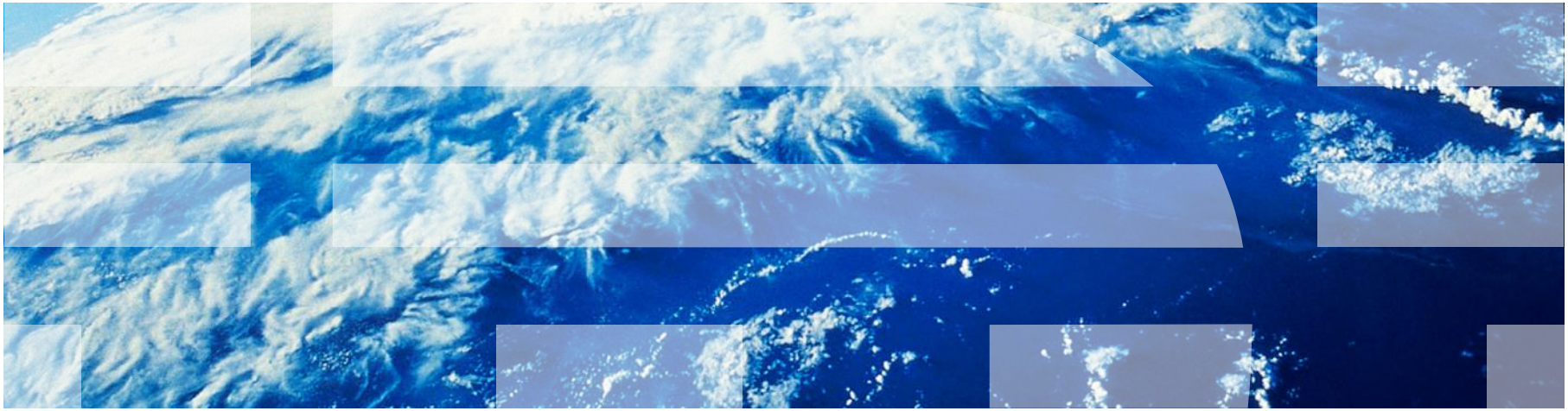
IBM Worklight Foundation V6.2.0
Getting Started

Custom device provisioning



20 June 2014

Trademarks

- IBM, the IBM logo, ibm.com, and Worklight are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. 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

- Overview
- Understanding custom device provisioning
- Configuring authenticationConfig.xml
- Implementing server-side components
- Implementing client-side components
- Examining the result

Overview

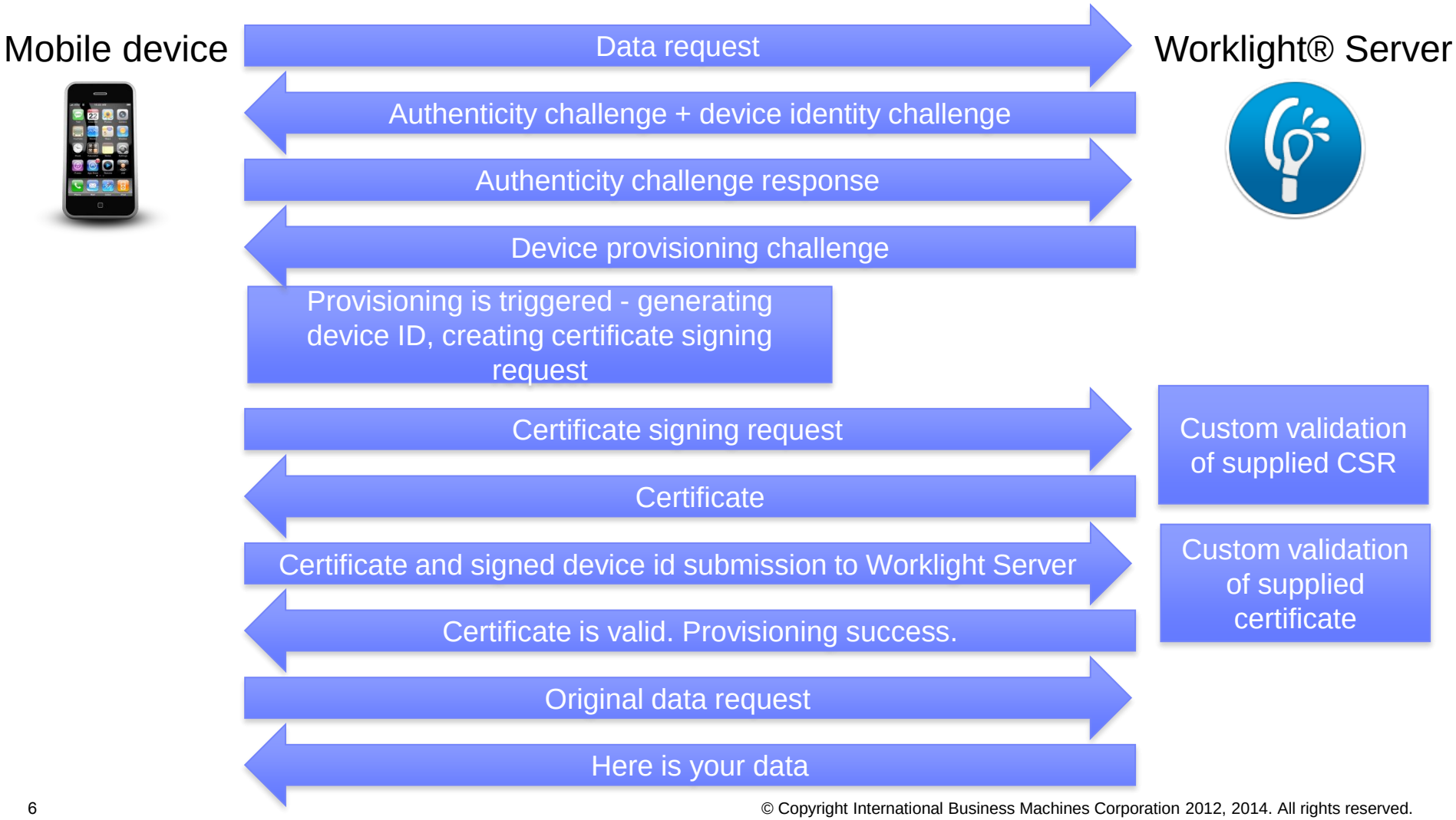
- In this training module, you learn how to enable and configure custom device provisioning
- Custom device provisioning is an extension of auto device provisioning, which you can implement custom validations of:
  - Certificate Signing Request during initial provisioning flow.
  - Certificate during every application start.
- It is vital to gain a solid understanding of the topics that are discussed in the *Device Provisioning Concepts* training module because this training module is fully based on them.

Agenda

- Overview
- Understanding custom device provisioning
- Configuring authenticationConfig.xml
- Implementing server-side components
- Implementing client-side components
- Examining the result

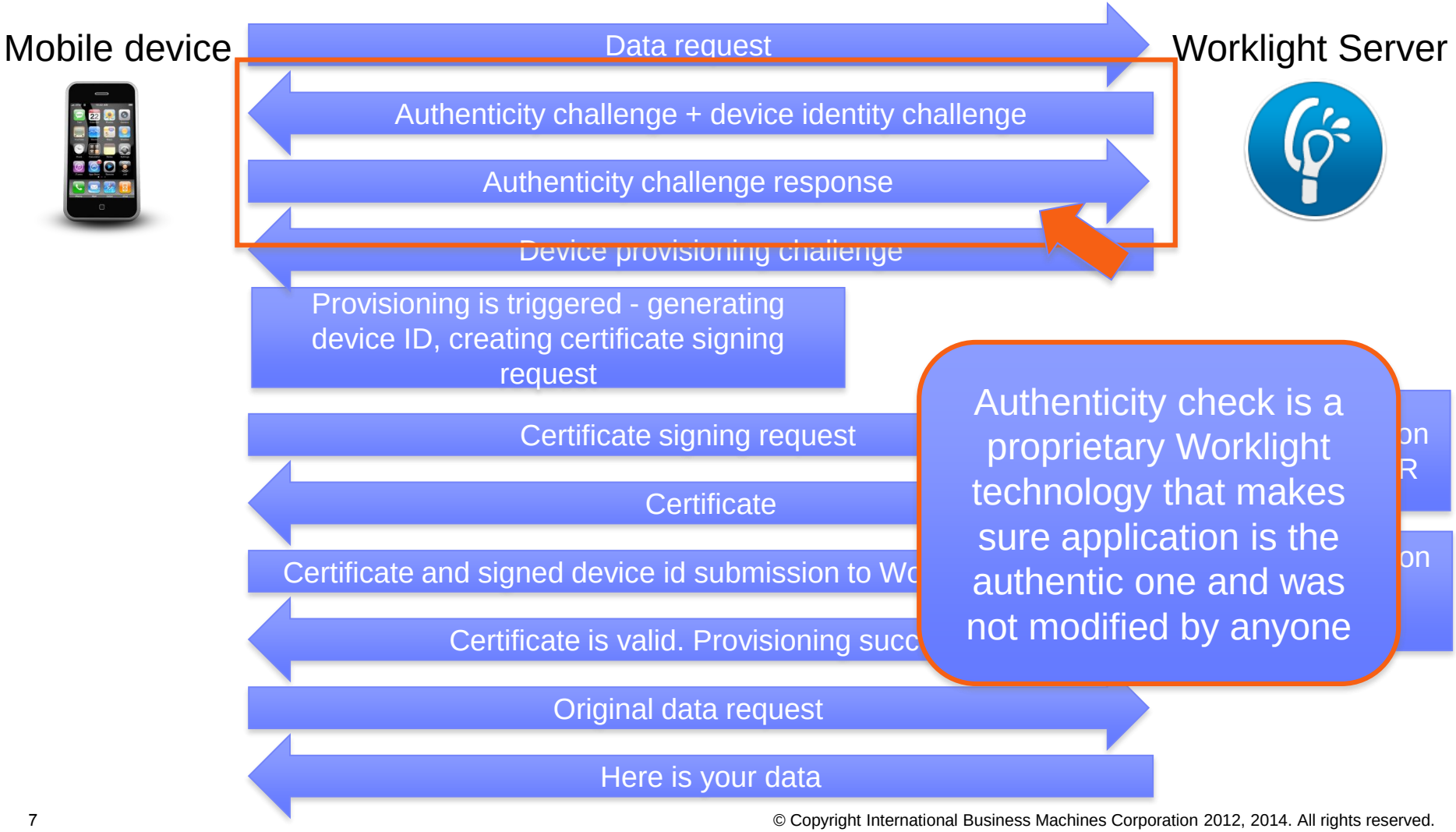
Understanding custom device provisioning (1 of 5)

- Custom device provisioning flow – first application start.



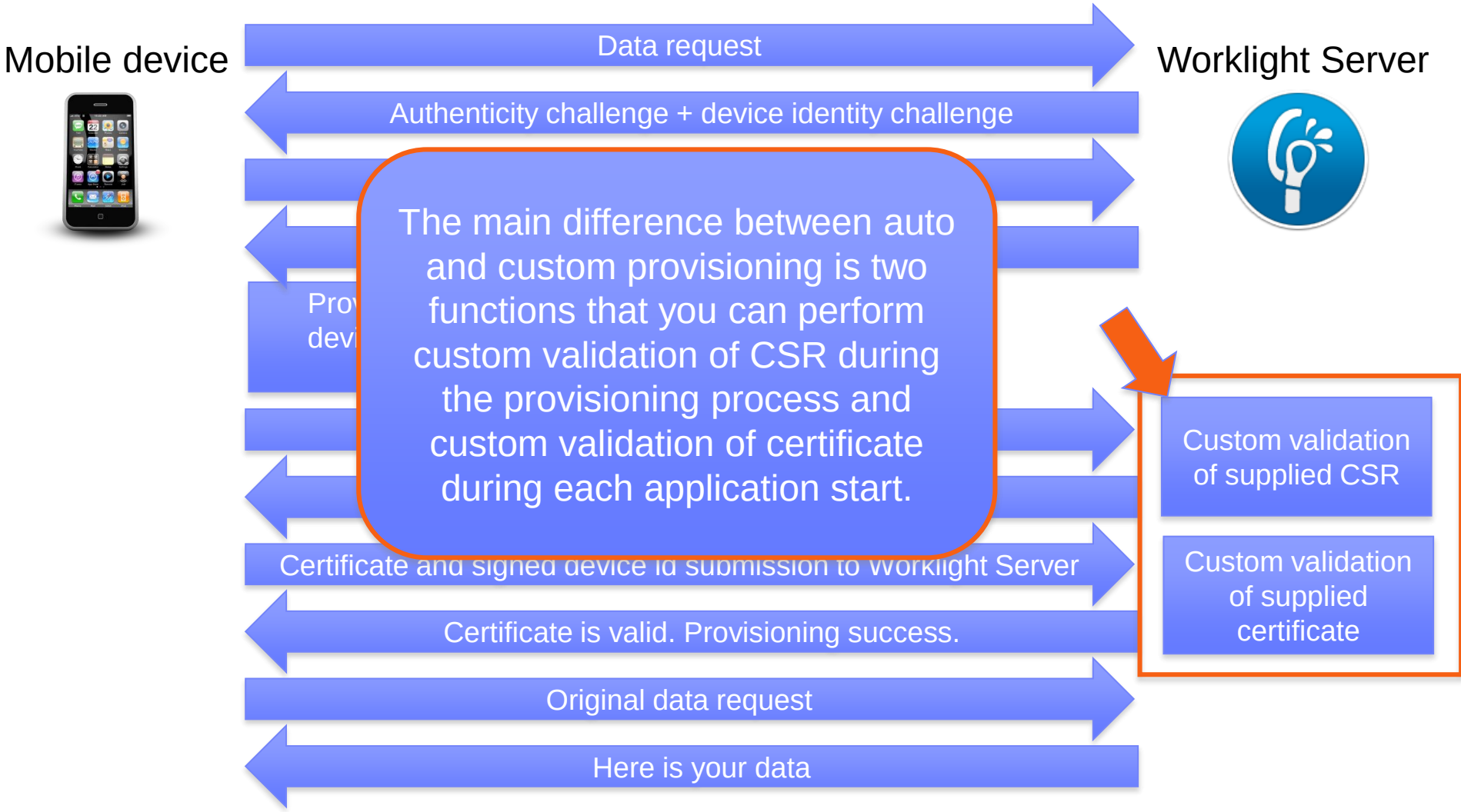
Understanding custom device provisioning (2 of 5)

- Custom device provisioning flow – first application start.



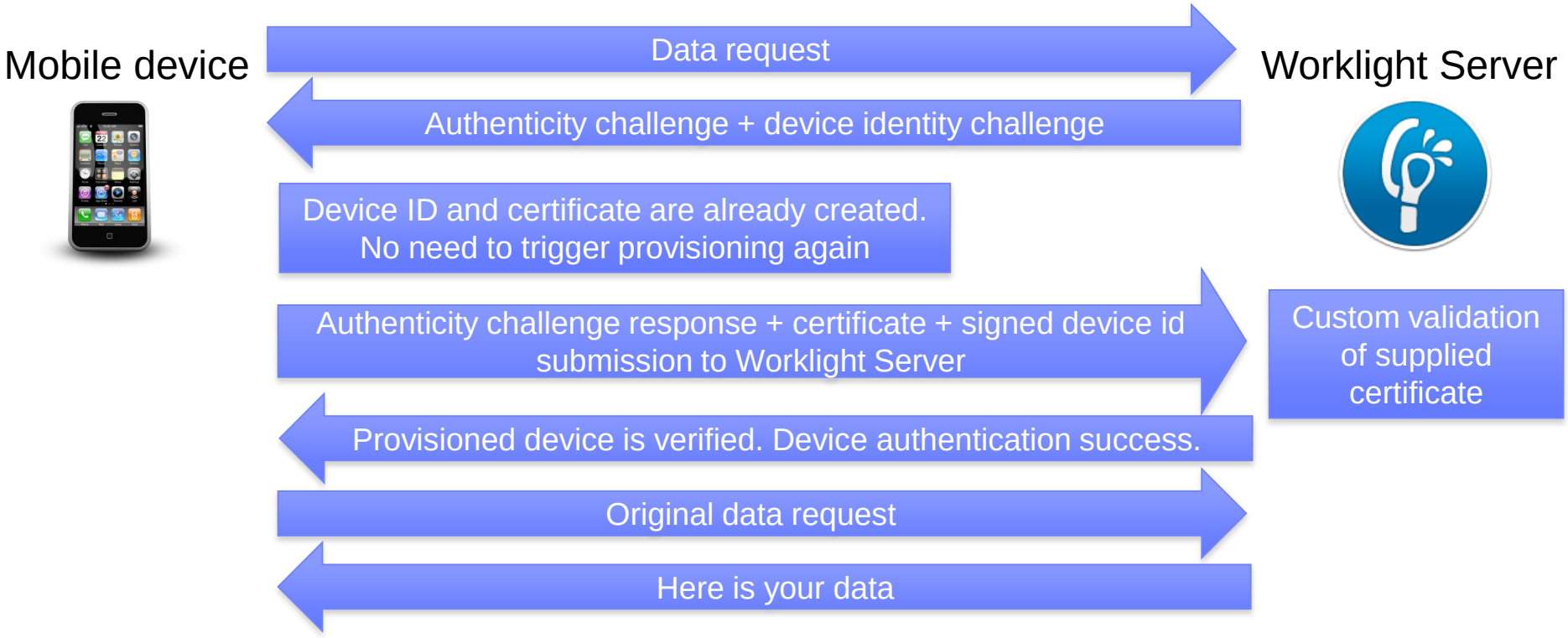
Understanding custom device provisioning (3 of 5)

- Custom device provisioning flow – first application start.



Understanding custom device provisioning (4 of 5)

- Custom device provisioning flow – subsequent application starts.



Understanding custom device provisioning (5 of 5)

- By default, the Worklight Server uses its internal keystore to issue a certificate.
- You can tell the Worklight Server to use your own keystore by adjusting the `worklight.properties` file.

```
##########################################################################################################################
#    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
```

- **Note:** The `wl.ca.keystore.path` property value can be either relative to the `/server/` folder of Worklight project or absolute to the file system.

Agenda

- Overview
- Understanding custom device provisioning
- Configuring authenticationConfig.xml
- Implementing server-side components
- Implementing client-side components
- Examining the result

Configuring authenticationConfig.xml (1 of 3)

- Start by adding a realm that is named **CustomDeviceProvisioningRealm** to the `authenticationConfig.xml` file.
- Use **CustomDeviceProvisioningLoginModule.**
- Use the auto provisioning authenticator className parameter.
- Add a **validate-csr-function** parameter.
- The value of this parameter points to an Adapter function that performs CSR validation.

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

Configuring authenticationConfig.xml (2 of 3)

- Add **CustomDeviceProvisioningLoginModule.**
- Use the auto provisioning login module className parameter.
- Add a **validate-certificate-function** parameter.
- The value of this parameter points to an Adapter function that performs certificate validation.

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

Configuring authenticationConfig.xml (3 of 3)

- Create a **mobileSecurityTest**.
- Add a mandatory `<testAppAuthenticity/>` test.
- Add a mandatory `<testDeviceId/>` test.
- Specify `provisioningType="custom".`
- Specify `realm="CustomDeviceProvisioningRealm".`

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

Agenda

- Overview
- Understanding custom device provisioning
- Configuring authenticationConfig.xml
- Implementing server-side components
- Implementing client-side components
- Examining the result

Implementing server-side components (1 of 7)

- Create an adapter that is named **ProvisioningAdapter**.
- Add two functions with following signatures to the JavaScript™ file of the adapter.
  - `validateCSR (clientDN, csrContent)` – this function is invoked only during initial device provisioning. It is used to check whether the device is authorized to be provisioned. Once the device is provisioned, this function is not invoked again.
  - `validateCertificate (certificate, customAttributes)` – this function is invoked every time that the mobile application establishes a new session with the Worklight server. It is used to validate that the certificate that the application/device possesses is still valid and that the application/device is allowed to communicate with Worklight Server.
- These functions are called internally by the Worklight authentication framework. Therefore, do not declare them in the XML file of the adapter XML file.

Implementing server-side components (2 of 7)

- Implement `validateCSR (clientDN, csrContent)` function.

```
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` is a custom property that you add to CSR on the client side.

Implementing server-side components (3 of 7)

- Implement `validateCSR (clientDN, csrContent)` function.

```
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;
}
```

Adapter functionality, for example access http web services, can be used to validate CSR information. For simplicity, the `activationCode` is checked whether it is equal to a predefined hardcoded string.

Implementing server-side components (4 of 7)

- Implement `validateCSR (clientDN, csrContent)` function.

```
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;
}
```

If CSR validation is successful, the `validateCSR` function returns a `clientDN` (note that it can be modified with more custom data). In addition, it is possible to specify custom **attributes** to be saved in certificate. Once `isSuccessful:true` is returned from the `validateCSR` function, the Worklight server generates a certificate and return it to the application.

Implementing server-side components (5 of 7)

- Implement `validateCSR (clientDN, csrContent)` function.

```
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;
}
```

If CSR validation fails, you must return `isSuccessful:false` and supply an error message.

Implementing server-side components (6 of 7)

- Implement `validateCertificate (certificate, customAttributes)` function.

```
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
    };
}
```

You can perform certificate validations according to your custom rules here. Adapter functionality, for example access http web services, can be used to validate the certificate. If the certificate is valid, you must return `isSuccessful:true`.

Implementing server-side components (7 of 7)

- Implement `validateCertificate (certificate, customAttributes)` function

```
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
    };
}
```

Returning `isSuccessful:false` means that application cannot operate and the only thing that can be done is to reinstall the application so it can be provisioned again.

Agenda

- Overview
- Understanding custom device provisioning
- Configuring authenticationConfig.xml
- Implementing server-side components
- Implementing client-side components
- Examining the result

Implementing client-side components (1 of 10)

- Create an application, add iPhone/iPad/Android environment to it.
- Add security test that is created in previous steps to protect created environment.

```
<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>
```

- In case it is required, configure your application for Application Authenticity test as described in the *Application Authenticity Protection* training module.

Implementing client-side components (2 of 10)

- Update application HTML file.

```
<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` element holds application content. `ProvBody` element holds device provisioning-related content. Note the `connectToServerButton` in `AppBody`

Implementing client-side components (3 of 10)

- Add listener to **connectToServerButton**
- Use `WL.Client.connect()` API to connect to the Worklight Server

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

Implementing client-side components (4 of 10)

- Add a `CustomDeviceProvisioningRealmChallengeHandler.js` file and reference it in the main HTML file.
- Device provisioning challenge handler requires following methods to be implemented.
  - `handler.createCustomCsr (challenge)` – This method is responsible for returning custom properties that are added to CSR. Here you add a custom **activationCode** property, which is used in the adapter's `validateCSR` function in previous slides. This method is asynchronous to allow collecting custom properties via native code or separate flow.
  - `handler.processSuccess(identity)` – This method is invoked when certificate validation is successfully completed by using the `validateCertificate` adapter function that you implemented earlier.
  - `handler.handleFailure()` – This method is invoked when certificate validation fails (`isSuccessful:false` is returned from `validateCertificate` function).

Implementing client-side components (5 of 10)

- Implement device provisioning challenge handler.

```
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
    });
};
```

Create device provisioning challenge handler by using the `WL.Client.createProvisioningChallengeHandler()` API. Specify realm name as parameter.

Implementing client-side components (6 of 10)

- Implement device provisioning challenge handler.

```
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
    });
};
```

When Worklight Server triggers device provisioning, the `createCustomCsr` function is invoked. Use it to manipulate your UI, for example to hide the application screen and show device provisioning-related components.

Implementing client-side components (7 of 10)

- Implement device provisioning challenge handler.

```
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);
    });
};
```

You can use information that is returned in authentication challenge, for example, error messages.

Implementing client-side components (8 of 10)

- Implement device provisioning challenge handler.

```
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);
    });
};
```

When required custom properties are collected, invoke the `submitCustomCsr()` API. Adding custom properties to CSR is optional. If you do not want to add custom properties, supply empty JSON object as a parameter.

Implementing client-side components (9 of 10)

- Implement device provisioning challenge handler.

```
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` function is called each time the certificate successfully passes validation. You can use it for UI manipulations.

Implementing client-side components (10 of 10)

- Implement device provisioning challenge handler.

```
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.");
};
```

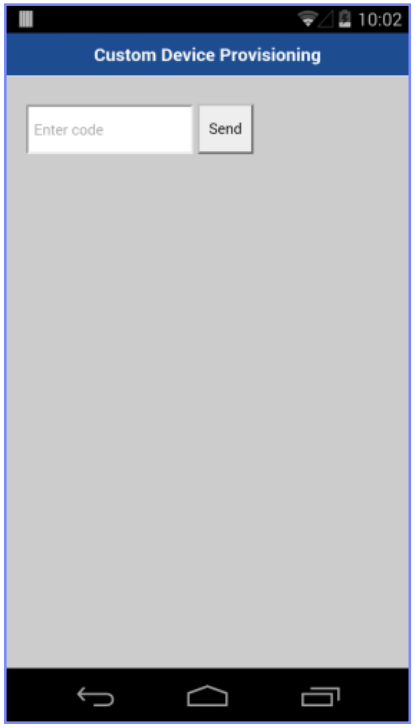
`handleFailure` function is called each time that the certificate fails validation. You can use it for UI manipulations and to notify the user that the application cannot connect to Worklight Server.

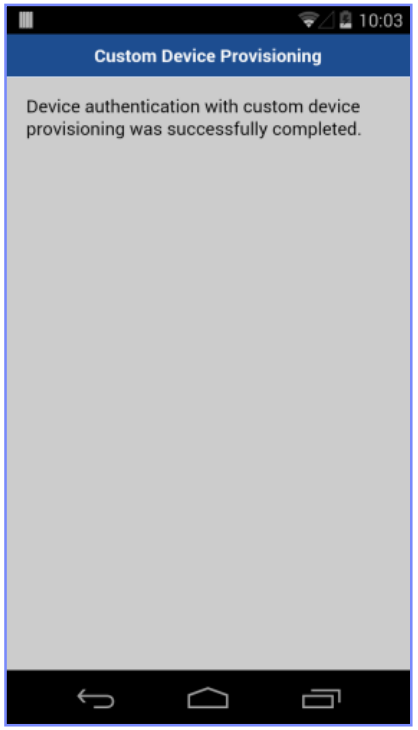
Agenda

- Overview
- Understanding custom device provisioning
- Configuring authenticationConfig.xml
- Implementing server-side components
- Implementing client-side components
- Examining the result

Examining the result

- Examining the result.





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