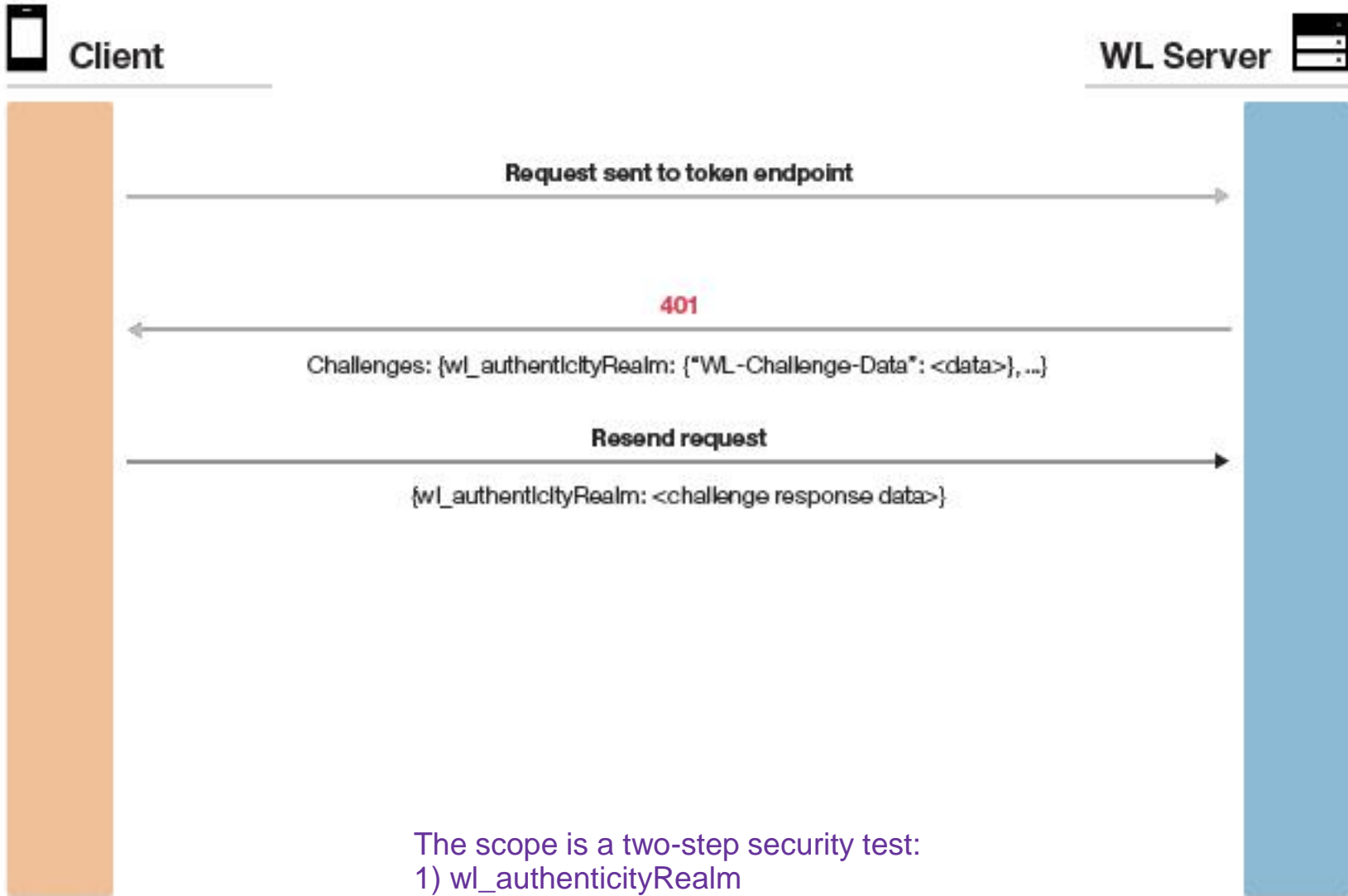
The scope is a two-step security test:
 1) wl_authenticityRealm
 2) wl_deviceAutoProvisioningRealm

Worklight authentication by using an access token over OAuth 2.0 - Challenge flow example (2 of 6)



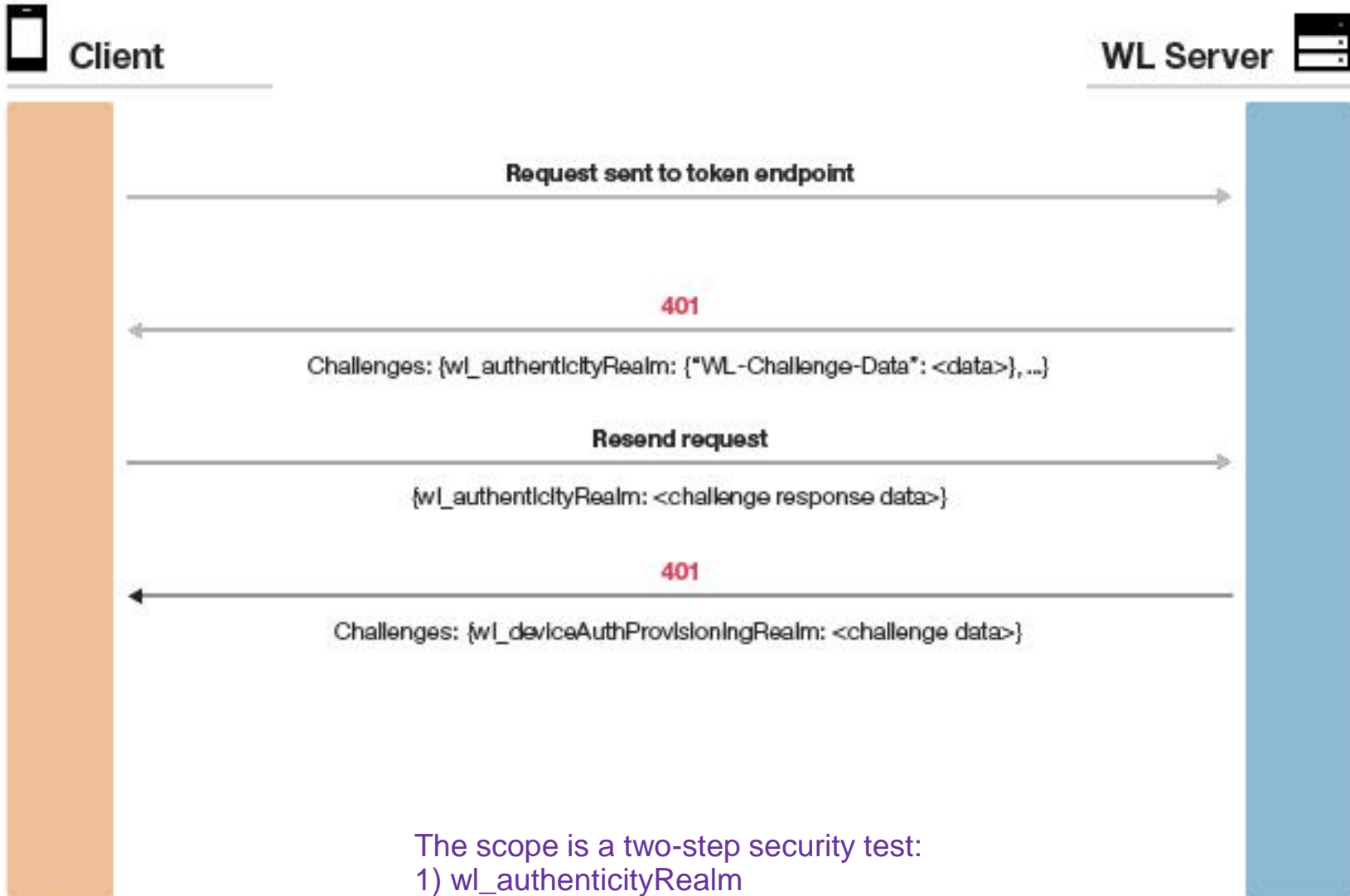
The scope is a two-step security test:
 1) wl_authenticityRealm
 2) wl_deviceAutoProvisioningRealm

Worklight authentication by using an access token over OAuth 2.0 - challenge flow example (3 of 6)



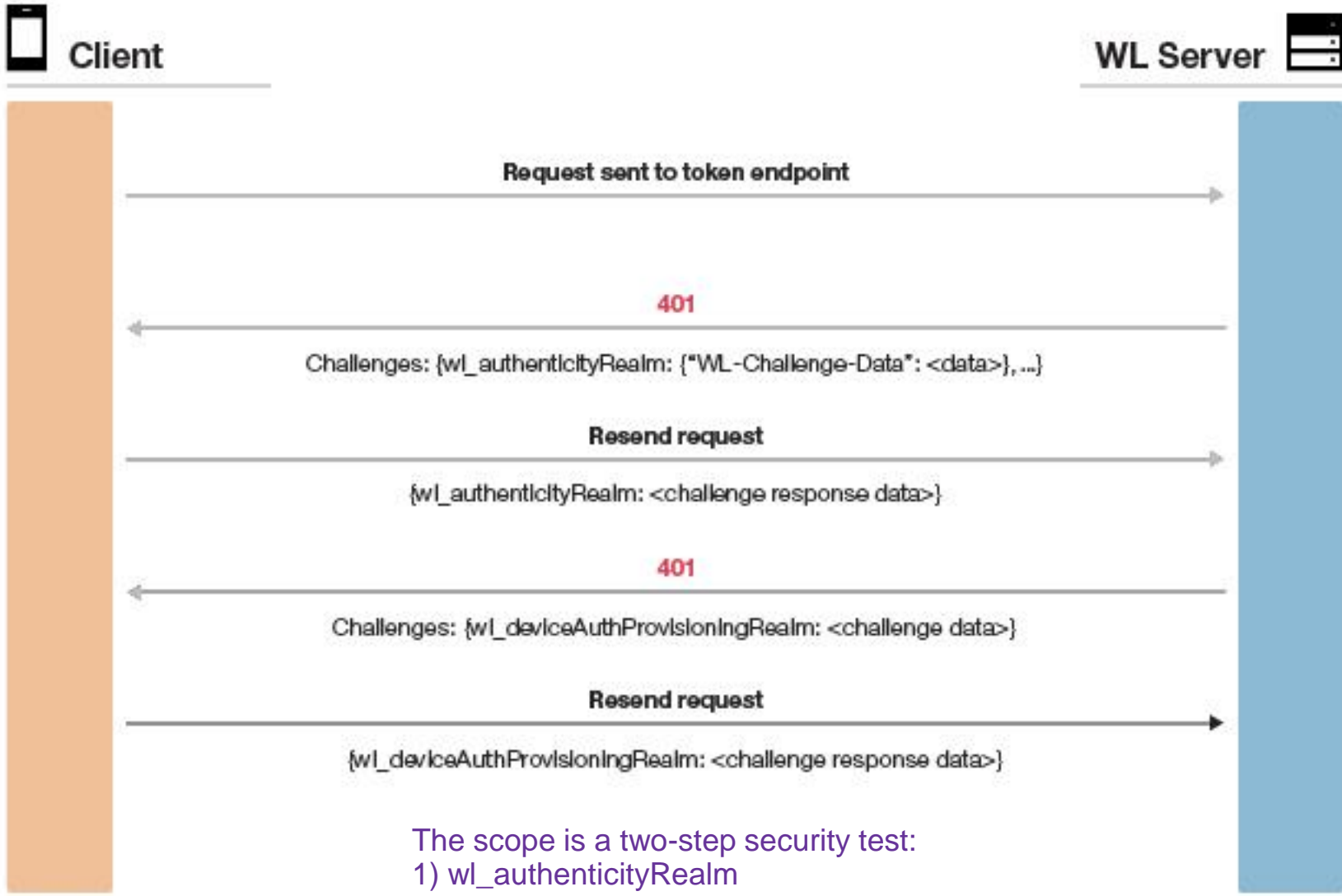
The scope is a two-step security test:
 1) wl_authenticityRealm
 2) wl_deviceAutoProvisioningRealm

Worklight authentication by using an access token over OAuth 2.0 - Challenge flow example (4 of 6)



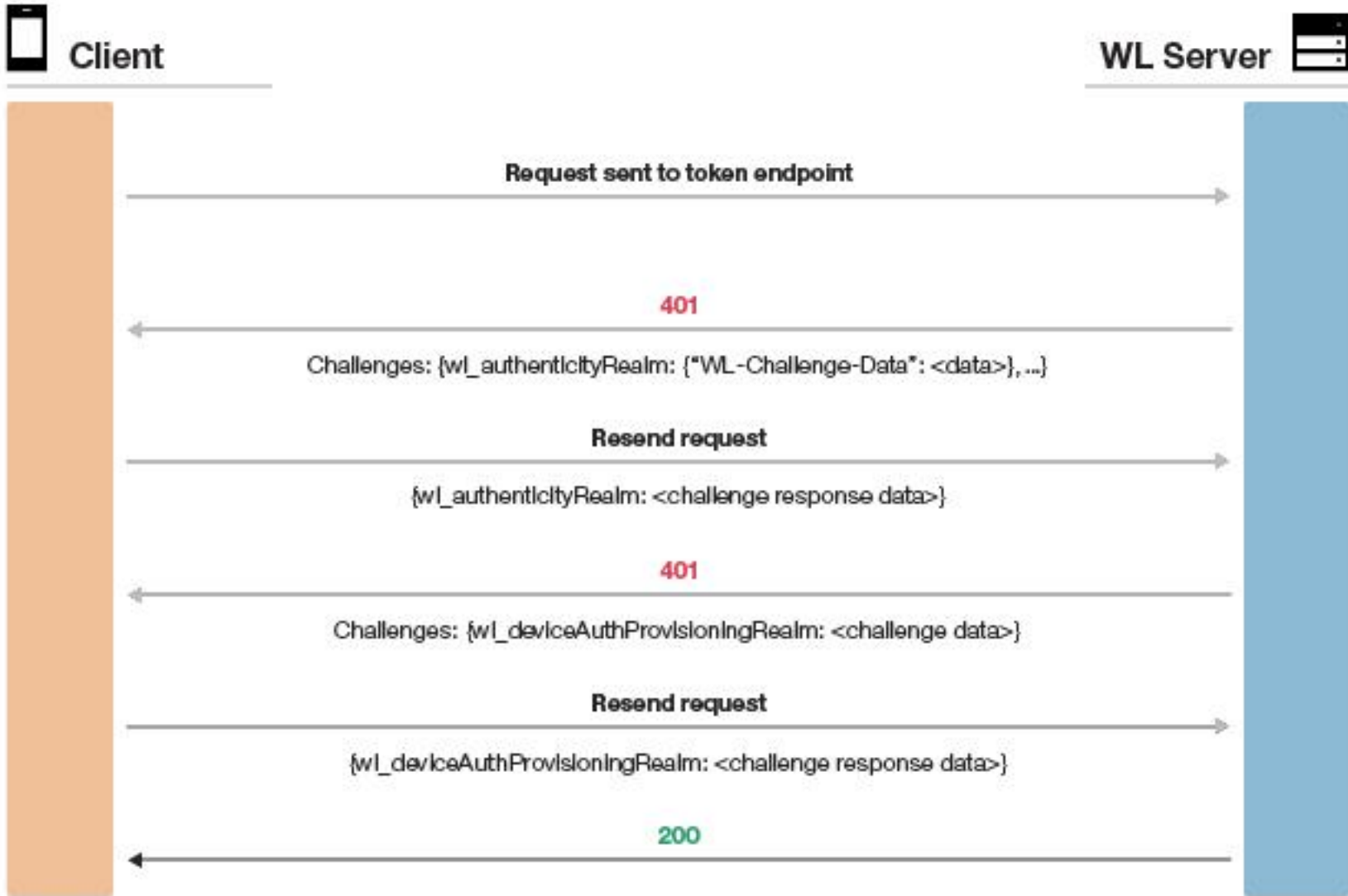
The scope is a two-step security test:
 1) wl_authenticityRealm
 2) wl_deviceAutoProvisioningRealm

Worklight authentication by using an access token over OAuth 2.0 - Challenge flow example (5 of 6)



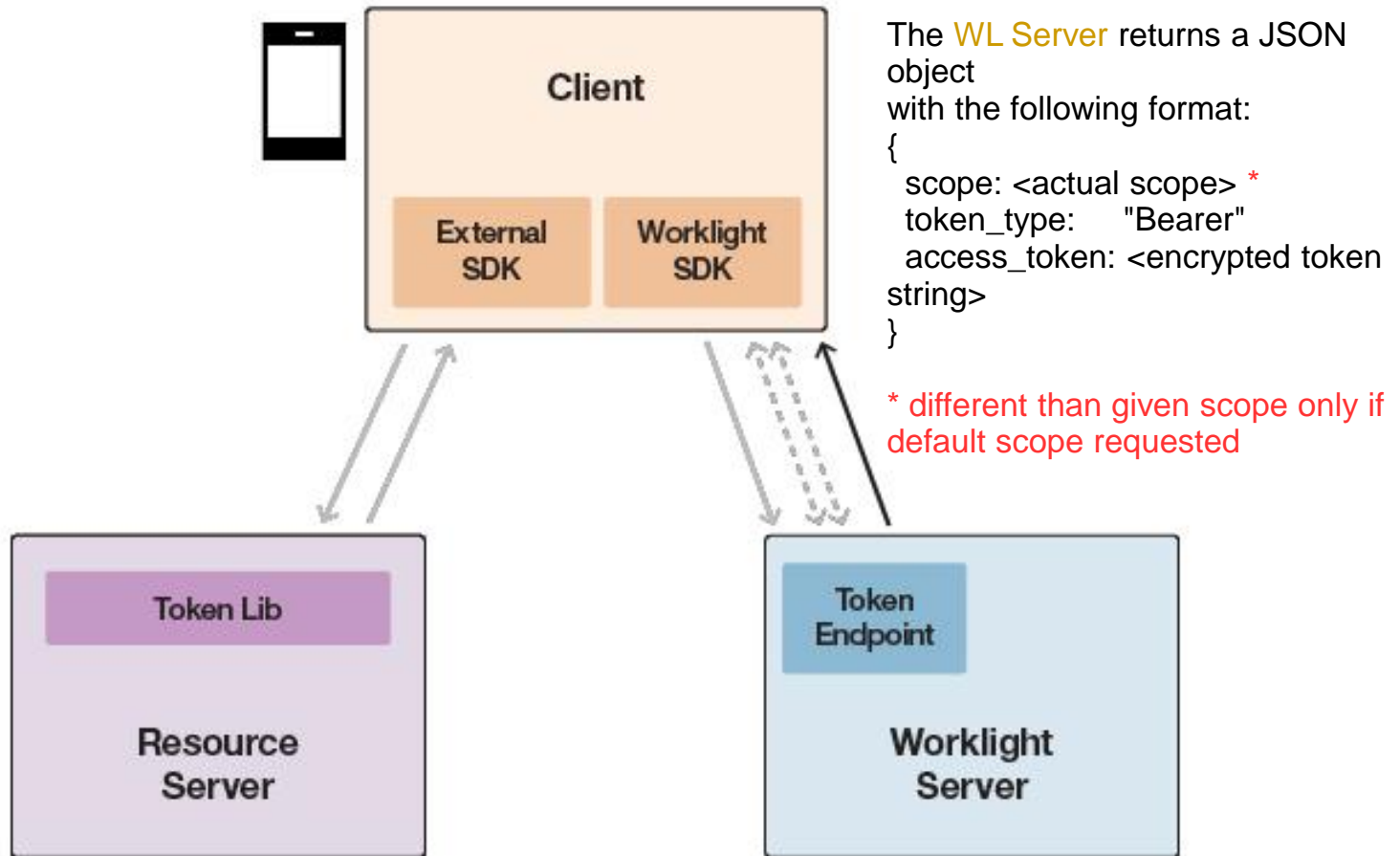
The scope is a two-step security test:
 1) wl_authenticityRealm
 2) wl_deviceAutoProvisioningRealm

Worklight authentication by using an access token over OAuth 2.0 - Challenge flow example (6 of 6)



Worklight authentication by using an access token over OAuth 2.0 (1 of 2)

- Flow Step 5 - After all relevant realms are authenticated, Worklight Server returns an access token to the client.



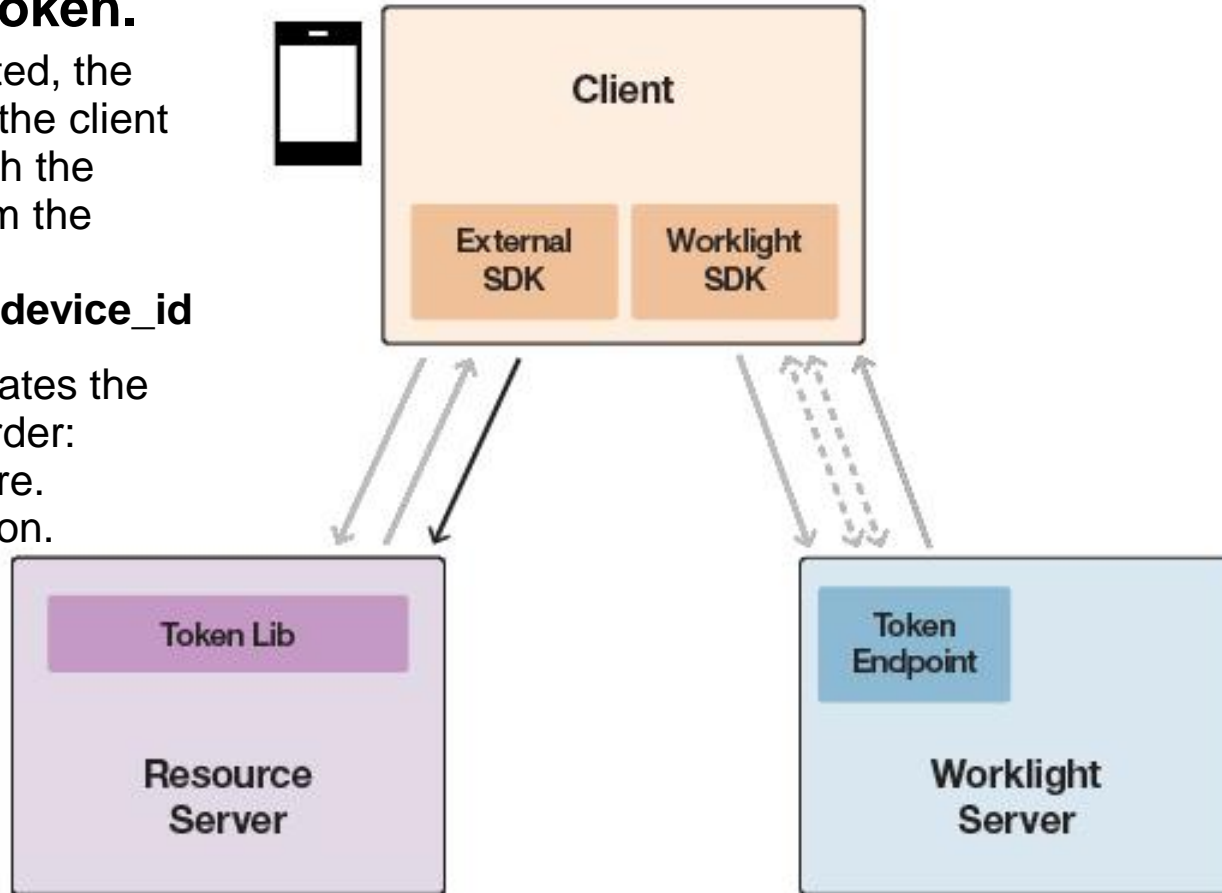
Worklight authentication by using an access token over OAuth 2.0 (2 of 2)

- **Flow Step 6 – The application developer now resends the original request, adding an "Authorization" header with the retrieved token.**

After it has validated, the **token lib** updates the client Context object with the following data from the access token:
app_id, user_id, device_id

The **token lib** validates the token in this order:

1. Check signature.
2. Check expiration.
3. Check scope.



Agenda

- Feature overview
- Worklight authentication by using an access token over OAuth 2.0
- **Worklight project configuration**
- External service configuration
- Use of the client-side API
- Exercise

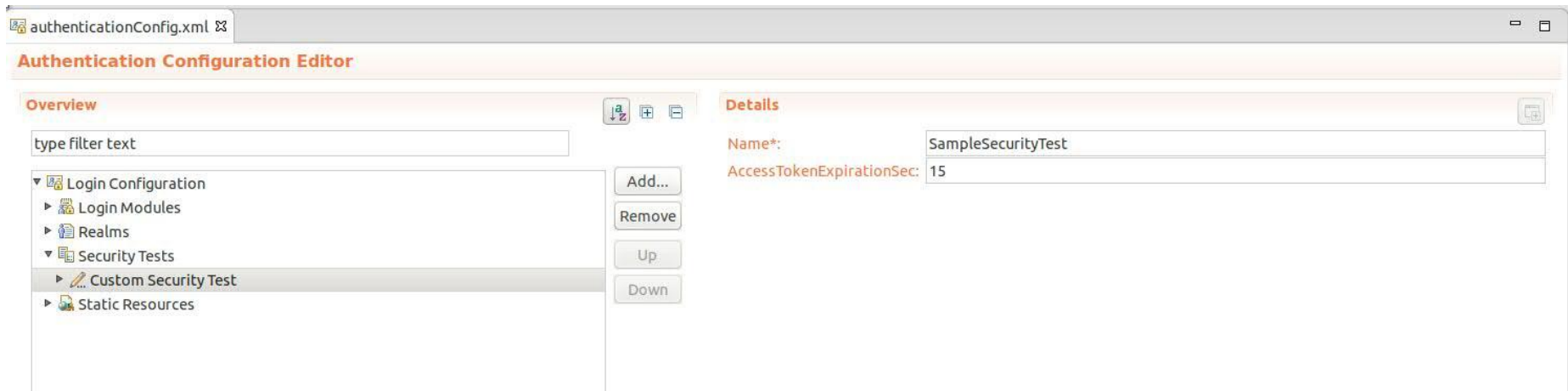
Worklight project configuration - Configuring the scope for the access token (1 of 2)

- The scope of an access token must be a predefined security test in a Worklight project. You configure it in this file:
`<WL_Project>/server/conf/authenticationConfig.xml`.
- The default lifetime for each token is 60 seconds. You can override this timeout by adding the **AccessTokenExpirationSec** attribute to the security test.
- For example, if you want to configure a security test, called “SampleSecurityTest”, with a lifetime of 15 seconds, edit the `authenticationConfig.xml` as follows.
 - From the source view:

```
<securityTests>
  <customSecurityTest name="SampleSecurityTest" AccessTokenExpirationSec="15">
    <test realm="SampleRealm" isInternalUserID="true"/>
  </customSecurityTest>
</securityTests>
```

Worklight project configuration - Configuring the scope for the access token (2 of 2)

- From the design view:



The screenshot shows the 'Authentication Configuration Editor' window. The title bar indicates the file is 'authenticationConfig.xml'. The interface is divided into two main sections: 'Overview' and 'Details'.

Overview Panel: Contains a search filter 'type filter text'. Below it is a tree view with the following structure:

- Login Configuration
 - Login Modules
 - Realms
 - Security Tests
 - Custom Security Test (selected)
 - Static Resources

Buttons for 'Add...', 'Remove', 'Up', and 'Down' are located to the right of the tree view.

Details Panel: Shows configuration for the selected 'Custom Security Test'.

- Name*: SampleSecurityTest
- AccessTokenExpirationSec: 15

Worklight project configuration - keystore

- To use this feature, preferably use or create your own keystore and configure the Worklight Server to use it.
- For an example in an unrelated context, see [Configuring device auto provisioning](#) in the product documentation.
- Using the default Worklight Server keystore is **NOT** secure!

Agenda

- Feature overview
- Worklight authentication by using an access token over OAuth 2.0
- Worklight project configuration
- External service configuration
- Use of the client-side API
- Exercise

External service configuration

- For your external service to accept the access token, you must add a validation library to your service, which must be able to validate the token with either online or offline validation.
- Two libraries are provided for this purpose:
 - `worklight-access-token-validator.jar` – Java lib
 - `worklight-access-token-validator.tgz` – node.js module
- For Worklight Server installation – You can find the libraries in:
`<installation dir>/WorklightServer/external-server-libraries`
- For the Worklight Studio – When you create a new project, you can find the libraries in:
`<project dir>/externalServerLibraries`

External service configuration - Using Java

- The purpose of this module is to enable offline validation of access tokens that are generated by Worklight Server for Java web applications.
- Validation of access tokens that are generated by Worklight Server is also possible for node.js servers.
- To use the Java library, you need two files:
 - `Certificate` – For the associated sample, the certificate that has been exported from the Worklight Server Keystore. You can use the Java keytool. In production, preferably use your own keystore as explained in slide [Slide 27 - Worklight Project Configuration - Keystore](#).
 - `Worklight-access-token-validator.jar`

External service configuration - Using Java: servlet filter (1 of 3)

- Add the `worklight-access-token-validator.jar` file to the class path of your web application and use the filter class `com.worklight.security.WLAccessTokenValidationFilter` as shown below.
- Example of a filter definition:

Filter name = FilterName:

Choose an name you want for the filter.

URL = /some/protected/url

The prefix for all the resources that you wish to protect.

Scope [Optional] = securityTestName

The name of the security test, as defined in the `authenticationConfig.xml` file, which is needed to authenticate against to gain access to the protected resources. If the scope is not specified, the filter accepts any valid token that is provided by Worklight Server.

CertificatePath = certificateLib/WorklightServerCertiificate.cert.

The path to the certificate of Worklight Server, relative to the WEB-INF folder.

External service configuration - using Java: servlet filter (2 of 3)

- In addition to the previous parameters, write the following code for the `web.xml` file of your external server:

```
<web-app ...>
...
<filter>
  <filter-name>FilterName</filter-name>
  <filter-class>com.worklight.security.WLAccessTokenValidationFilter</filter-class>
  <init-param>
    <param-name>worklightCertificateFile</param-name>
    <param-value>certificateLib/WorklightServerCertificate.cert</param-value>
  </init-param>
  <init-param>
    <param-name>scope</param-name>
    <param-value>securityTestName</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>FilterName</filter-name>
  <url-pattern>/some/protected/url</url-pattern>
</filter-mapping>
...
</web-app>
```


External service configuration - Using Java: servlet filter (3 of 3)

- After successful validation, the filter updates the `ClientContext` object that the service can use to access user, application, or device identities that are contained in the access token.
- Example of `ClientContext` usage:

```
ClientContext context = ClientContext.getInstance();  
String appId = context.getApplication();  
String userId = context.getUser();  
String deviceId = context.getDevice();
```

Agenda

- Feature overview
- Worklight authentication by using an access token over OAuth 2.0
- Worklight project configuration
- External service configuration
- Use of the client-side API
- Exercise

Use of the client-side API: which methods are allowed (1 of 4)

- The Worklight `WL.Client` API provides built-in support for using Worklight access tokens for the following platforms:
 - Hybrid – JavaScript™
 - Android
 - iOS
- The methods included in this API provide the following services:
 - Obtaining and caching a token for a specified scope
 - Getting the last obtained access token
 - Getting the required scope from the external service response
- See the following slides. For more information, see also [Using SSO between IBM Worklight Foundation and external services](#) in the product documentation.

Use of the client-side API: which methods are allowed (2 of 4)

- **Obtaining and caching a token for a specified scope**

In JavaScript, the method is called **obtainAccessToken**.

- The `WL.Client` instance requests a new token from the Worklight Server instance. To obtain the token, the client must be authenticated in all realms of the requested scope (which is represented by a security test in the Worklight Server `AuthenticationConfig.xml` configuration file). Thus, calling this method triggers an authentication sequence for all realms for which authentication is still required.
- This method is asynchronous in all platforms. It does not return a value but, instead, triggers a response handler.
 - Note: It is not necessary to parse the response from the server in the response handler. The token is automatically parsed and cached inside the `WL.Client` instance and can be retrieved by using the next method.

Use of the client-side API: which methods are allowed (3 of 4)

- **Getting the last obtained access token**

In JavaScript, this method is called **getLastAccessToken**.

- The `WL.Client` instance returns the last access token for a certain scope as a string. Alternatively, if no scope is specified, the last obtained token is returned. This capability is useful when an application is using one scope **only**.
- Add a header named "Authorization" and for the header content, add "Bearer ", followed by the token. For example, when issuing an Ajax request, you can write the following code:

```
var token = WL.Client.getLastAccessToken();
$.ajax({
  type : "GET",
  url : MY_URL,
  headers : {"Authorization" : "Bearer " + token}
})
```

Use of the client-side API: which methods are allowed (4 of 4)

- **Getting the required scope from the external service response**

In JavaScript, this method is called **getRequiredAccessTokenScope**.

When a request to the external service fails, the `WL.Client` instance can identify the cause of the failure.

- If the failure is related to access token issues, the client returns the name of the scope that is required to access the service. In this case, it is necessary to obtain a new token for the returned scope. For example, the token does not match the required scope, or the token has expired.
- If the error is not related to access token issues, the method returns `null`.

Use of the client-side API: JavaScript example (1 of 2)

- This JavaScript example shows how to use the client-side API for access to an external service.

```
function callProtectedRestAPI(retries) {  
  
    // You must be able to call this method  
    // recursively, because in some cases it is  
    // necessary to obtain a new token and try a  
    // second time.  
    if (retries == 0) {  
        return;  
    }  
    // Get the last obtained access token.  
    // On the first call, the token can be null.  
    var token = WL.Client.getLastAccessToken();  
    var headersObject = (token != null) ?  
    {"Authorization" : "Bearer " + token} : {};
```

(Continued on the following slide...)

Use of the client-side API: JavaScript example (2 of 2)

```

$.ajax({
    type : "GET",
    url : MY_EXTERNAL_SERVER_URL,
    headers : headersObject

}).done(function(response) {
    showResult(response);
}).fail(
    function(response) {
        // Need to extract this header from
        // the response to know the scope.
        var header = response.getResponseHeader("WWW- Authenticate");
        var scope = WL.Client.getRequiredAccessTokenScope(response.status,header);
        if (scope != null) {
            // The failure is related to the access token. Get a new one.
            WL.Client.obtainAccessToken(scope, getTokenSuccess,getTokenFailure);
        } else {
            showErrorResult("request failed");
        }
    }
);
function getTokenSuccess(response) {
    // Obtained a token. Try to access the external server one more time.
    callProtectedRestAPI(retries - 1);
};
function getTokenFailure(response) {
    showErrorResult(response);
};
}

```


Agenda

- Feature overview
- Worklight authentication by using an access token over OAuth 2.0
- Worklight project configuration
- External service configuration
- Use of the client-side API
- **Exercise**

Exercise (1 of 3)

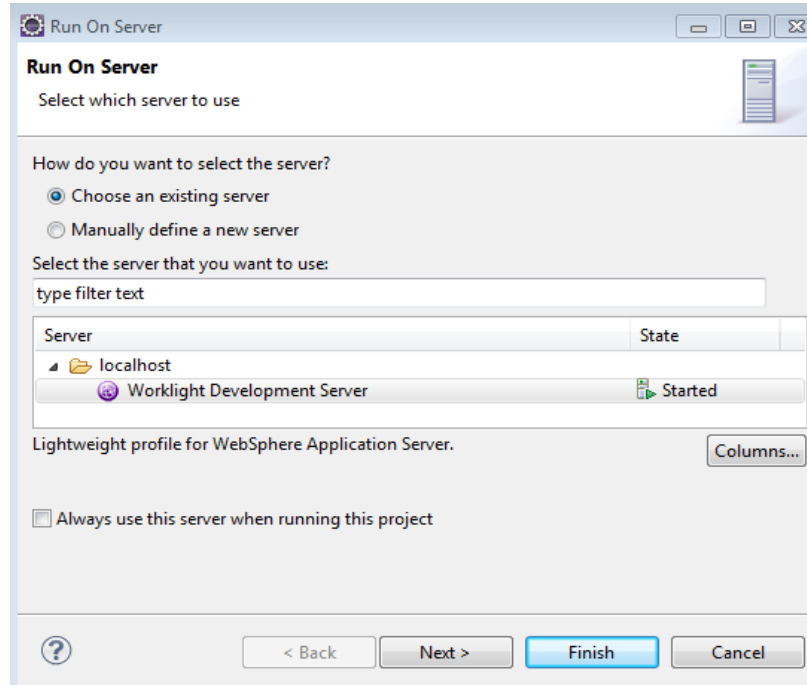
- See the project `WorklightAsAuthorizationServer.zip`. You can find the sample for this training module from the Getting Started page of the IBM® Worklight Foundation documentation website at <http://www.ibm.com/mobile-docs>.
- The `WorklightAsAuthorizationServer` project contains two items.
 - `REST-Caller` is a standard Worklight project.
 - Project `REST-Caller` contains the **REST-Server** folder. This folder contains the `REST-Server.zip` archive, which contains a Java web application.

Exercise (2 of 3)

1. In Eclipse, select **File > Import** and then in the wizard, select **Existing Projects into Workspace**.
2. Click the **Select archive file** option, navigate to your current workspace project `REST-Caller`, and import the `REST-Server.zip` file.
3. To deploy the REST-Caller Worklight App, select the **Run on Worklight Development Server** command.

Exercise (3 of 3)

4. To deploy the REST-Server Java web application, right-click the project and select **Run On Server**, then **Worklight Development Server**.



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> the sections entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

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 Developer Edition support community at:
 - <https://www.ibm.com/developerworks/mobile/worklight/connect.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

