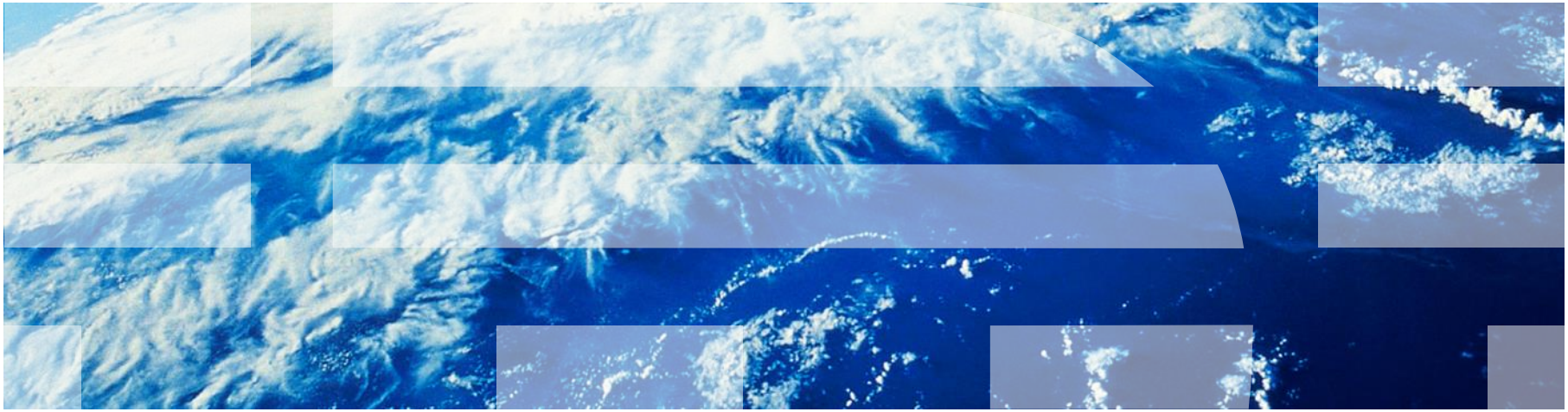
IBM Worklight Foundation V6.2.0 入門

Worklight Server を使用した外部リソースの認証



2014 年 8 月 27 日

商標

- IBM、IBM ロゴ、ibm.com および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。
- 本記述は公式な Joyent Node.js のオープン・ソースまたは商業プロジェクトと正式に関連はなく、承認もされていません。

IBM® について

- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- 機能の概要
- OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証
- Worklight プロジェクトの構成
- 外部サービスの構成
- クライアント・サイド API の使用
- 演習

機能の概要 (1/2)

- Worklight Server を使用して外部リソースを認証することで、IBM Worklight Foundation と外部サービスの間でシングル・サインオン (SSO) を使用できます。
- この機能では、Worklight Security Framework でこれらのサービスを保護します。
- Worklight Server は許可サーバーとして機能し、外部サービスが検証できるアクセス・トークンを発行します。
- クライアント・アプリケーションは、トークン・エンドポイントを介して Worklight に対してアクセス・トークンを要求し、アクセス・トークンを外部サービスに送信します。

機能の概要 (2/2)

- アクセス・トークンのスコープは、Worklight プロジェクト内で定義されているセキュリティー・テストです。
- 各スコープには、トークンの存続期間を決定するタイムアウト・プロパティーがあります。例えば、トークンのタイムアウトが 30 秒に構成されている場合、Worklight Server がトークンを発行すると、そのトークンは 30 秒間有効になります。この時間が経過すると、トークンは外部サービスによって拒否され、新しいトークンを要求および発行する必要があります。

アジェンダ

- 機能の概要
- OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証
- Worklight プロジェクトの構成
- 外部サービスの構成
- クライアント・サイド API の使用
- 演習

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - 概要

- OAuth 2.0 許可フレームワークにより、独立系ソフトウェア・ベンダー (「サード・パーティー・アプリケーション」とも呼ばれます) が HTTP サービスに限定的にアクセスできるようにすることが可能です。
- この実装では、OAuth プロトコルの以下の 3 つのロールを使用しています。
  - **リソース・サーバー: サード・パーティー・サーバー**

    保護リソースをホストするサーバー。アクセス・トークンを使用して保護リソース要求を受け入れたり、応答したりできる。
  - **クライアント: アプリケーション**

    保護リソース要求を行うアプリケーション。
  - **許可サーバー: Worklight Server**

    正常にリソース所有者を認証して許可を取得した後にアクセス・トークンをクライアントに発行するサーバー。

クイック・フロー:

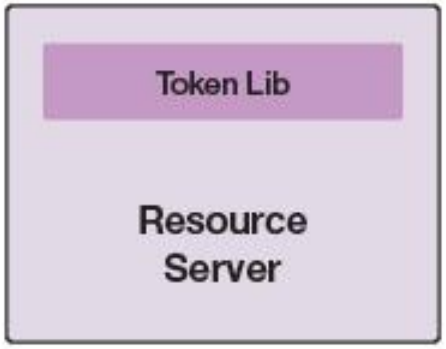
- クライアントは、許可サーバーに対してトークンを要求し、受け取ったトークンを使用して、リソース・サーバー上の保護リソースにアクセスします。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (1/8)

- リソース・サーバーのコンポーネントの概要

リソース・サーバーは、利用可能なリソースをホストしている外部サーバーです。

MbaaS などのクラウドにデプロイされたサービスが 1 つのユース・ケースとして考えられます。しかし、このフローはそのケースに限定されるわけではなく、いずれのサード・パーティー・サーバーでも機能します。



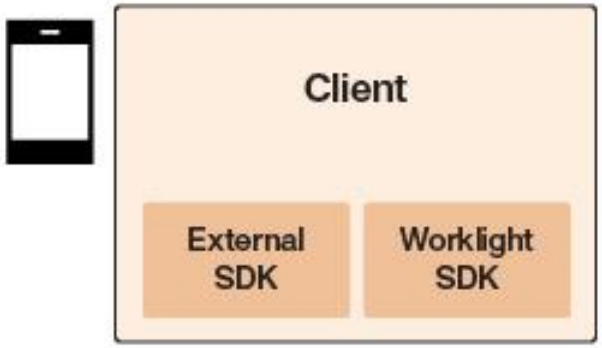
トークン lib ライブラリーは、ソース・サーバーで使用するために提供されます。

トークンを検証するために必要な公開鍵をこのコンポーネント用に構成する必要があります。

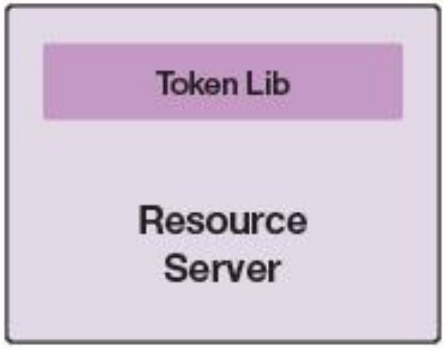
オフライン検証用に、Java™ ライブラリーおよび `node.js` ライブラリーが提供されています。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (2/8)

- **クライアント・コンポーネントの概要**



外部 SDK を使用して、リソース・サーバーにアクセスする。ヘッダーを要求に付加できる必要があります。

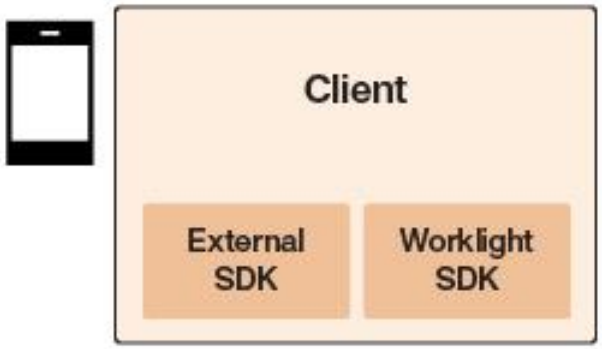


WL SDK は、ハイブリッド・コードとネイティブ・コードの両方用に以下の API を公開します。

1. Worklight Server からアクセス・トークンを要求する。
2. 最後のアクセス・トークン (ローカル) を取得する。
3. リソース・サーバーのエラー応答を分析して、必要なスコープを取得する。

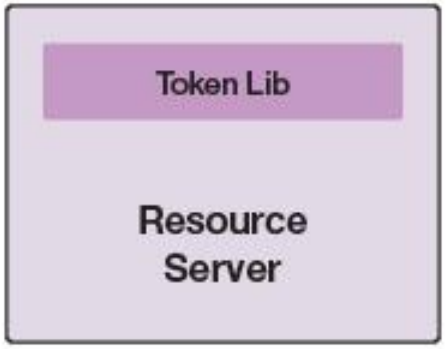
OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (3/8)

- **Worklight Server コンポーネントの概要**

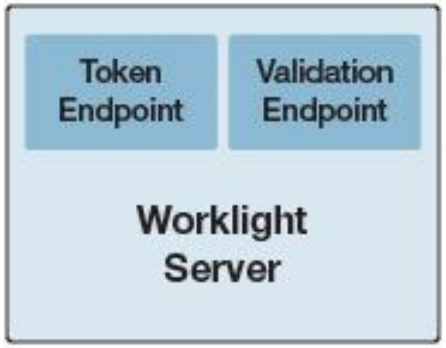


検証エンドポイントは、 以下のパスで公開されている REST API です。
/oauth/validation.s

これは、リソース・サーバーまたはリバース・プロキシーによってオンライン検証のために使用できます。
トークン・エンドポイントは、以下のパスで公開されている REST API です。
/oauth/token



Worklight Server は認証インフラストラクチャーを使用して、要求されたスコープのアクセス・トークンを発行します (WL セキュリティー・テスト)。



OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (4/8)

- トークン・フォーマットのコンポーネントの概要

```
{
    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>
    }
}
```

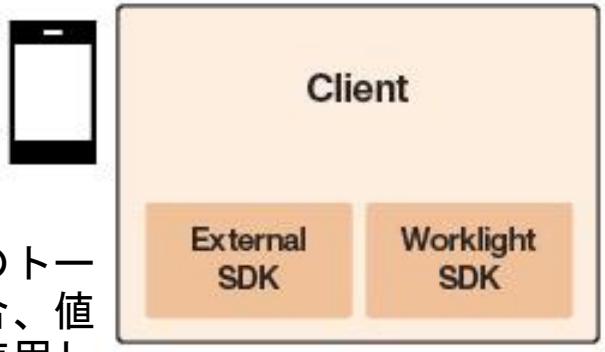
トークンは、Worklight Server インスタンスによって署名されます。

data フィールドの説明:

- user_id フィールドは、セキュリティー・テストにユーザー・レルムがある場合にのみ追加されます。
- device_id フィールドは、セキュリティー・テストにデバイス・レルムがある場合にのみ追加されます。
- application_id フィールドは常に追加されます。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (5/8)

- フロー・ステップ 1 – アプリケーション開発者が、リモート・サーバー上の保護リソースにアクセスしようとします。

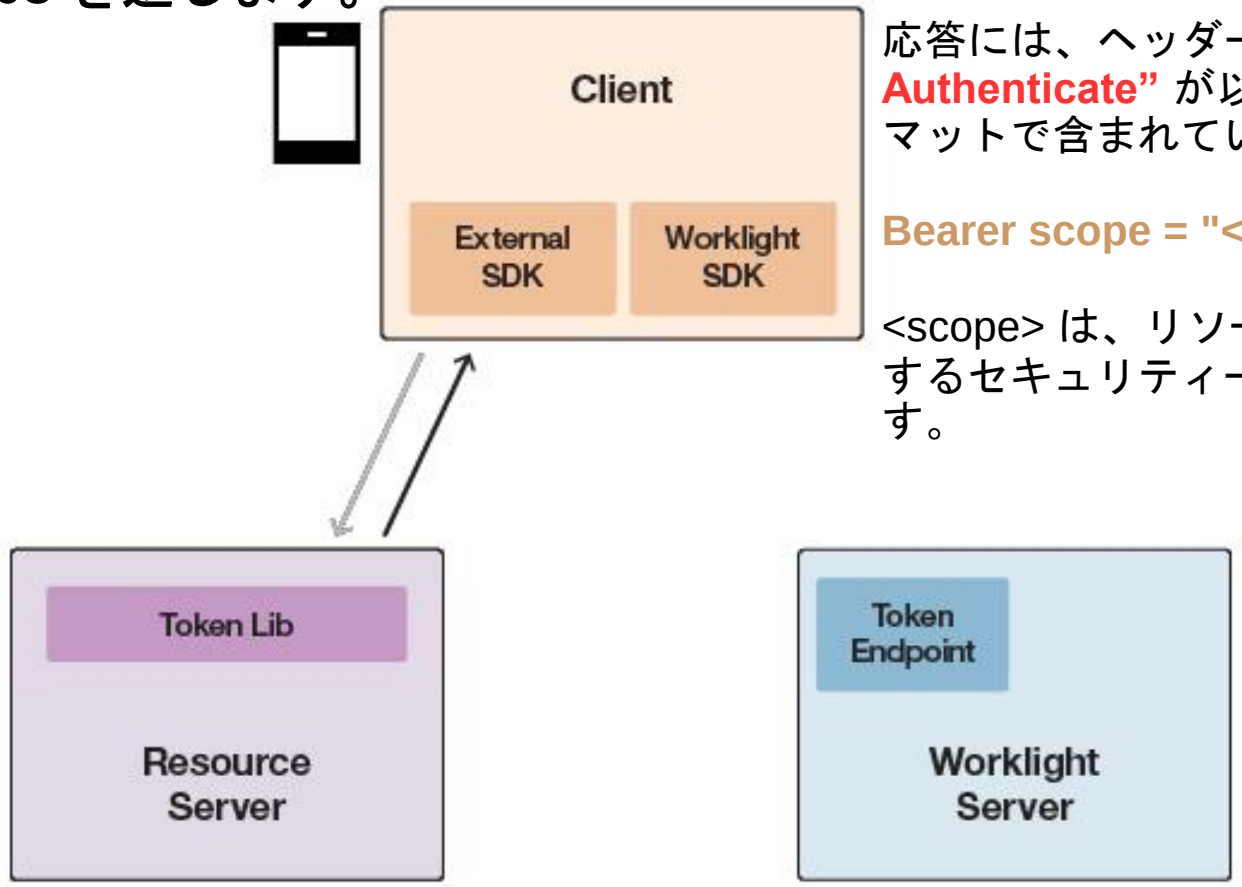


ユーザーは、既に以前のトークンを所有している場合、値としてそのトークンを使用して "Authorization" ヘッダーを追加します。

注:
デモのフローでは、提供されているトークン lib を使用しています。検証エンドポイントでカスタム・フィルターを使用することもできます。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (6/8)

- **フロー・ステップ 2 - 要求にトークンが含まれていない場合、またはトークンが無効な場合、リソース・サーバーは、提供されている lib を介して 401/403 を返します。**



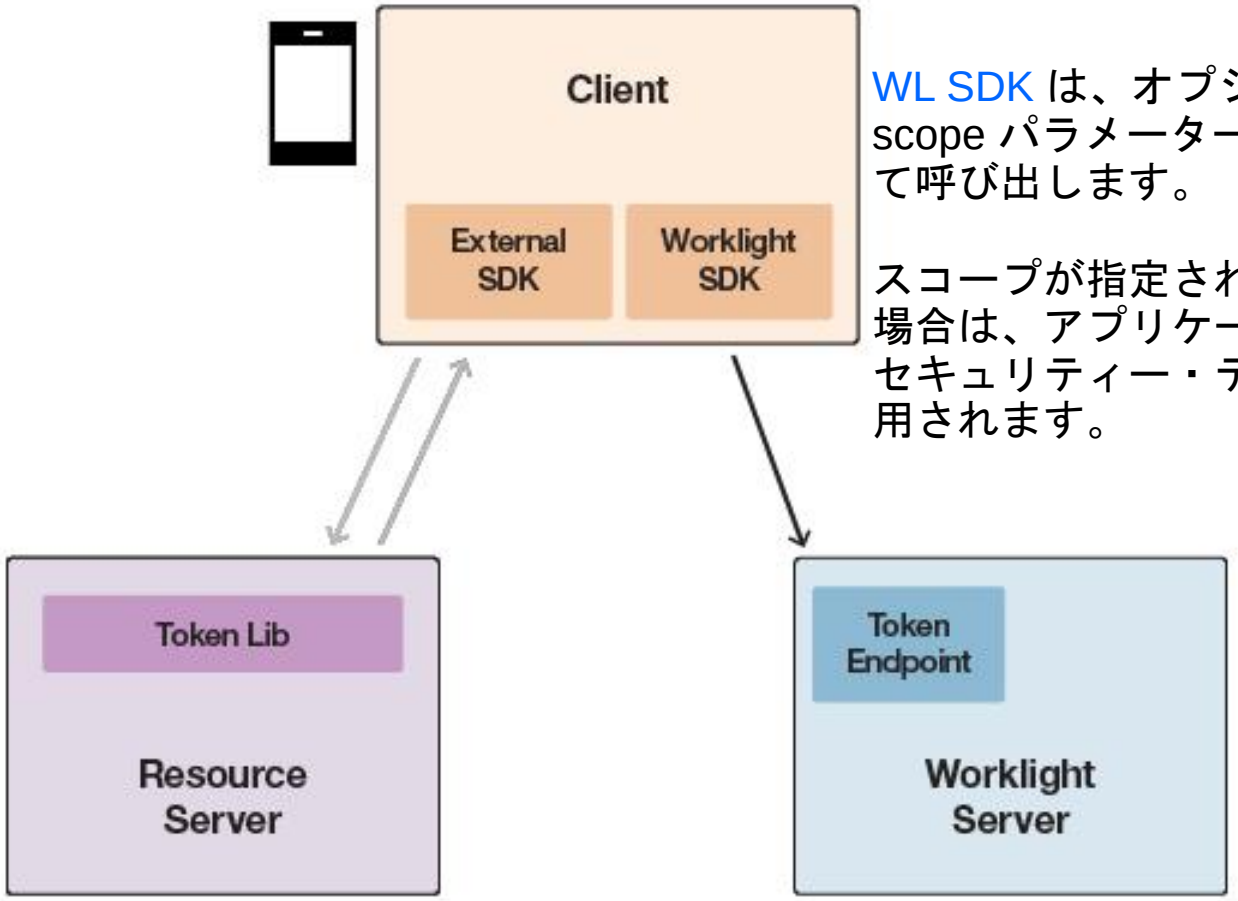
応答には、ヘッダー **“WWW-Authenticate”** が以下のフォーマットで含まれています。

Bearer scope = "<scope>"

<scope> は、リソースを保護するセキュリティー・テストです。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (7/8)

- フロー・ステップ 3 - アプリケーション開発者は、Worklight SDK を使用して、応答を解析し、アクセス・トークンを取得します。

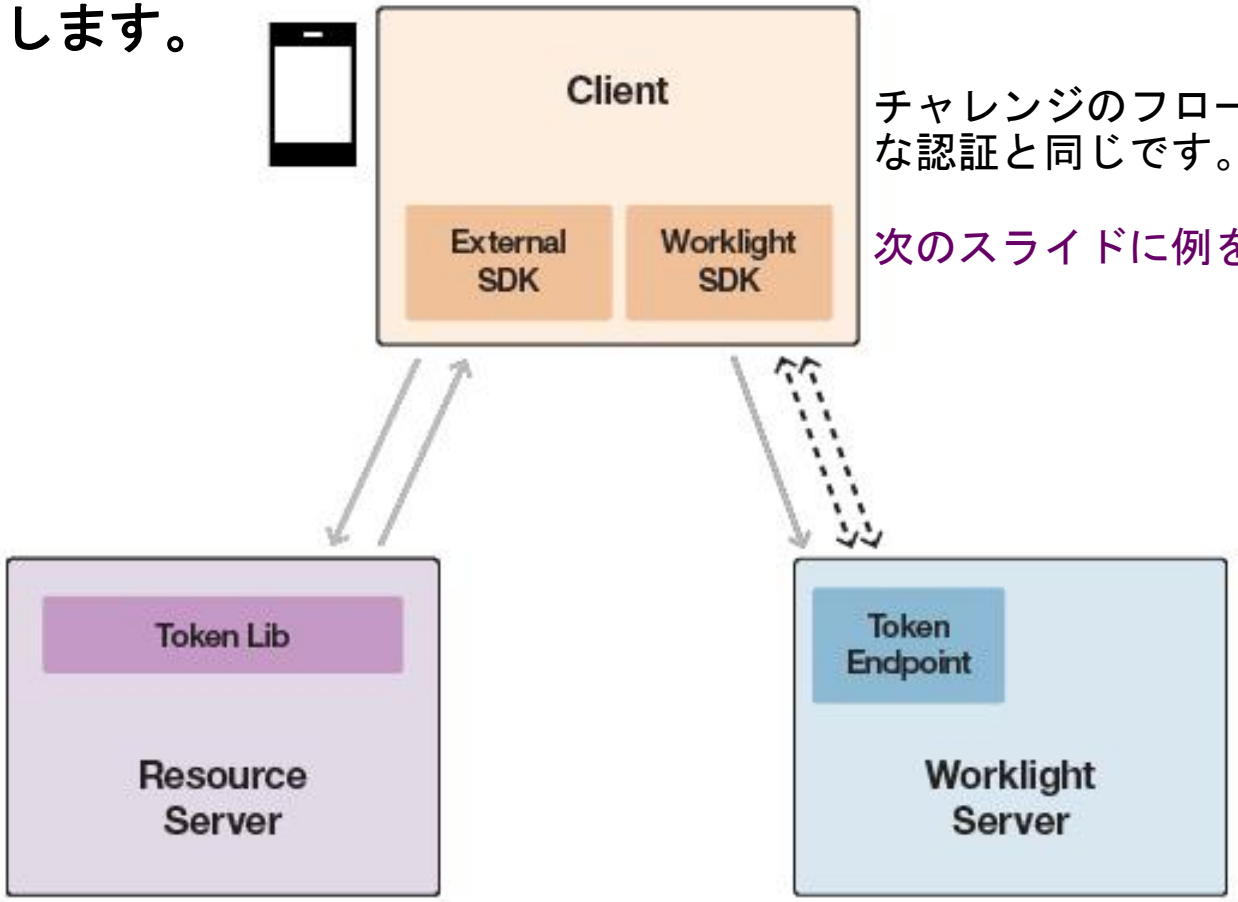


WL SDK は、オプションの scope パラメーターを指定して呼び出します。

スコープが指定されていない場合は、アプリケーションのセキュリティー・テストが使用されます。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (8/8)

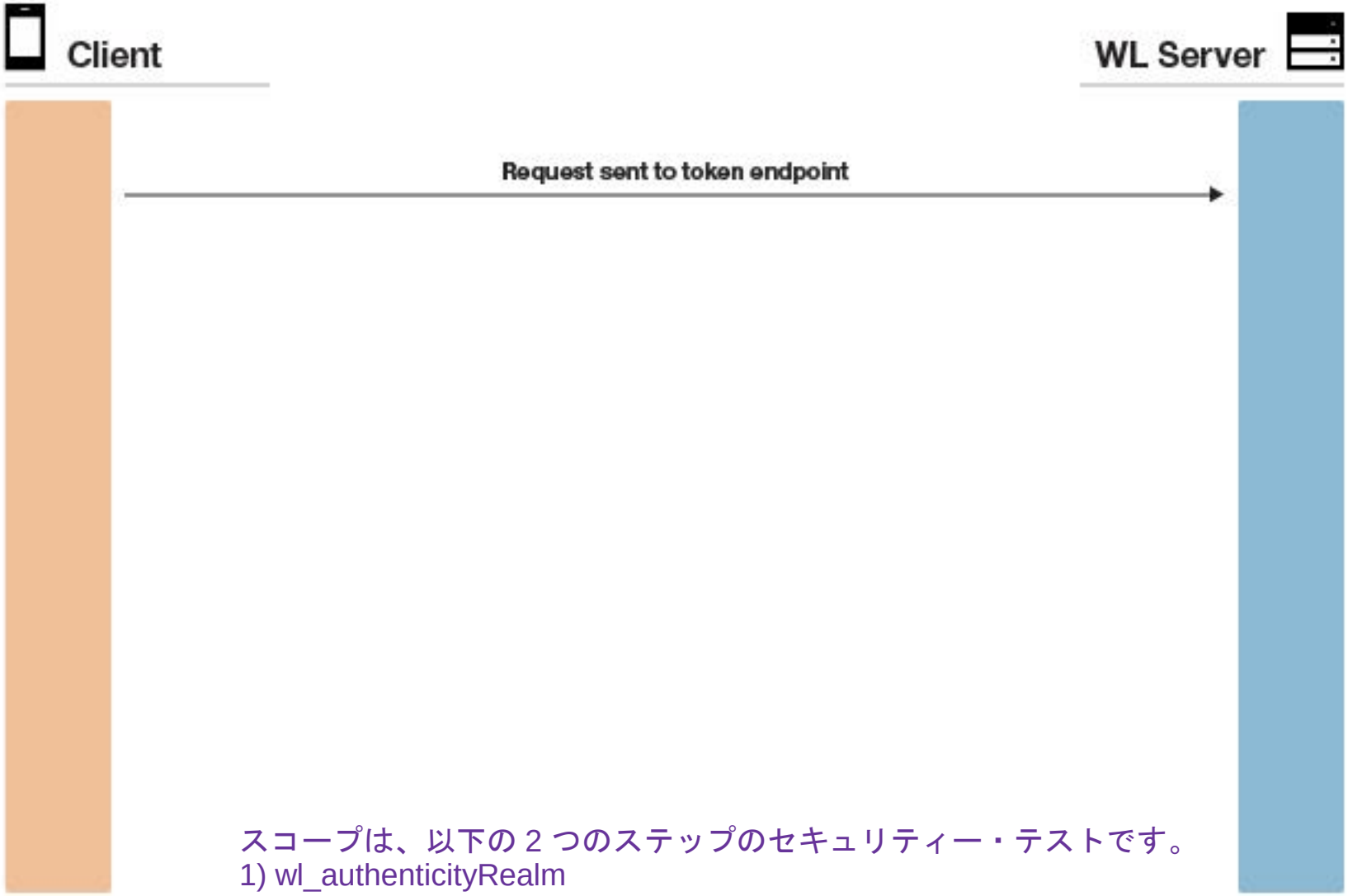
- **フロー・ステップ 4 - 一部のレルムがまだ認証されていない場合、Worklight Server はチャレンジをクライアントに返送し、クライアントが応答します。**



チャレンジのフローは、標準的な認証と同じです。

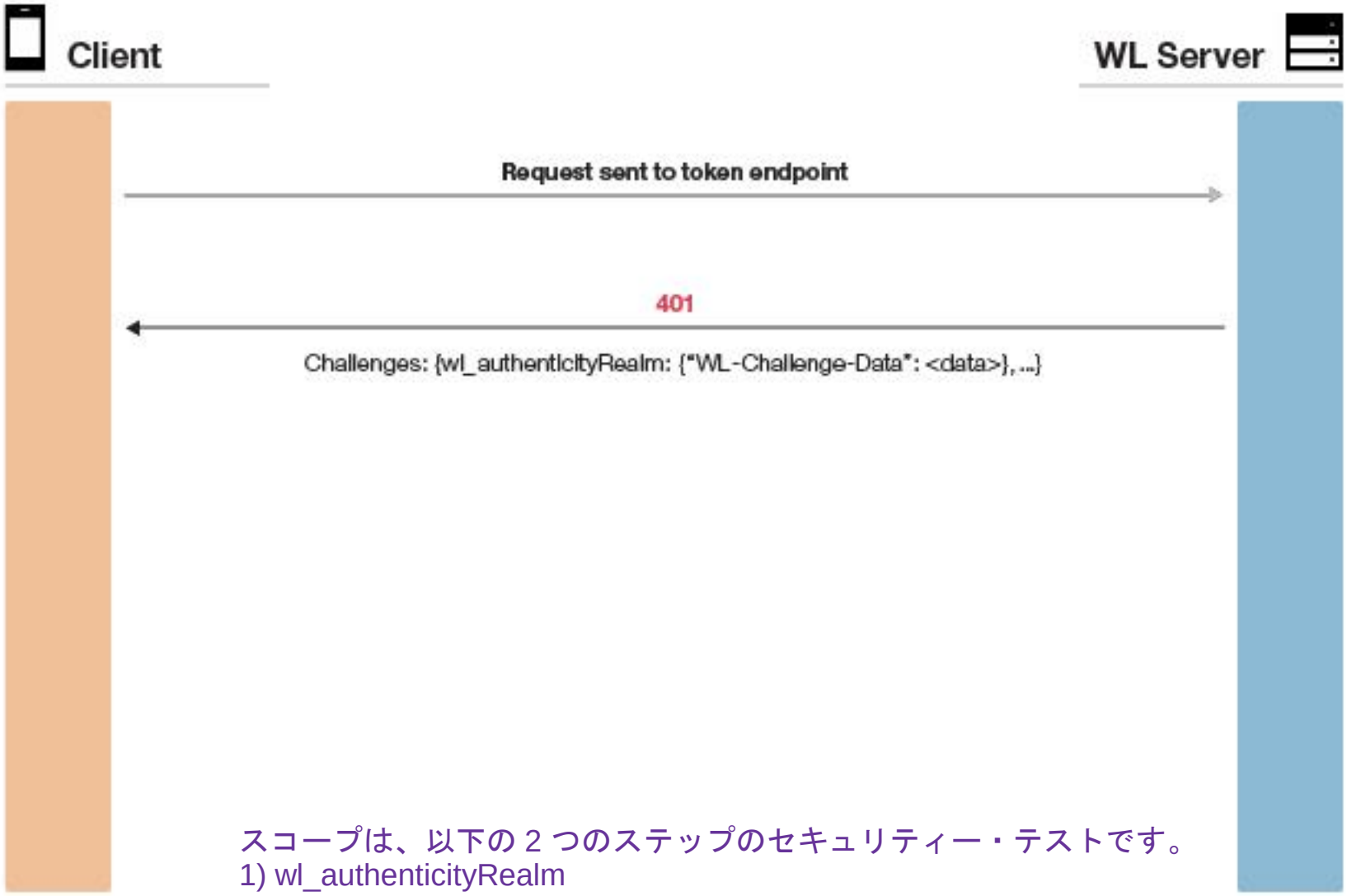
次のスライドに例を示します。

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - チャレンジ・フロー例 (1/6)



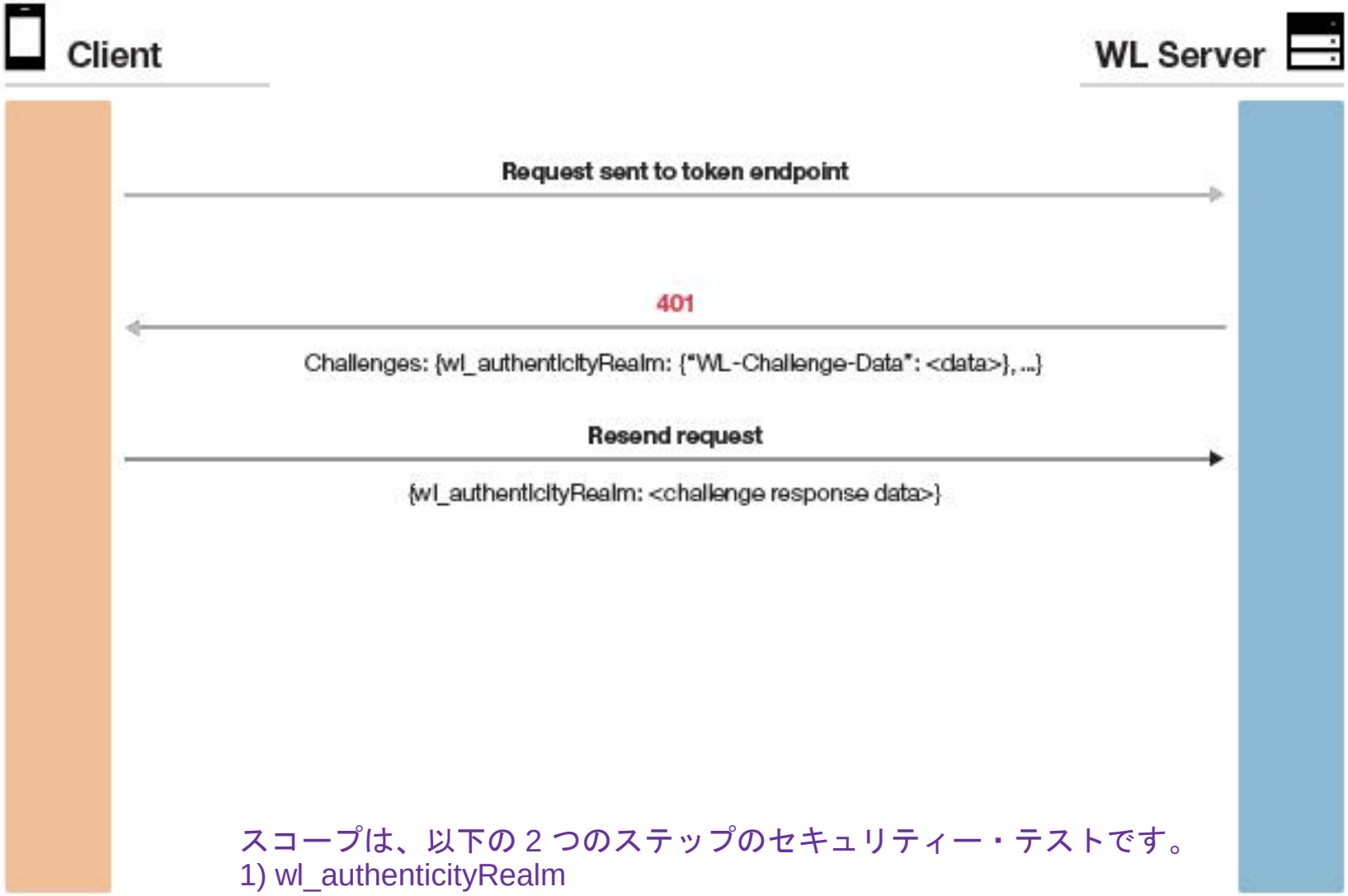
スコープは、以下の 2 つのステップのセキュリティー・テストです。
1) wl_authenticityRealm
2) wl_deviceAutoProvisioningRealm

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - チャレンジ・フロー例 (2/6)



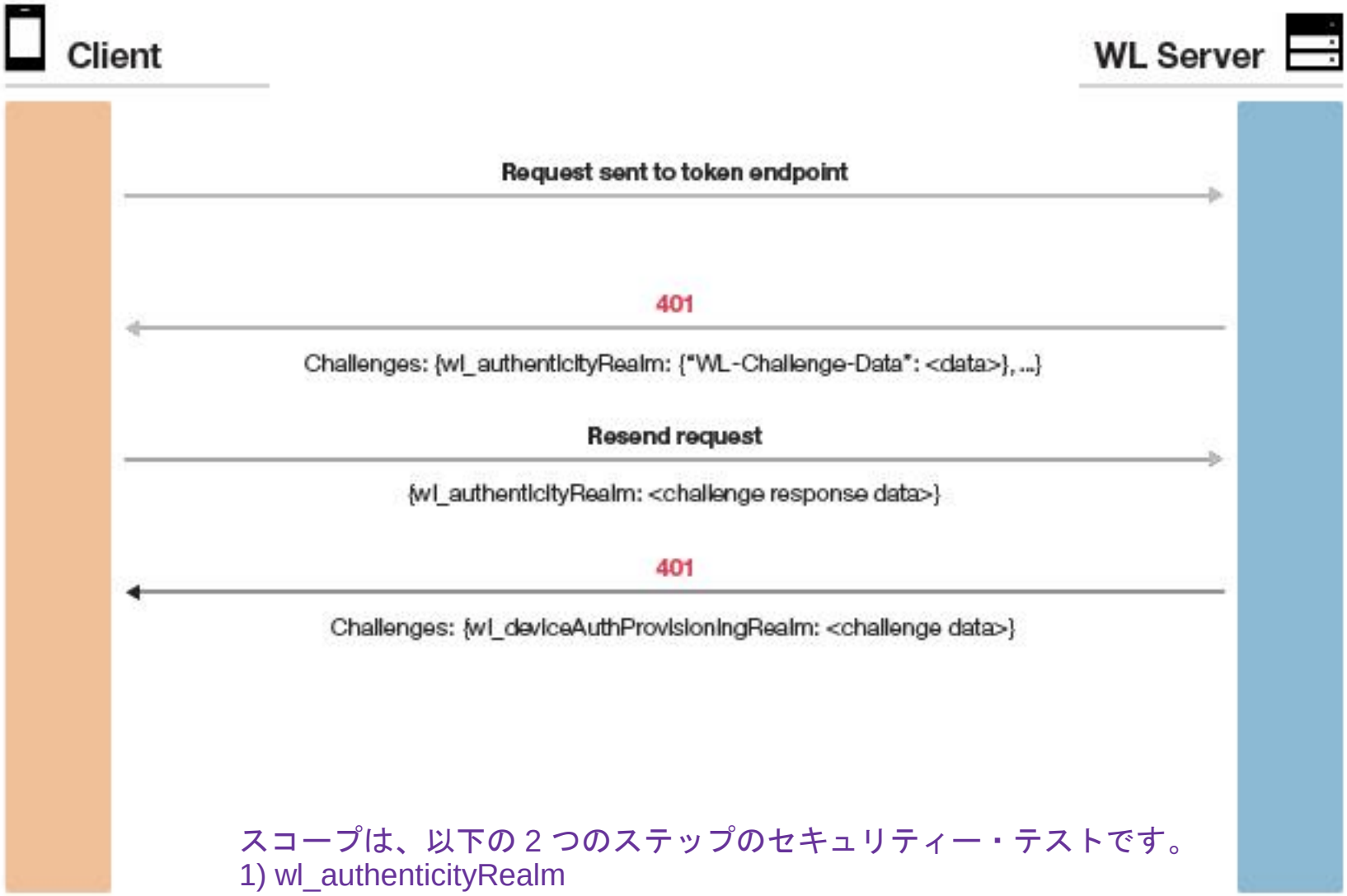
スコープは、以下の 2 つのステップのセキュリティー・テストです。
1) wl_authenticityRealm
2) wl_deviceAutoProvisioningRealm

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - チャレンジ・フロー例 (3/6)



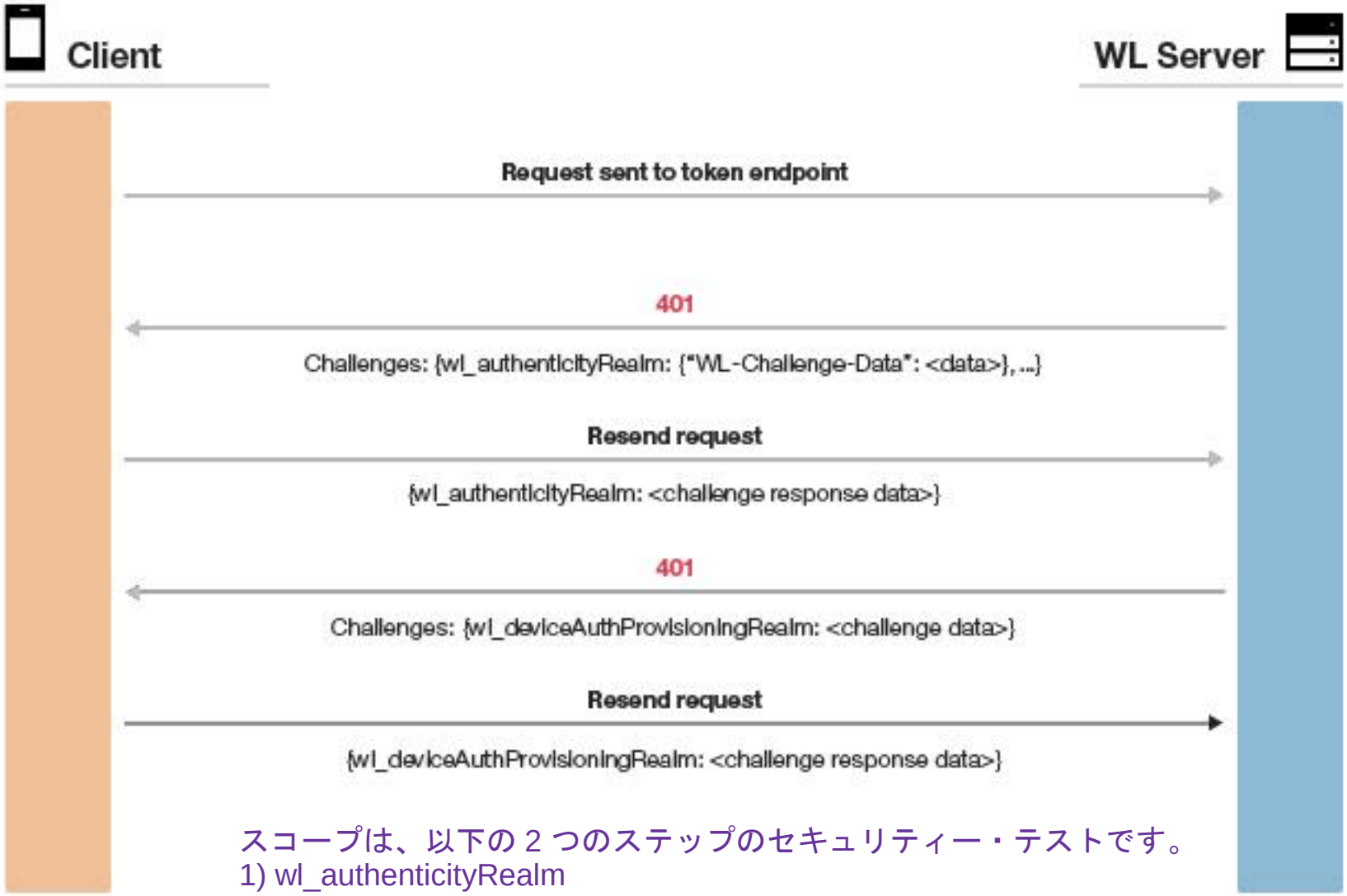
スコープは、以下の 2 つのステップのセキュリティー・テストです。
1) wl_authenticityRealm
2) wl_deviceAutoProvisioningRealm

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - チャレンジ・フロー例 (4/6)



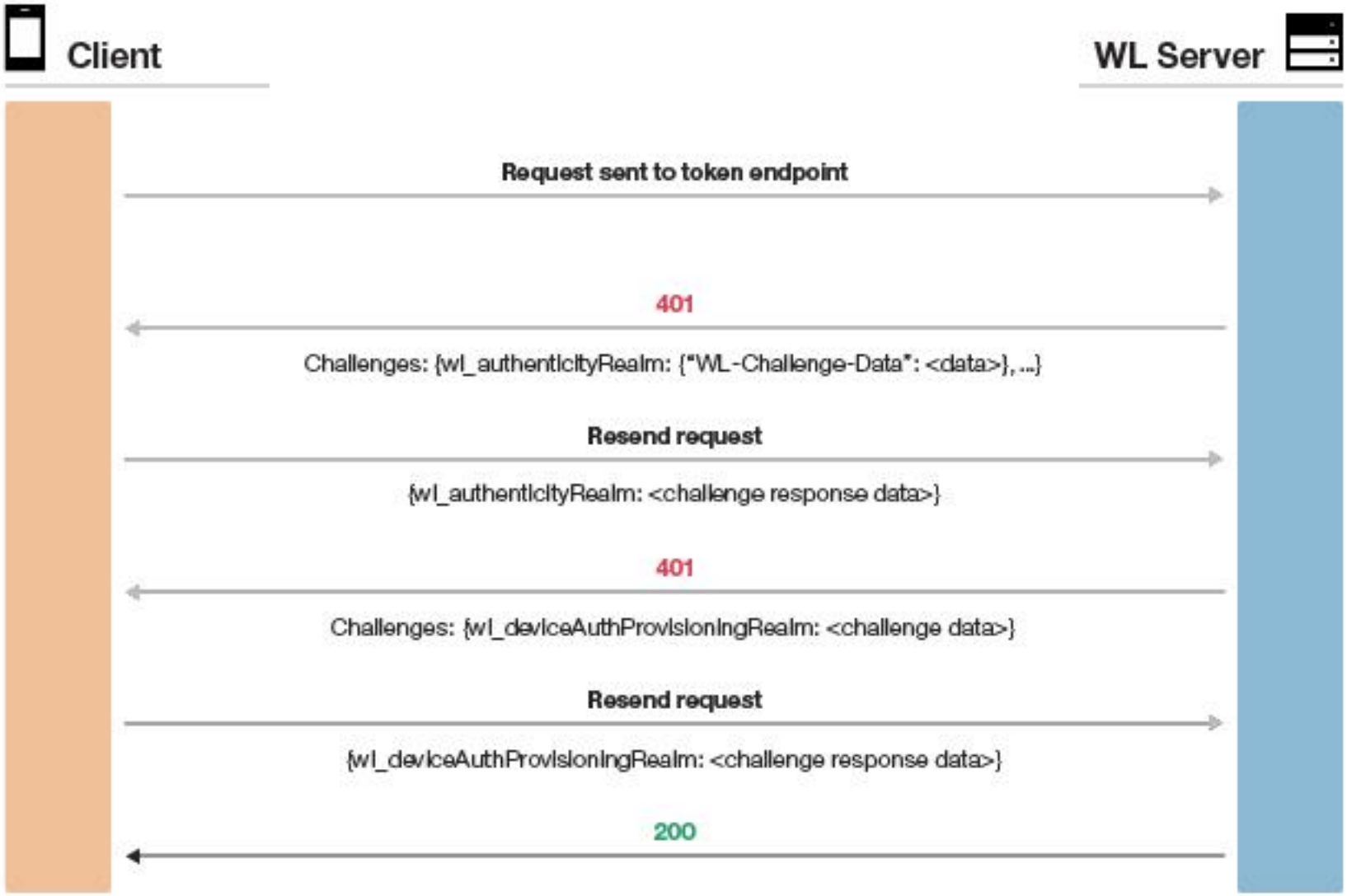
スコープは、以下の 2 つのステップのセキュリティー・テストです。
1) wl_authenticityRealm
2) wl_deviceAutoProvisioningRealm

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - チャレンジ・フロー例 (5/6)



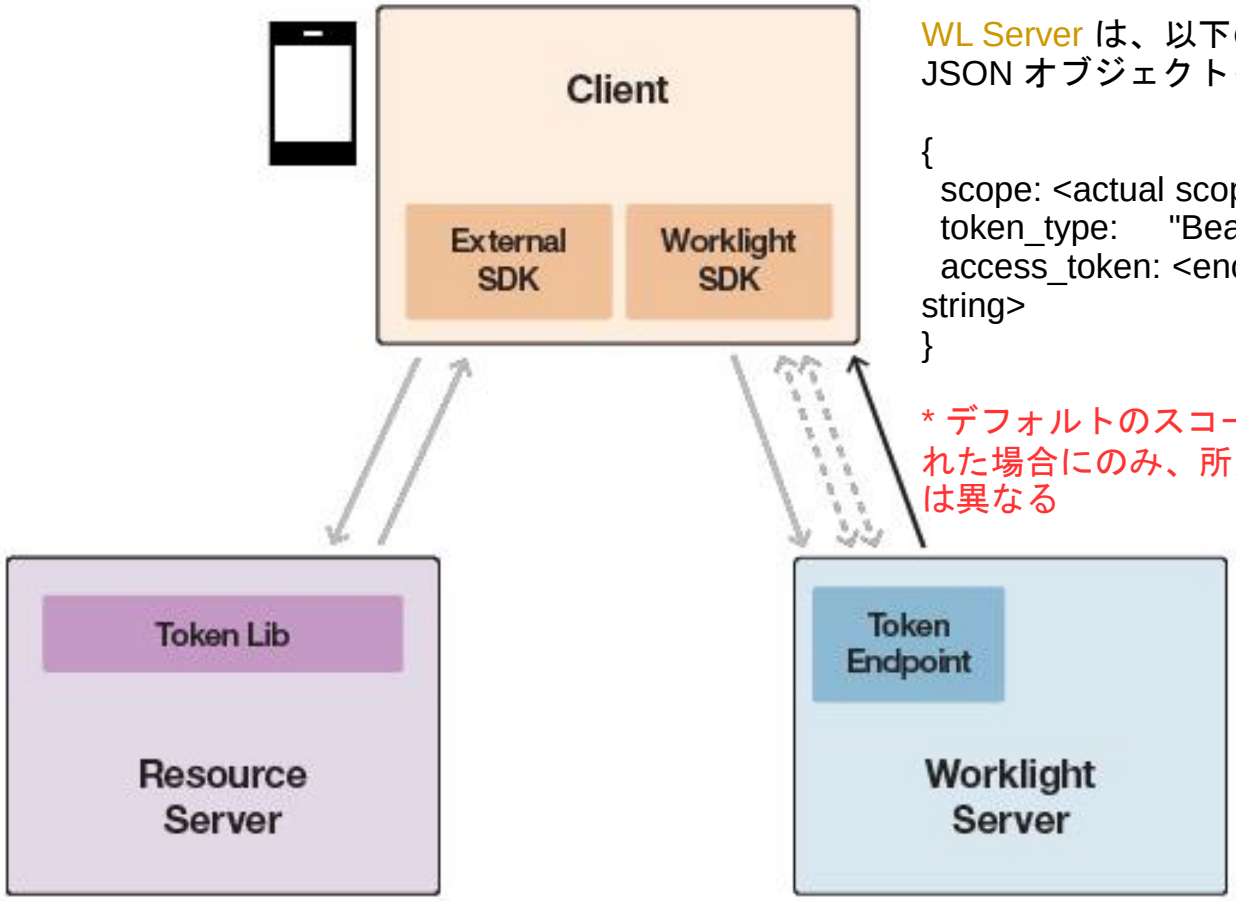
スコープは、以下の 2 つのステップのセキュリティー・テストです。
1) wl_authenticityRealm
2) wl_deviceAutoProvisioningRealm

OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 - チャレンジ・フロー例 (6/6)



OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (1/2)

- フロー・ステップ 5 - 関連するすべてのレルムが認証された後に、Worklight Server は、アクセス・トークンをクライアントに返します。



WL Server は、以下の形式の JSON オブジェクトを返します。

```
{
  scope: <actual scope> *
  token_type:     "Bearer"
  access_token: <encrypted token string>
}
```

* デフォルトのスコープが要求された場合にのみ、所定スコープとは異なる

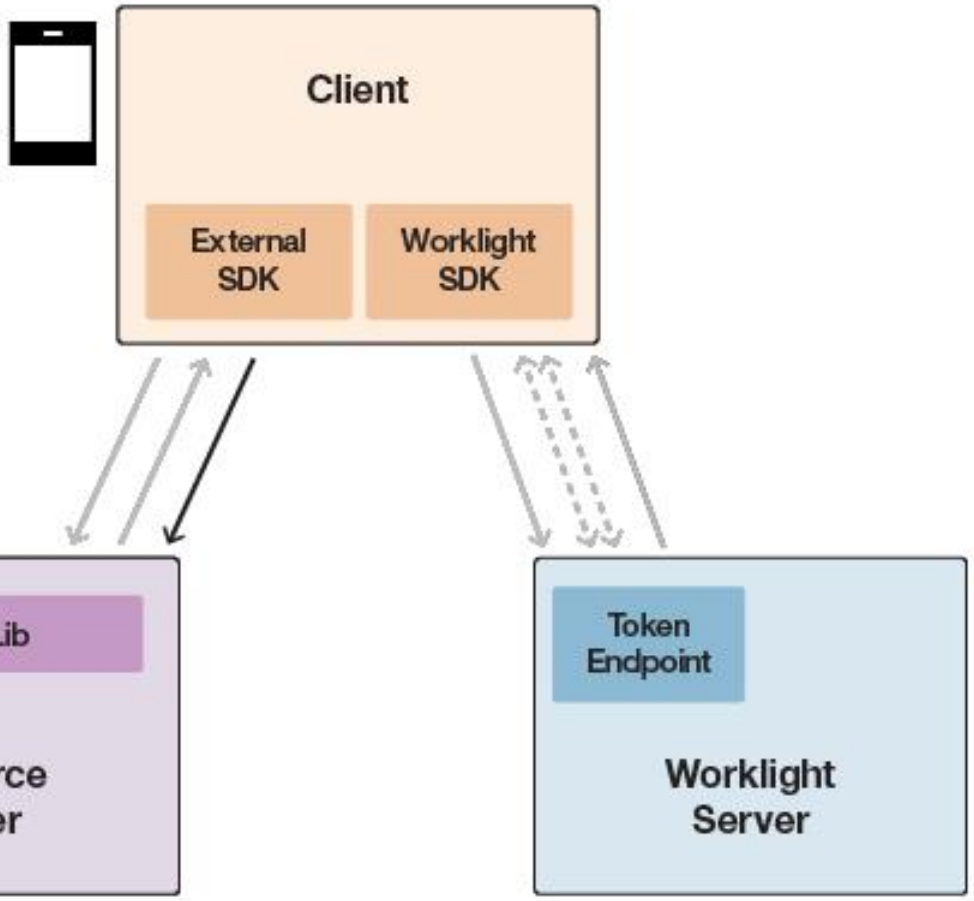
OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証 (2/2)

- **フロー・ステップ 6 – アプリケーション開発者は次に、取得したトークンが含まれた「Authorization」ヘッダーを追加して、元の要求を再送します。**

検証後に、トークン lib は、アクセス・トークンからの以下のデータを使用してクライアント・コンテキスト・オブジェクトを更新します:
app_id、user_id、device_id

トークン lib は、以下の順序でトークンを検証します。

1. 署名の検査。
2. 有効期限の検査。
3. スコープの検査。



アジェンダ

- 機能の概要
- OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証
- Worklight プロジェクトの構成
- 外部サービスの構成
- クライアント・サイド API の使用
- 演習

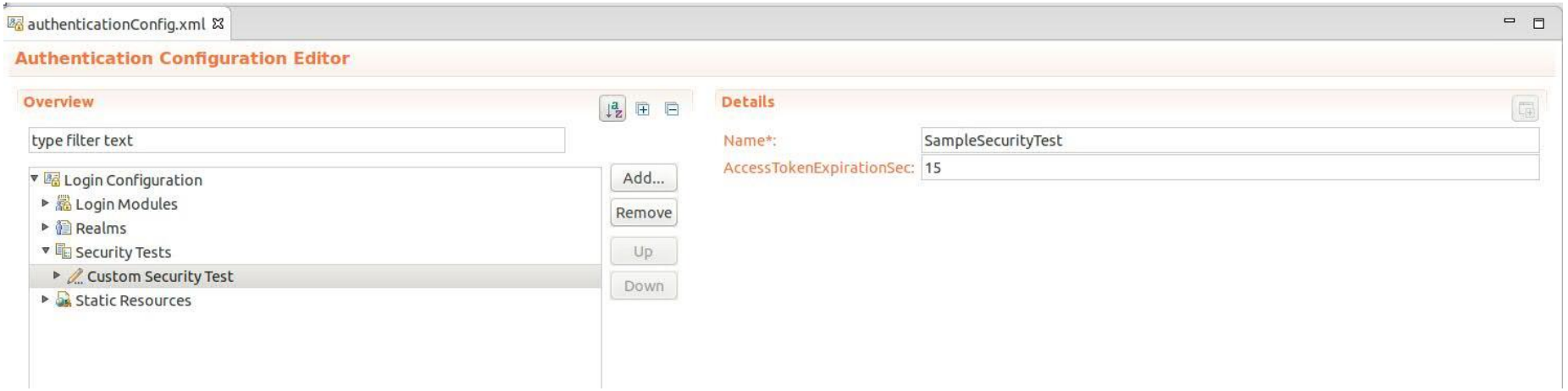
Worklight プロジェクトの構成 - アクセス・トークンのスコープの構成 (1/2)

- アクセス・トークンのスコープは、Worklight プロジェクト内の事前定義のセキュリティー・テストでなければなりません。次のファイルでスコープを構成します。 `<WL_Project>/server/conf/authenticationConfig.xml`.
- 各トークンのデフォルトの存続期間は 60 秒です。セキュリティー・テストに **AccessTokenExpirationSec** 属性を追加することで、このタイムアウトをオーバーライドできます。
- 例えば、存続期間が 15 秒の「SampleSecurityTest」というセキュリティー・テストを構成する場合、`authenticationConfig.xml` を以下のように編集してください。
  - ソース・ビューから:

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

Worklight プロジェクトの構成 - アクセス・トークンのスコープの構成 (2/2)

- 設計ビューから:



Worklight プロジェクトの構成 - 鍵ストア

- この機能を使用する場合は、独自の鍵ストアを使用または作成し、その鍵ストアを使用するように Worklight Server を構成することをお勧めします。
- 関係のないコンテキストでの例については、製品資料の『デバイス自動プロビジョニングの構成』を参照してください。
- デフォルトの Worklight Server 鍵ストアの使用は安全では**ありません**。

アジェンダ

- 機能の概要
- OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証
- Worklight プロジェクトの構成
- 外部サービスの構成
- クライアント・サイド API の使用
- 演習

外部サービスの構成

- 外部サービスがアクセス・トークンを受け入れるようにするには、オンラインまたはオフラインの検証でトークンを検証できる検証ライブラリーをサービスに追加する必要があります。
- この目的で以下の 2 つのライブラリーが提供されています。
  - `worklight-access-token-validator.jar` – Java ライブラリー
  - `worklight-access-token-validator.tgz` – node.js モジュール
- Worklight Server インストール済み環境の場合 – ライブラリーは、以下の場所にあります。
  <インストール・ディレクトリー>/WorklightServer/external-server-libraries
- Worklight Studio の場合 – 新規プロジェクトを作成すると、ライブラリーは、以下の場所にあります。
  <プロジェクト・ディレクトリー>/externalServerLibraries

外部サービスの構成 - Java の使用

- このモジュールの目的は、Java Web アプリケーション用に Worklight Server によって生成されたアクセス・トークンをオフラインで検証できるようにすることです。
- node.js サーバーの場合も、Worklight Server によって生成されたアクセス・トークンの検証は可能です。
- Java ライブラリーを使用するには、以下の 2 つのファイルが必要です。
  - 証明書 – 関連サンプルでは、Worklight Server 鍵ストアからエクスポートされた証明書。Java keytool を使用できます。実動では、「スライド 27 - Worklight プロジェクト構成 - 鍵ストア」で説明したように、独自の鍵ストアを使用することをお勧めします。
  - `Worklight-access-token-validator.jar`

外部サービスの構成 - Java の使用: サーブレット・フィルター (1/3)

- Web アプリケーションのクラスパスに `worklight-access-token-validator.jar` ファイルを追加し、以下に示すように、フィルター・クラス `com.worklight.security.WLAccessTokenValidationFilter` を使用します。
- フィルター定義の例:

Filter name = FilterName:

フィルターの名前を選択します。

URL = /some/protected/url

保護するすべてのリソースのプレフィックス。

Scope [オプション] = securityTestName

`authenticationConfig.xml` ファイルで定義された、セキュリティー・テストの名前。保護リソースにアクセスできるように、認証するために必要です。スコープが指定されていない場合、フィルターは、Worklight Server で提供された有効なトークンはすべて受け入れます。

CertificatePath = certificateLib/WorklightServerCeriificate.cert。

WEB-INF フォルダーを基準とした、Worklight Server の証明書の相対パス。

外部サービスの構成 - Java の使用: サーブレット・フィルター (2/3)

- 前述のパラメーターに加え、外部サーバーの `web.xml` ファイルに以下のコードを記述します。

```
<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>
```

外部サービスの構成 - Java の使用: サーブレット・フィルター (3/3)

- 検証が成功すると、フィルターは、アクセス・トークンに含まれているユーザー ID、アプリケーション ID、またはデバイス ID にアクセスするためにサービスが使用できる `ClientContext` オブジェクトを更新します。
- `ClientContext` の使用例:

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

アジェンダ

- 機能の概要
- OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証
- Worklight プロジェクトの構成
- 外部サービスの構成
- クライアント・サイド API の使用
- 演習

クライアント・サイドAPI の使用方法: 許可されるメソッド(1/4)

- Worklight の `WL.Client` API は、以下のプラットフォームでの Worklight アクセス・トークンの使用のための組み込みサポートを提供します。
  - ハイブリッド – JavaScript™
  - Android
  - iOS
- この API 内のメソッドには、以下のサービスが用意されています。
  - 指定スコープのトークンの取得およびキャッシュ
  - 最後に取得したアクセス・トークンの取得
  - 外部サービス応答からの必要なスコープの取得
- 以降のスライドを参照してください。詳しくは、製品資料の「IBM Worklight Foundation と外部サービスの間での SSO の使用」を参照してください。

クライアント・サイドAPI の使用方法: 許可されるメソッド(2/4)

- **指定スコープのトークンの取得およびキャッシュ**

JavaScript では、このメソッドは **obtainAccessToken** と呼ばれています。

- `WL.Client` インスタンスは、Worklight Server インスタンスに対して新規トークンを要求します。トークンを取得するには、クライアントは、(Worklight Server の `AuthenticationConfig.xml` 構成ファイル 内のセキュリティー・テストで表された) 要求スコープのすべてのレルムで認証されている必要があります。したがって、このメソッドを呼び出すと、まだ認証が必要なすべてのレルムに対して認証シーケンスがトリガーされます。
- このメソッドは、すべてのプラットフォームで非同期です。値を返しませんが、代わりに、応答ハンドラーをトリガーします。
  - 注: 応答ハンドラーでサーバーからの応答を解析する必要はありません。トークンは自動的に解析されて、`WL.Client` インスタンス内にキャッシュされるため、次のメソッドを使用して取得できます。

クライアント・サイドAPI の使用方法: 許可されるメソッド(3/4)

- **最後に取得したアクセス・トークンの取得**

JavaScript では、このメソッドは **getLastAccessToken** と呼ばれています。

- WL.Client インスタンスは、特定スコープの最後のアクセス・トークンをストリングとして返します。あるいは、スコープを指定しなかった場合、最後に取得したトークンが返されます。この機能は、アプリケーションが 1 つのスコープのみを使用している場合に役に立ちます。
- 「Authorization」というヘッダーを追加し、ヘッダーの内容として、「Bearer」に続いてトークンを追加します。例えば、Ajax 要求を発行する場合、以下のコードのように記述できます。

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

クライアント・サイドAPI の使用方法: 許可されるメソッド(4/4)

- **外部サービス応答からの必要なスコープの取得**

JavaScript では、このメソッドは **getRequiredAccessTokenScope** と呼ばれています。

外部サービスへの要求が失敗した場合、`WL.Client` インスタンスは失敗の原因を特定できます。

- その失敗がアクセス・トークンの問題に関連している場合は、サービスにアクセスするために必要なスコープの名前を返します。この場合、返されたスコープの新規トークンを取得する必要があります。例えば、必要なスコープにトークンが一致しない場合やトークンの有効期限が切れている場合です。
- エラーがアクセス・トークンの問題に関係していない場合、このメソッドは `null` を返します。

クライアント・サイドAPI の使用方法: JavaScript の例 (1/2)

- 以下の JavaScript の例では、外部サービスにアクセスする際にクライアント・サイド API を使用する方法を示します。

```
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} : {};
```

(次のスライドに続く…)

クライアント・サイドAPI の使用方法: JavaScript の例 (2/2)

```
        $.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);
        };
}
```

アジェンダ

- 機能の概要
- OAuth 2.0 を介したアクセス・トークンを使用した Worklight の認証
- Worklight プロジェクトの構成
- 外部サービスの構成
- クライアント・サイド API の使用
- 演習

演習 (1/3)

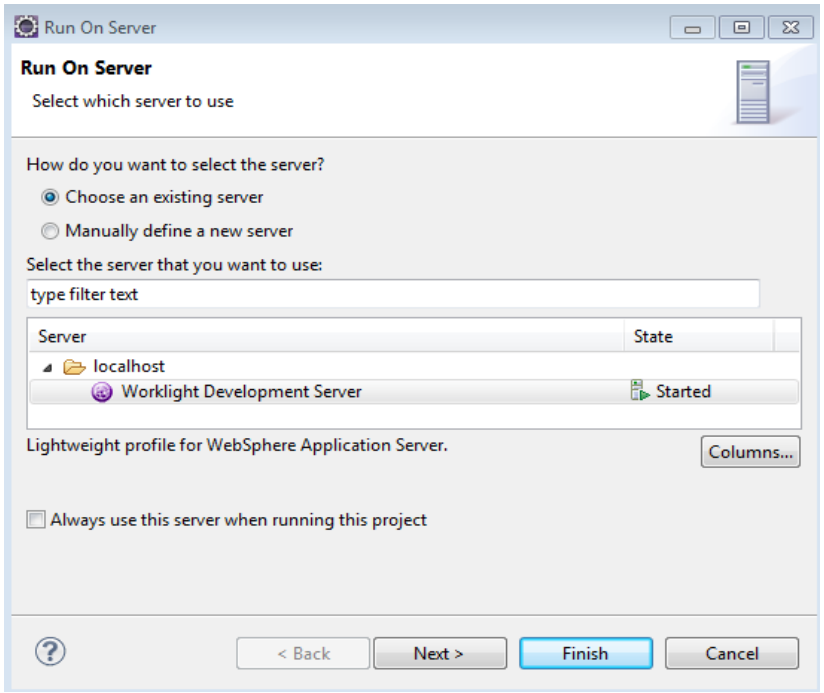
- プロジェクト `WorklightAsAuthorizationServer.zip` を参照してください。このトレーニング・モジュールのサンプルは、IBM® Worklight Foundation 文書 Web サイト (http://www.ibm.com/mobile-docs) の「入門」ページにあります。
- `WorklightAsAuthorizationServer` プロジェクトには 2 つのアイテムが含まれています。
  - `REST-Caller` は、標準 Worklight プロジェクトです。
  - プロジェクト `REST-Caller` には、**REST-Server** フォルダーが含まれています。このフォルダーには、アーカイブ `REST-Server.zip` が含まれており、その中には Java Web アプリケーションがあります。

演習 (2/3)

1. Eclipse で、**「ファイル」**>**「インポート」**を選択してから、ウィザードで、**「既存プロジェクトをワークスペースへ」**を選択します。
2. **「アーカイブ・ファイルの選択」**オプションをクリックし、現在のワークスペース・プロジェクト `REST-Caller` にナビゲートし、`REST-Server.zip` ファイルをインポートします。
3. REST-Caller Worklight アプリケーションをデプロイするには、コマンド**「Worklight Development Server で実行 (Run on Worklight Development Server)」**を選択します。

演習 (3/3)

4. REST-Server Java Web アプリケーションをデプロイするには、プロジェクトを右クリックして**「サーバーで実行 (Run On Server)」**を選択してから、**「Worklight Development Server」**を選択します。



特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました