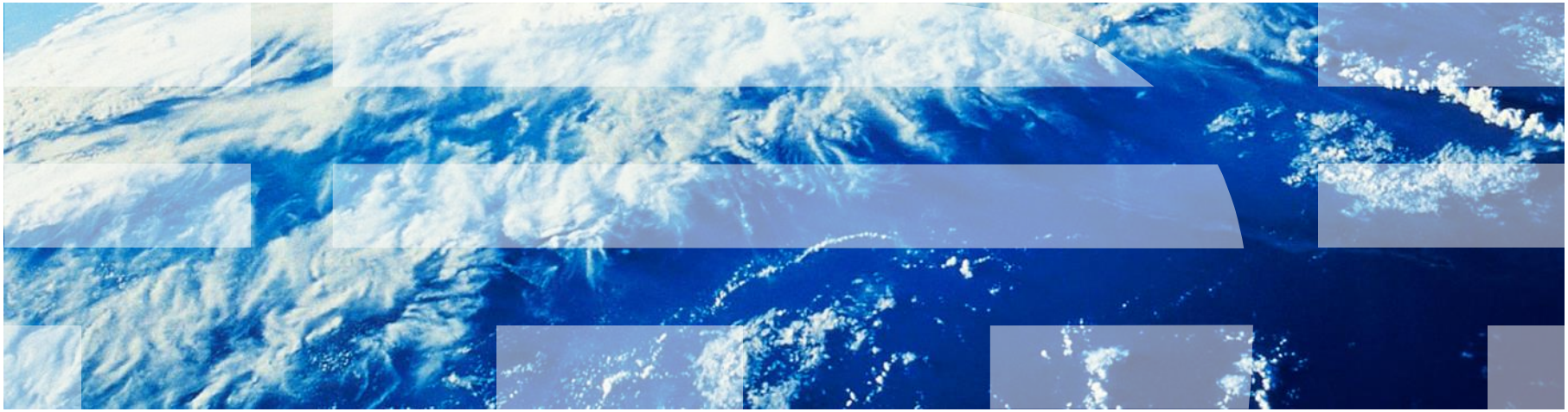
IBM Worklight Foundation V6.2.0
入門

カスタム・デバイス・プロビジョニング



2014 年 6 月 20 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。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://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。

- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

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

- 概要
- カスタム・デバイス・プロビジョニングについて
- authenticationConfig.xml の構成
- サーバー・サイド・コンポーネントの実装
- クライアント・サイド・コンポーネントの実装
- 結果の確認

概要

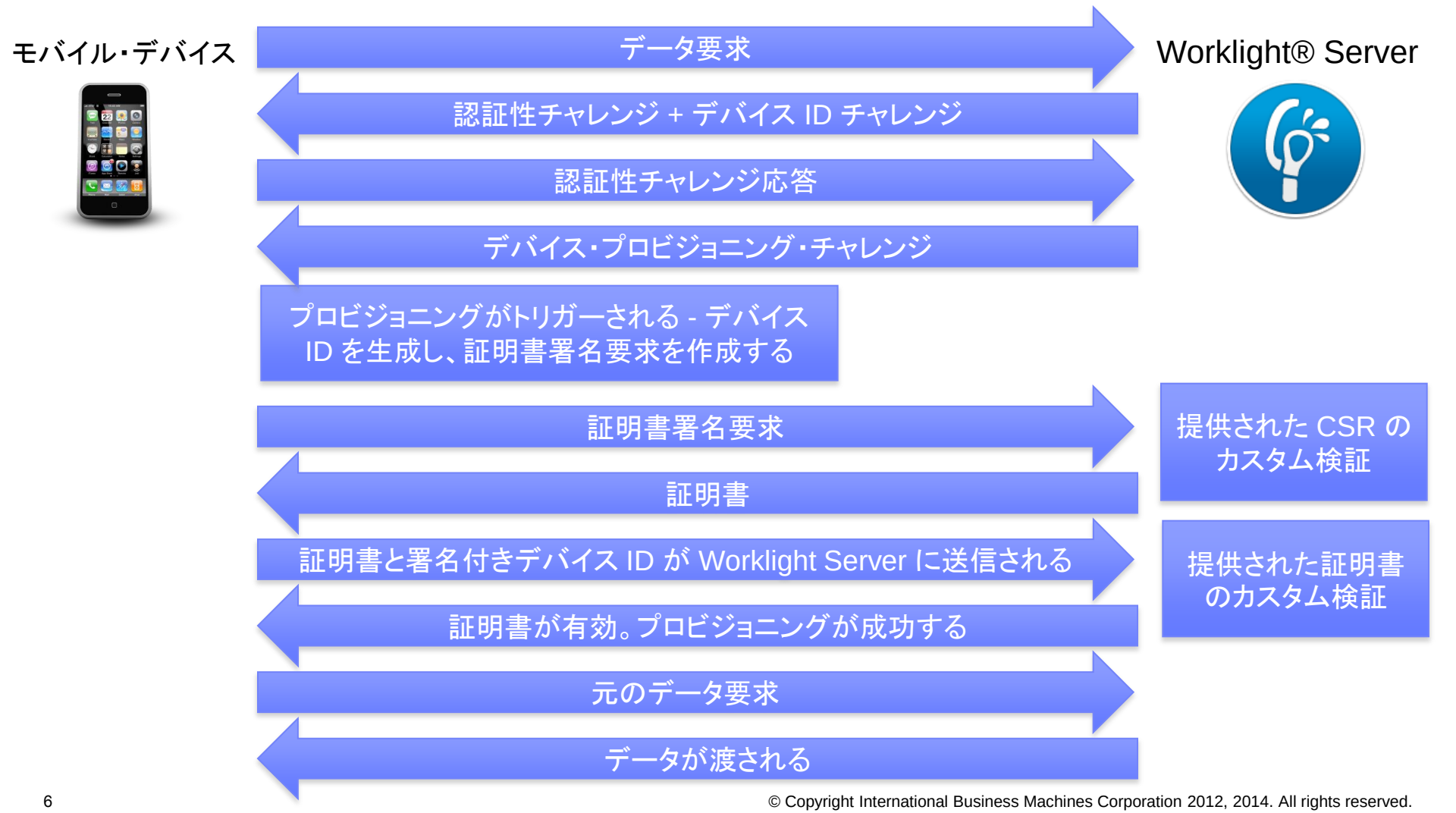
- このトレーニング・モジュールでは、カスタム・デバイス・プロビジョニングの有効化方法と構成方法について学習します。
- カスタム・デバイス・プロビジョニングは自動デバイス・プロビジョニングを拡張したもので、以下のカスタム検証を実装できます。
  - 初期プロビジョニング・フロー時の証明書署名要求。
  - 毎回のアプリケーション開始時の証明書。
- このトレーニング・モジュールはデバイス・プロビジョニングの概念に基づいているため、*『デバイス・プロビジョニングの概念』*トレーニング・モジュールで説明するトピックを十分に理解しておくことが重要です。

アジェンダ

- 概要
- カスタム・デバイス・プロビジョニングについて
- authenticationConfig.xml の構成
- サーバー・サイド・コンポーネントの実装
- クライアント・サイド・コンポーネントの実装
- 結果の確認

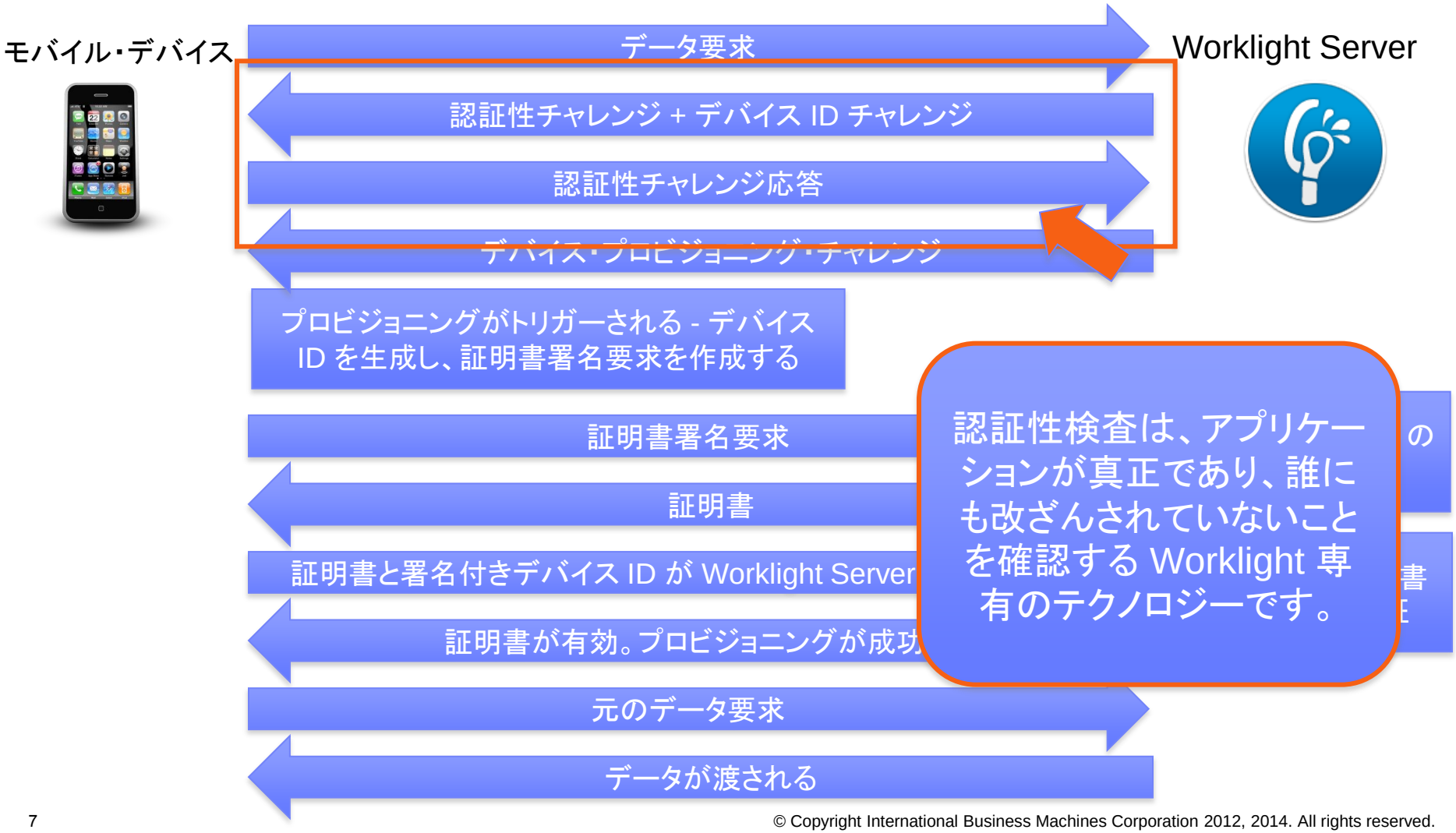
カスタム・デバイス・プロビジョニングについて(1/5)

- カスタム・デバイス・プロビジョニングのフロー - 初回のアプリケーション始動時



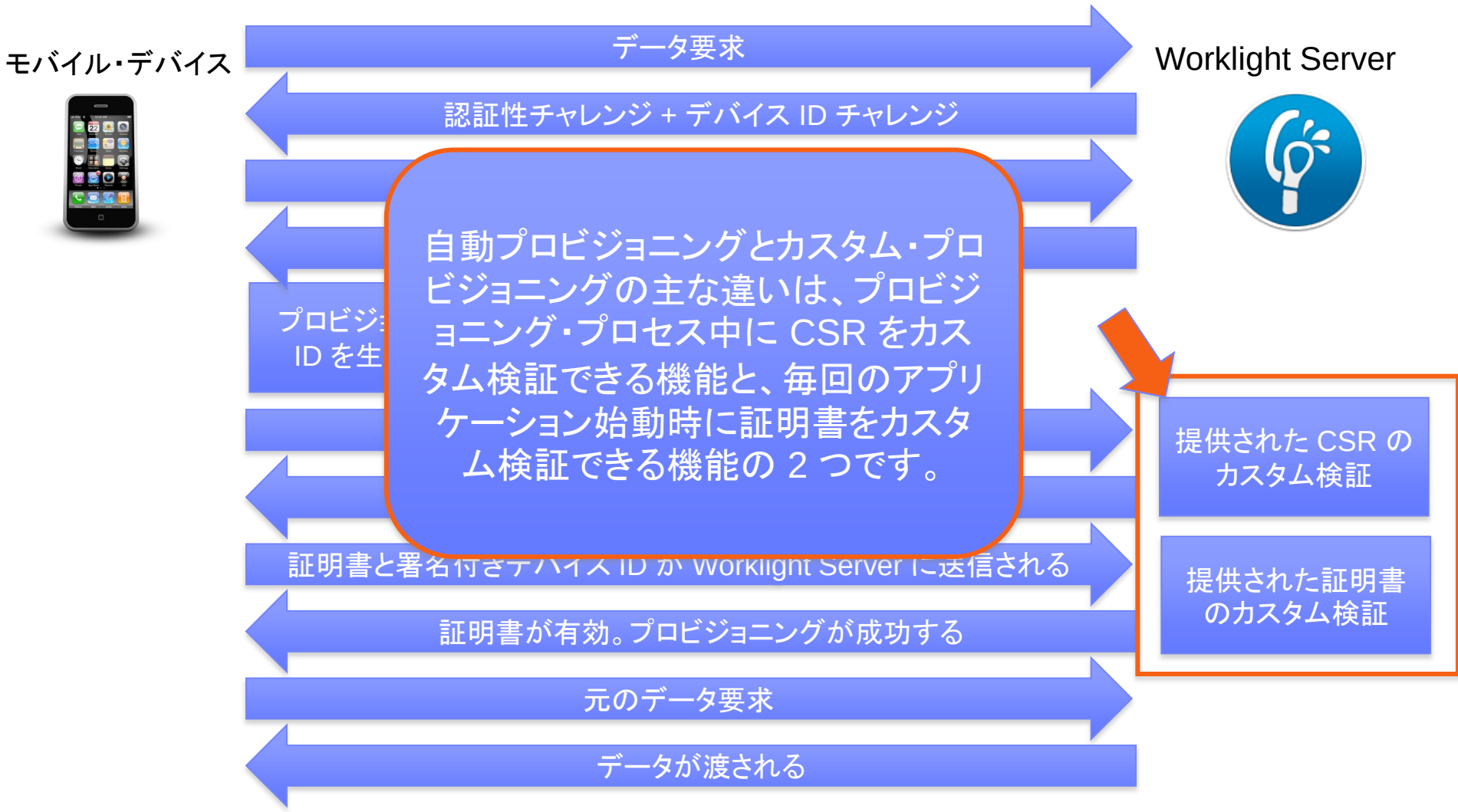
カスタム・デバイス・プロビジョニングについて(2/5)

- カスタム・デバイス・プロビジョニングのフロー - 初回のアプリケーション始動時



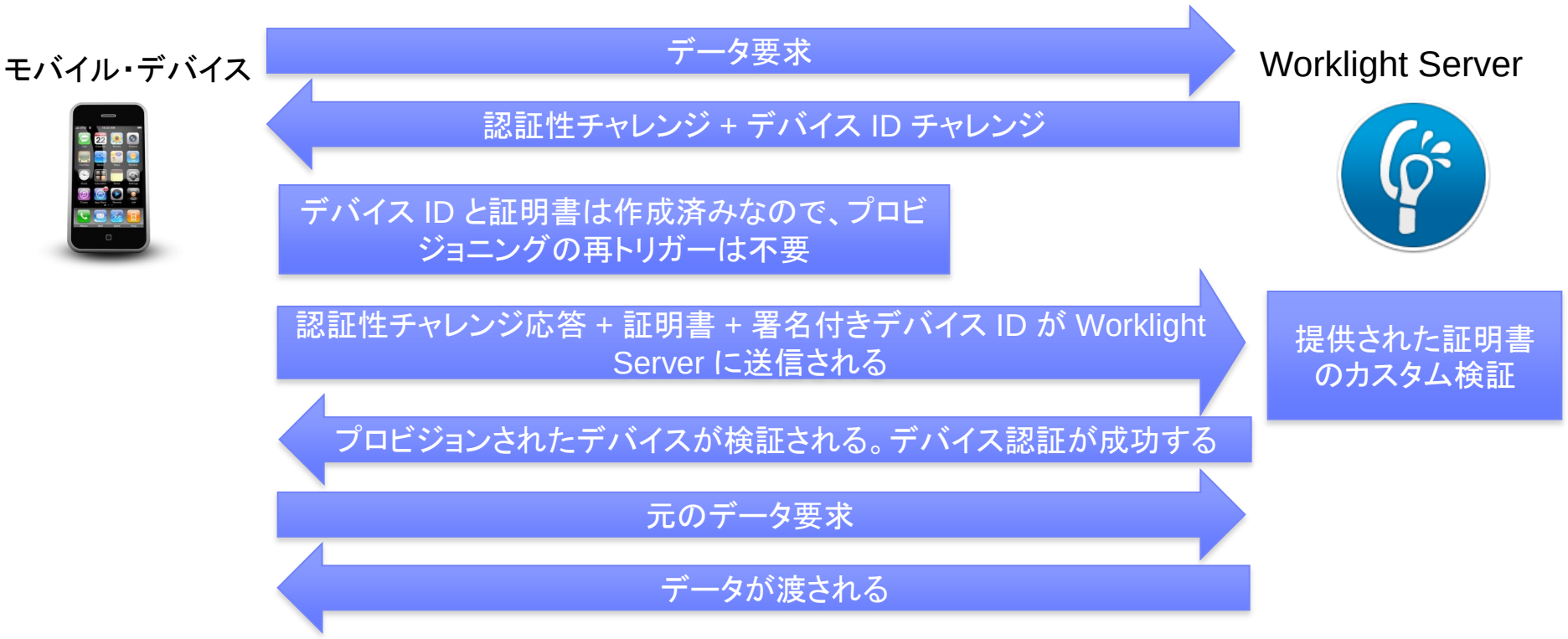
カスタム・デバイス・プロビジョニングについて(3/5)

- カスタム・デバイス・プロビジョニングのフロー - 初回のアプリケーション始動時



カスタム・デバイス・プロビジョニングについて(4/5)

- カスタム・デバイス・プロビジョニングのフロー - 2 回目以降のアプリケーション開始時



カスタム・デバイス・プロビジョニングについて (5/5)

- デフォルトでは、Worklight Server は、内部鍵ストアを使用して証明書を発行します。
- 独自の鍵ストアを使用するように Worklight Server に指示するには、`worklight.properties` ファイルを調整します。

```
##########################################################################################################################
#    Worklight Default Certificate (For device provisioning)
##########################################################################################################################
# You can change the default behavior with regard to CA certificates. You can also implement custom provisioning.
# If you want to change the auto-provisioning mechanism to use different granularity (application, device or group) or a
# different list of pre-required realms, you can create your own customized authenticator, login module and challenge handler.
# For more information, see the "Custom Authenticator and Login Module" Getting Started training module.

#The path to the keystore, relative to the server folder in the Worklight Project, for example: conf/my-cert.jks
#wl.ca.keystore.path=
#The type of the keystore file. Valid values are jks or pkcs12.
#wl.ca.keystore.type=
#The password to the keystore file.
#wl.ca.keystore.password=
#The alias of the entry where the private key and certificate are stored, in the keystore.
#wl.ca.key.alias=
#The password to the alias in the keystore.
#wl.ca.key.alias.password=

##########################################################################################################################
#    Worklight SSL keystore
##########################################################################################################################
#SSL certificate keystore location.
ssl.keystore.path=conf/default.keystore
#SSL certificate keystore type (jks or PKCS12)
ssl.keystore.type=jks
#SSL certificate keystore password.
ssl.keystore.password=worklight
```

- **注:** `wl.ca.keystore.path` プロパティー値は、Worklight プロジェクトの `/server/` フォルダーに対する相対パスか、ファイル・システムの絶対パスのいずれかにすることができます。

アジェンダ

- 概要
- カスタム・デバイス・プロビジョニングについて
- authenticationConfig.xml の構成
- サーバー・サイド・コンポーネントの実装
- クライアント・サイド・コンポーネントの実装
- 結果の確認

authenticationConfig.xml の構成 (1/3)

- 最初に、**CustomDeviceProvisioningRealm** という名前のレルムを `authenticationConfig.xml` ファイルに追加します。
- **CustomDeviceProvisioningLoginModule** を使用します。
- 自動プロビジョニング・オーセンティケーターの className パラメーターを使用します。
- **validate-csr-function** パラメーターを追加します。
- このパラメーターの値は、CSR の検証を実行するアダプター関数を指しています。

```
<realms>
    <realm name="CustomDeviceProvisioningRealm" loginModule="CustomDeviceProvisioningLoginModule">
        <className>com.worklight.core.auth.ext.DeviceAutoProvisioningAuthenticator</className>
        <parameter  name="validate-csr-function"
                    value="ProvisioningAdapter.validateCSR"/>
    </realm>
</realms>
```

authenticationConfig.xml の構成 (2/3)

- **CustomDeviceProvisioningLoginModule** を追加します。
- 自動プロビジョニング・ログイン・モジュールの className パラメーターを使用します。
- **validate-certificate-function** パラメーターを追加します。
- このパラメーターの値は、証明書の検証を実行するアダプター関数を指しています。

```
<loginModules>
    <loginModule name="CustomDeviceProvisioningLoginModule">
        <className>com.worklight.core.auth.ext.DeviceAutoProvisioningLoginModule</className>
        <parameter  name="validate-certificate-function"
                    value="ProvisioningAdapter.validateCertificate"/>
    </loginModule>
</loginModules>
```

authenticationConfig.xml の構成 (3/3)

- **mobileSecurityTest** を作成します。
- 必須の `<testAppAuthenticity/>` テストを追加します。
- 必須の `<testDeviceId/>` テストを追加します。
- `provisioningType="custom"` を指定します。
- `realm="CustomDeviceProvisioningRealm"` を指定します。

```
<securityTests>
    <mobileSecurityTest name="CustomDeviceProvisioningSecurityTest">
        <testAppAuthenticity/>
        <testDeviceId provisioningType="custom" realm="CustomDeviceProvisioningRealm"/>
    </mobileSecurityTest>
</securityTests>
```

アジェンダ

- 概要
- カスタム・デバイス・プロビジョニングについて
- authenticationConfig.xml の構成
- サーバー・サイド・コンポーネントの実装
- クライアント・サイド・コンポーネントの実装
- 結果の確認

サーバー・サイド・コンポーネントの実装 (1/7)

- **ProvisioningAdapter** というアダプターを作成します。
- 以下の署名付きの 2 つの関数をアダプターの JavaScript™ ファイルに追加します。
  - `validateCSR(clientDN, csrContent)` – この関数は、初回のデバイス・プロビジョニング時にのみ呼び出されます。この使用目的は、当該デバイスのプロビジョンが許可されているかどうかを検査することです。デバイスが一度プロビジョンされたら、この関数が再度呼び出されることはありません。
  - `validateCertificate (certificate, customAttributes)` – この関数は、モバイル・アプリケーションが Worklight Server と新しいセッションを確立するたびに呼び出されます。この使用目的は、アプリケーション/デバイスが所有する証明書がまだ有効であることと、アプリケーション/デバイスが Worklight Server との通信を許可されているか検証することです。
- これらの関数は Worklight 認証フレームワークによって内部で呼び出されます。そのため、アダプター XML ファイルの XML ファイルで宣言しないでください。

サーバー・サイド・コンポーネントの実装 (2/7)

- validateCSR (clientDN, csrContent) 関数を実装します。

```
function validateCSR(clientDN, csrContent){
    WL.Logger.info("validateCSR :: clientDN :: " + JSON.stringify(clientDN));
    WL.Logger.info("validateCSR :: csrContent :: " + JSON.stringify(csrContent));

    var activationCode = csrContent.activationCode;
    var response;

    // This is a place to perform validation of csrContent and update clientDN if required.
    // You can do it using adapter backend connectivity
    if (activationCode === "worklight"){
        response = {
            isSuccessful: true,
            clientDN: clientDN + ",CN=someCustomData",
            attributes: {
                customAttribute: "some-custom-attribute"
            }
        };
    } else {
        response = {
            isSuccessful: false,
            errors: ["Invalid activation code"]
        };
    }

    return response;
}
```

activationCode は、クライアント・サイドの CSR に追加するカスタム・プロパティーです。

サーバー・サイド・コンポーネントの実装 (3/7)

- validateCSR (clientDN, csrContent) 関数を実装します。

```
function validateCSR(clientDN, csrContent){
    WL.Logger.info("validateCSR :: clientDN :: " + JSON.stringify(clientDN));
    WL.Logger.info("validateCSR :: csrContent :: " + JSON.stringify(csrContent));

    var activationCode = csrContent.activationCode;
    var response;

    // This is a place to perform validation of csrContent and update clientDN if required.
    // You can do it using adapter backend connectivity
    if (activationCode === "worklight"){
        response = {
            isSuccessful: true,
            clientDN: clientDN + ",CN=someCustomData",
            attributes: {
                customAttribute: "some-custom-attribute"
            }
        };
    } else {
        response = {
            isSuccessful: false,
            errors: ["Invalid activation code"]
        };
    }

    return response;
}
```

アダプター機能 (http Web サービスへのアクセスなど) を使用して、CSR 情報を検証できます。ここでは単純にするために、事前定義でハードコーディングされたストリングと activationCode が同じかどうかのみを検査します。

サーバー・サイド・コンポーネントの実装 (4/7)

- validateCSR (clientDN, csrContent) 関数を実装します。

```
function validateCSR(clientDN, csrContent){
    WL.Logger.info("validateCSR :: clientDN :: " + JSON.stringify(clientDN));
    WL.Logger.info("validateCSR :: csrContent :: " + JSON.stringify(csrContent));

    var activationCode = csrContent.activationCode;
    var response;

    // This is a place to perform validation of csrContent and update clientDN if required.
    // You can do it using adapter backend connectivity
    if (activationCode === "worklight"){
        response = {
            isSuccessful: true,
            clientDN: clientDN + ",CN=someCustomData",
            attributes: {
                customAttribute: "some-custom-attribute"
            }
        };
    } else {
        response = {
            isSuccessful: false,
            errors: ["Invalid activation code"]
        };
    }

    return response;
}
```

CSR 検証が成功した場合は、validateCSR 関数が clientDN を返します (注: カスタム・データで変更を加えることができます)。さらに、証明書に保存するカスタム属性を指定することもできます。validateCSR 関数から isSuccessful:true が返されると、Worklight Server が証明書を生成してアプリケーションに返します。

サーバー・サイド・コンポーネントの実装 (5/7)

- validateCSR (clientDN, csrContent) 関数を実装します。

```
function validateCSR(clientDN, csrContent){
    WL.Logger.info("validateCSR :: clientDN :: " + JSON.stringify(clientDN));
    WL.Logger.info("validateCSR :: csrContent :: " + JSON.stringify(csrContent));

    var activationCode = csrContent.activationCode;
    var response;

    // This is a place to perform validation of csrContent and update clientDN if required.
    // You can do it using adapter backend connectivity
    if (activationCode === "worklight"){
        response = {
            isSuccessful: true,
            clientDN: clientDN + ",CN=someCustomData",
            attributes: {
                customAttribute: "some-custom-attribute"
            }
        };
    } else {
        response = {
            isSuccessful: false,
            errors: ["Invalid activation code"]
        };
    }

    return response;
}
```

CSR の検証が失敗した場合は、`isSuccessful:false` を返してエラー・メッセージを表示する必要があります。

サーバー・サイド・コンポーネントの実装 (6/7)

- `validateCertificate (certificate, customAttributes)` 関数を実装します。

```
function validateCertificate(certificate,customAttributes){
    WL.Logger.info("validateCertificate :: certificate :: " + JSON.stringify(certificate));
    WL.Logger.info("validateCertificate :: customAttributes :: " + JSON.stringify(customAttributes));

    // Additional custom certificate validations can be performed here.

    return {
        isSuccessful: true
    };
}
```

ここでカスタム・ルールに従って、証明書の検証を実行できます。アダプター機能 (http Web サービスへのアクセスなど) を使用して、証明書を検証できます。証明書が有効な場合は、`isSuccessful:true` を返す必要があります。

サーバー・サイド・コンポーネントの実装 (7/7)

- `validateCertificate (certificate, customAttributes)` 関数を実装します。

```
function validateCertificate(certificate,customAttributes){
    WL.Logger.info("validateCertificate :: certificate :: " + JSON.stringify(certificate));
    WL.Logger.info("validateCertificate :: customAttributes :: " + JSON.stringify(customAttributes));

    // Additional custom certificate validations can be performed here.

    return {
        isSuccessful: true
    };
}
```

`isSuccessful:false` を返すことは、アプリケーションが作動できず、アプリケーションを再度プロビジョンするにはアプリケーションを再インストールする必要があることを意味します。

アジェンダ

- 概要
- カスタム・デバイス・プロビジョニングについて
- authenticationConfig.xml の構成
- サーバー・サイド・コンポーネントの実装
- クライアント・サイド・コンポーネントの実装
- 結果の確認

クライアント・サイド・コンポーネントの実装 (1/10)

- アプリケーションを作成し、iPhone/iPad/Android 環境をそのアプリケーションに追加します。
- 前のステップで作成したセキュリティー・テストを追加し、作成した環境を保護します。

```
<iphone bundleId="com.CustomProvisioningApp" securityTest="CustomDeviceProvisioningSecurityTest" version="1.0" >
    <worklightSettings include="true"/>
    <security>
        <encryptWebResources enabled="false"/>
        <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg, jpeg, gif, mp4, mp3"/>
    </security>
</iphone>
```

- 必要に応じて、*『アプリケーション認証性保護』*トレーニング・モジュールの説明に従ってアプリケーション認証性テスト用にアプリケーションを構成します。

クライアント・サイド・コンポーネントの実装 (2/10)

- アプリケーションの HTML ファイルを更新します。

```
<body style="display: none;">
    <div id="header">
        <h1>Custom Provisioning Application</h1>
    </div>

    <div id="wrapper">
        <div id="AppBody">
            <p id="beforeProv">
                Device authentication with custom device provisioning was not complete
                <button id="connectToServerButton" class="appButton">Connect to Worklight server</button>
            </p>
            <p id="provisioningError" style="display: none"></p>
        </div>

        <div id="ProvBody" style="display: none">
            <p id="provisioningError">
                <input id="provisioningCode" placeholder="Enter code" type="text" />
                <button id="submitProvCodeButton" class="formButton">Send</button>
            </p>
        </div>
    </div>
    <script src="js/initOptions.js"></script>
    <script src="js/main.js"></script>
    <script src="js/messages.js"></script>
    <script src="js/CustomDeviceProvisioningRealmChalleng
</body>
```

AppBody エレメントには、アプリケーション・コンテンツが保持されます。ProvBody エレメントには、デバイス・プロビジョニング関連のコンテンツが保持されます。AppBody 内の connectToServerButton に注意してください。

クライアント・サイド・コンポーネントの実装 (3/10)

- **connectToServerButton** にリスナーを追加します。
- `WL.Client.connect()` API を使用して Worklight Server に接続します。

```
function wlCommonInit(){
    $("#connectToServerButton").click(function(){
        WL.Client.connect();
    });
}
```

クライアント・サイド・コンポーネントの実装 (4/10)

- `CustomDeviceProvisioningRealmChallengeHandler.js` ファイルを追加して、これをメイン HTML ファイルで参照します。
- デバイス・プロビジョニング・チャレンジ・ハンドラーを使用するには、以下のメソッドを実装する必要があります。
  - `handler.createCustomCsr (challenge)` – このメソッドは、CSR に追加されるカスタム・プロパティーを返す役割があります。ここで、**activationCode** カスタム・プロパティーを追加します。このプロパティーは、前のスライドで説明したアダプターの `validateCSR` 関数で使用されます。このメソッドは、ネイティブ・コードまたは別のフローを介したカスタム・プロパティーの収集を可能にするため非同期になっています。
  - `handler.processSuccess(identity)` – このメソッドは、証明書の検証が、以前に実装した `validateCertificate` アダプター関数を使用して正常に完了した場合に呼び出されます。
  - `handler.handleFailure()` – このメソッドは、証明書の検証が失敗した (`isSuccessful:false` が `validateCertificate` 関数から返された) 場合に呼び出されます。

クライアント・サイド・コンポーネントの実装 (5/10)

- デバイス・プロビジョニング・チャレンジ・ハンドラーを実装します。

```
var customDevProvChallengeHandler =
    WL.Client.createProvisioningChallengeHandler("CustomDeviceProvisioningRealm");

customDevProvChallengeHandler.createCustomCsr = function(challenge){
    WL.Logger.debug("createCustomCsr :: " + JSON.stringify(challenge));

    $("#AppBody").hide();
    $("#ProvBody").show();
    $("#provisioningCode").val("");

    if (challenge.error) {
        $("#provisioningError").html(new Date() + " " + challenge.error);
    } else {
        $("#provisioningError").html(new Date() + " Enter activation code.");
    }

    $("#submitProvCodeButton").click(funct
        var customCsrProperties = {
            activationCode: $("#provisioni
        };
        customDevProvChallengeHandler.subm
    });
};
```

WL.Client.createProvisioningChallengeHandler() API を使用して、デバイス・プロビジョニング・チャレンジ・ハンドラーを作成します。レルム名をパラメーターとして指定します。

クライアント・サイド・コンポーネントの実装 (6/10)

- デバイス・プロビジョニング・チャレンジ・ハンドラーを実装します。

```
var customDevProvChallengeHandler =
    WL.Client.createProvisioningChallengeHandler("CustomDeviceProvisioningRealm");

customDevProvChallengeHandler.createCustomCsr = function(challenge){
    WL.Logger.debug("createCustomCsr :: " + JSON.stringify(challenge));

    $("#AppBody").hide();
    $("#ProvBody").show();
    $("#provisioningCode").val("");

    if (challenge.error) {
        $("#provisioningError").html(new Date() + " " + challenge.error);
    } else {
        $("#provisioningError").html(new Date() + " Enter activation code.");
    }

    $("#submitProvCodeButton").click(funct
        var customCsrProperties = {
            activationCode: $("#provisioni
        };
        customDevProvChallengeHandler.subm
    });
};
```

Worklight Server がデバイス・プロビジョニングをトリガーすると、`createCustomCsr` 関数が呼び出されます。この関数を使用して、UI の操作 (アプリケーション画面の非表示、デバイス・プロビジョニング関連コンポーネントの表示など) を行います。

クライアント・サイド・コンポーネントの実装 (7/10)

- デバイス・プロビジョニング・チャレンジ・ハンドラーを実装します。

```
var customDevProvChallengeHandler =
    WL.Client.createProvisioningChallengeHa

customDevProvChallengeHandler.createCustomC
    WL.Logger.debug("createCustomCsr :: " +

    $("#AppBody").hide();
    $("#ProvBody").show();
    $("#provisioningCode").val("");

    if (challenge.error) {
        $("#provisioningError").html(new Date() + " " + challenge.error);
    } else {
        $("#provisioningError").html(new Date() + " Enter activation code.");
    }

    $("#submitProvCodeButton").click(function(){
        var customCsrProperties = {
            activationCode: $("#provisioningCode").val()
        };
        customDevProvChallengeHandler.submitCustomCsr(customCsrProperties, challenge);
    });
};
```

エラー・メッセージなど、認証チャレンジで返された情報を使用できます。

クライアント・サイド・コンポーネントの実装 (8/10)

- デバイス・プロビジョニング・チャレンジ・ハンドラーを実装します。

```
var customDevProvChallengeHandler =
    WL.Client.createProvisioningChallengeHand

customDevProvChallengeHandler.createCustomCsr
    WL.Logger.debug("createCustomCsr :: " + J

    $("#AppBody").hide();
    $("#ProvBody").show();
    $("#provisioningCode").val("");

    if (challenge.error) {
        $("#provisioningError").html(new Date() + " " + challenge.error);
    } else {
        $("#provisioningError").html(new Date() + " Enter activation code.");
    }

    $("#submitProvCodeButton").click(function(){
        var customCsrProperties = {
            activationCode: $("#provisioningCode").val()
        };
        customDevProvChallengeHandler.submitCustomCsr(customCsrProperties, challenge);
    });
};
```

必要なカスタム・プロパティーが収集されたら、`submitCustomCsr()` API を呼び出します。CSR にカスタム・プロパティーを追加することはオプションです。
カスタム・プロパティーを追加したくない場合は、空の JSON オブジェクトをパラメーターとして指定します。

クライアント・サイド・コンポーネントの実装 (9/10)

- デバイス・プロビジョニング・チャレンジ・ハンドラーを実装します。

```
customDevProvChallengeHandler.processSuccess = function(identity) {
    WL.Logger.debug("processSuccess :: " + JSON.stringify(identity));
    $("#connectToServerButton").hide();
    $("#AppBody").show();
    $("#ProvBody").hide();
    $("#wrapper").text("Device authentication with custom device provisioning "+
            "was successfully complete");
};

customDevProvChallengeHandler.handleFailure = function(){
    WL.Logger.debug("handleFailure");
    $("#AppBody").show();
    $("#ProvBody").hide();
    $("#wrapper").text("Server has rejec
            "reinstall the application a
};
```

`processSuccess` 関数は、証明書が正常に検証を通過するたびに呼び出されます。この関数を使用して UI を操作できます。

クライアント・サイド・コンポーネントの実装 (10/10)

- デバイス・プロビジョニング・チャレンジ・ハンドラーを実装します。

```
customDevProvChallengeHandler.processSuc
    WL.Logger.debug("processSuccess :: "
    $("#connectToServerButton").hide();
    $("#AppBody").show();
    $("#ProvBody").hide();
    $("#wrapper").text("Device authentic
            "was successfully complete")
};

customDevProvChallengeHandler.handleFailure = function(){
    WL.Logger.debug("handleFailure");
    $("#AppBody").show();
    $("#ProvBody").hide();
    $("#wrapper").text("Server has rejected your device. You will need to "+
            "reinstall the application and perform device provisioning again.");
};
```

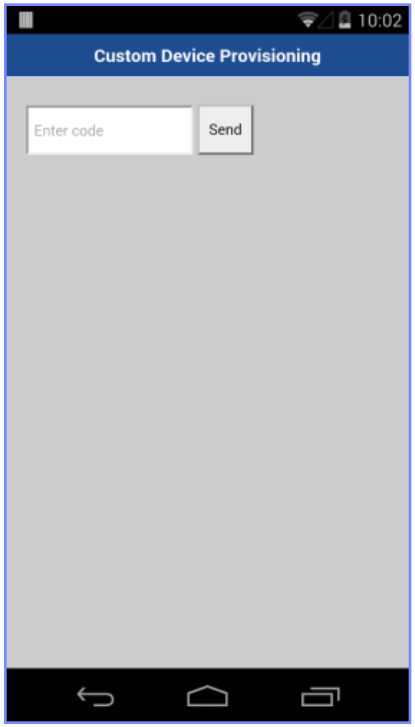
`handleFailure` 関数は、証明書が検証に失敗するたびに呼び出されます。この関数を使用して UI を操作したり、アプリケーションが Worklight Server に接続できないことをユーザーに通知したりできます。

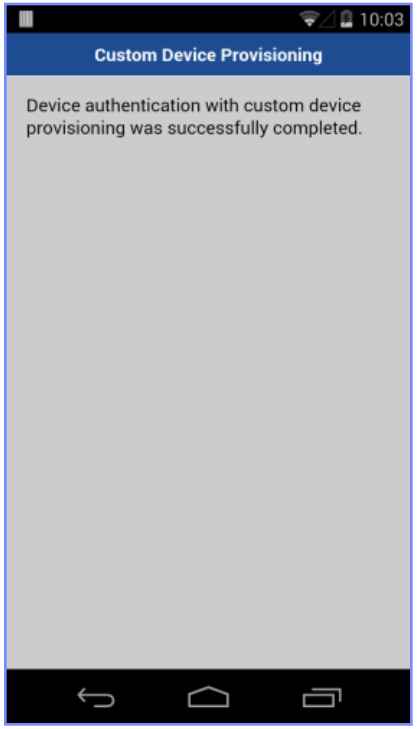
アジェンダ

- 概要
- カスタム・デバイス・プロビジョニングについて
- authenticationConfig.xml の構成
- サーバー・サイド・コンポーネントの実装
- クライアント・サイド・コンポーネントの実装
- 結果の確認

結果の確認

- 結果の確認。





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