

Course Exercises Guide

# Web Services Support in IBM DataPower V7.5

Course code WE754 / ZE754 ERC 1.0



## May 2017 edition

### Notices

This information was developed for products and services offered in the US.

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, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

### Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., 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](http://www.ibm.com/legal/copytrade.shtml).

© Copyright International Business Machines Corporation 2017.

**This document may not be reproduced in whole or in part without the prior written permission of IBM.**

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Trademarks</b> .....	<b>iv</b>
<b>Exercises description</b> .....	<b>v</b>
<b>Exercise 1. Web service encryption and digital signatures</b> .....	<b>1-1</b>
1.1. Initialize the lab environment .....	1-5
1.2. Create cryptographic objects .....	1-6
1.3. Import the BookingServiceProxy service .....	1-9
1.4. Configure BookingServiceProxy to verify a signed message .....	1-11
1.5. Test the BookingServiceProxy signature verification .....	1-15
1.6. Configure BookingServiceProxy to sign a message .....	1-18
1.7. Test that the BookingServiceProxy sends a signed message .....	1-22
1.8. Configure BookingServiceProxy to decrypt a message .....	1-24
1.9. Test the BookingServiceProxy message decryption .....	1-26
1.10. Configure BookingServiceProxy to encrypt the response message .....	1-27
1.11. Test the BookingServiceProxy message encryption .....	1-29
1.12. Examine the rule processing by using the multi-step probe .....	1-32
<b>Exercise 2. Configuring a web service proxy</b> .....	<b>2-1</b>
2.1. Create a web service proxy for the Booking Service web service .....	2-5
2.2. Verify the generated components .....	2-9
2.3. Test the web service proxy processing .....	2-14
2.4. Add an operation-level rule to BookingServiceWSPProxy .....	2-16
2.5. Test the web service proxy processing with a Transform rule .....	2-18
2.6. Configure message decryption on the web service proxy .....	2-19
2.7. Test the web service proxy processing with decryption added .....	2-21
2.8. Use the probe to observe the encrypted request message processing .....	2-23
<b>Appendix A. Exercise solutions</b> .....	<b>A-1</b>
Part 1: Dependencies .....	A-1
Part 2: Importing solutions .....	A-2
<b>Appendix B. Lab environment setup</b> .....	<b>B-1</b>
Part 1: Configure the SoapUI variables for use .....	B-1
Part 2: Confirm that the Booking and Baggage web services are up .....	B-3
Part 3: Identify the student image IP address .....	B-6
Part 4: Port and variable table values .....	B-8

---

# Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

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

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DataPower®

DB™

DB2®

IMS™

Notes®

Rational®

Tivoli®

WebSphere®

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the “Marks”) of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

# Exercises description

## FLY airline case study

The exercises in this course build upon a common case study: the FLY airline services. The services are composed of a Booking Service web service and a Baggage Service web service. The services are implemented as a BookingServiceBackend MPGW and a BaggageStatusMockService MPGW, both running within the FLYService domain.

The Booking Service has one operation: BookTravel. The SOAP request that is named BookingRequest contains billing details, payment card details, booking type, and the reservation code. The SOAP response is a BookingResponse, which contains the confirmation code and much of the original message. The endpoint is:

`http://<dp_internal_ip>:9080/BookingService/`

The Baggage Service has two operations:

- **BaggageStatus.** The SOAP request that is named BaggageStatusRequest contains the passenger's last name and reference number. The SOAP response is BaggageStatusResponse, which contains the status of each bag that is attached to the passenger's reference number.
- **BagInfo.** The SOAP request that is named BagInfoRequest contains the ID number of the bag in question. The SOAP response is BagInfoResponse, which contains the status of the bag and which passenger it belongs to. The Baggage Service does not have a WSDL, and cannot be proxied by using a web service proxy. The endpoint is:

`http://<dp_internal_ip>:2068/BaggageService/`

Technically, the FLY airline services are self-contained MPGWs that mimic a web services back end that might be on WebSphere Application Server.

This application minimizes its dependencies on data sources by relying on data from a flat file, and allowance of read-only operations.

### Exercises

This course includes the following exercises:

- **Exercise 1:** Web service encryption and digital signatures

Create cryptographic objects. Import a starting BookingServiceProxy MPGW. Configure the MPGW to verify a signed request message. Test it with SoapUI. Update the MPGW to sign a response message. Test it with SoapUI. Update the MPGW to decrypt a request message. Test it with SoapUI. Update the MPGW to encrypt a response message. Test it with SoapUI.

- **Exercise 2:** Create a web service proxy for the Booking Service web service

Create a web service proxy from the Booking Service WSDL. Test it with SoapUI. Add an operation-level request Transform action to a response rule to transform the response message. Test it with SoapUI. Update the WS-Proxy to decrypt a message at the service level. Test it with SoapUI.

**Note**

The lab exercises were written on DataPower V7.5.1.3 firmware. A consistent problem at this level is a failure of the “save configuration” operation in the WebGUI. If this failure happens to you, a workaround exists. Instead of clicking “save configuration”, click “review changes” instead. Scroll to the bottom of the Review Configuration Changes page and click **Save Config**.

In the exercise instructions, you see that each step has a blank preceding it. You might want to check off each step as you complete it to track your progress.

Most exercises include required sections, which must always be completed. These exercises might be required before doing later exercises. Some exercises also include optional sections that you might want to do when you have sufficient time and want an extra challenge.

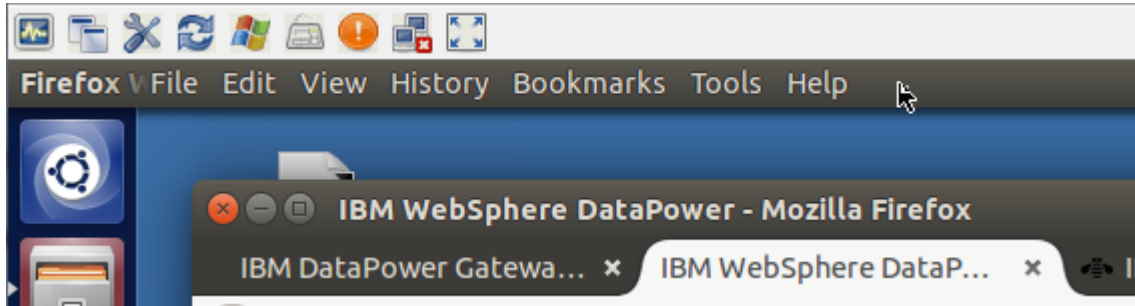
## If you are using the IBM remote lab environment:

As of September 2016, the environment that is used to support the IBM-supplied images and DataPower gateways is Skytap. Each student is supplied an Ubuntu student image and a DataPower gateway.

- Ignore all offers to upgrade any of the software on the image. The image and exercise steps are designed to operate at the supplied levels of the contained software.
- The supplied image is Ubuntu 14.04 LTS. The desktop uses Unity, which is different from the more common Gnome desktop. Some hints on using Unity are at:  
<http://www.howtogeek.com/113330/how-to-master-ubuntus-unity-desktop-8-things-you-need-to-know/>
- A noticeable difference is that the menus on the windows within the desktop are not typically visible. When a window is the “active” window, that window does not have any menu items, but the application type is displayed in the black bar that spans the top of the desktop. In the following screen capture, observe that the browser window does not have a menu bar, and that its type of “Firefox Web Browser” is listed in the black bar.



If you hover the mouse over the black bar, the menu items for the active window are displayed.

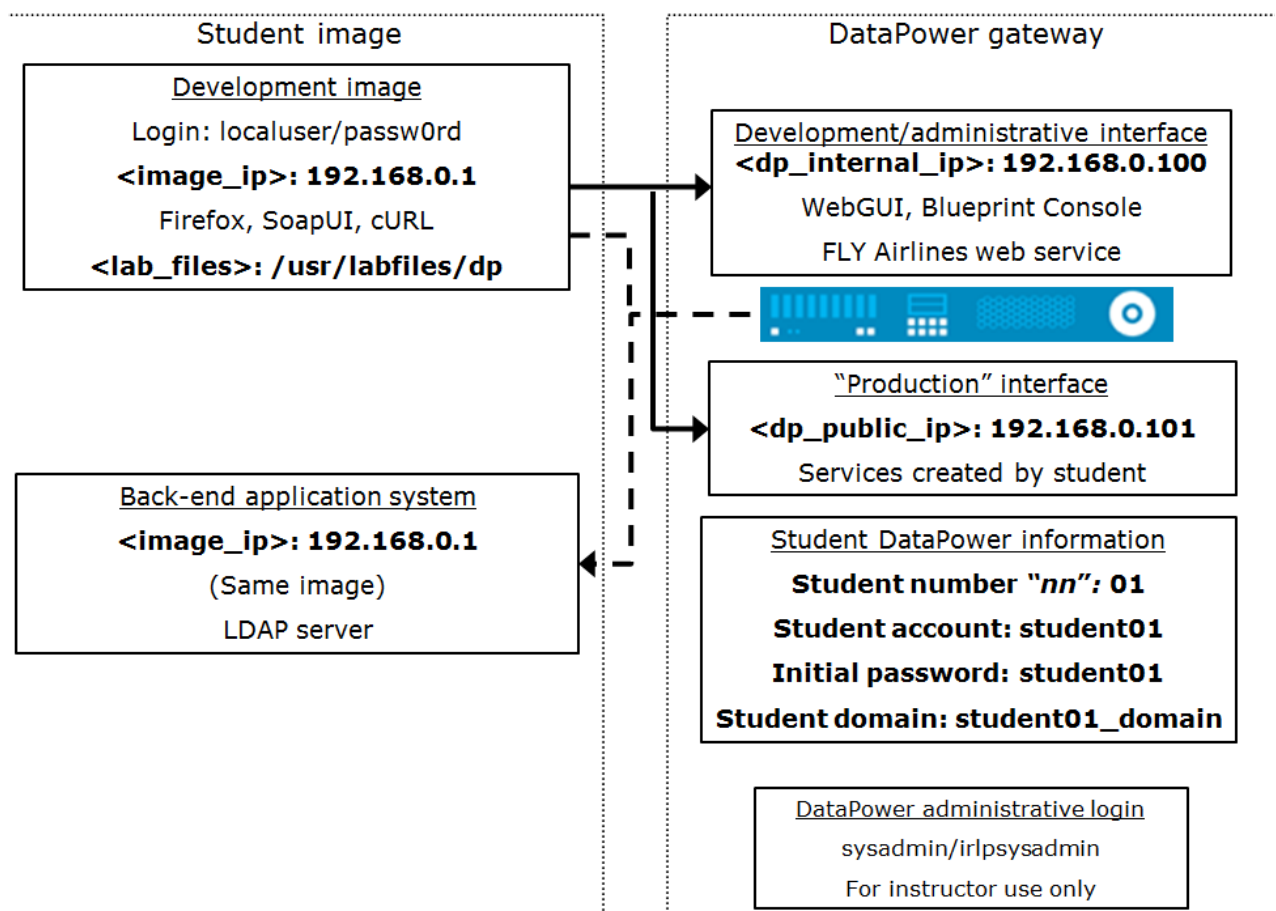


Another noticeable difference is that when a window is maximized, the Close, Minimize, and Restore buttons are not visible until you hover the mouse in the black bar.

This Unity behavior is discussed in the “Hidden Global Menus” section of the previously mentioned [howtogeek.com](http://howtogeek.com) page.

- The IBM supplied environment has values that are pre-assigned for some of the variables, such as the IP addresses of the student image and the DataPower gateway, the student number, and the initial password. The following graphic shows those assignments.

## Variable values for DataPower courses on IBM/Skytap



**Important**

Online course material updates might exist for this course. To check for updates, see the Instructor wiki at <http://ibm.biz/CloudEduCourses>.

---



---

# Exercise 1. Web service encryption and digital signatures

## Estimated time

01:00

## Overview

In this exercise, you learn how to perform web services security functions by using the IBM DataPower Gateway. The DataPower gateway supports security-related tasks that both a client and a server need to perform. You use SoapUI to send signed and encrypted messages to the MPGW. The MPGW decrypts and verifies the message. On the response, the MPGW signs and encrypts the message before it returns it to SoapUI.

## Objectives

After completing this exercise, you should be able to:

- Configure a multi-protocol gateway to decrypt and encrypt an XML message
- Configure a multi-protocol gateway to verify and sign an XML message
- Test encryption and digital signatures by using the SoapUI tool

## Introduction

In this exercise, you configure a request and response rule that is used for various cryptographic processes.

The steps in the exercise are:

- Section 1: Initialize the lab environment – The student configures the test environment.
- Section 2: Create cryptographic objects – The student creates the DataPower crypto objects that are needed in later steps.
- Section 3: Import the BookingServiceProxy service – Import the BookingServiceProxy MPGW and update its variables.
- Section 4: Configure BookingServiceProxy to verify a signed message – A request rule is created that validates a signed message.
- Section 5: Test the BookingServiceProxy signature verification – A signed message is sent from SoapUI to test the new validation rule.
- Section 6: Configure BookingServiceProxy to sign a message – The student creates a response rule to sign the response message that is being returned to SoapUI.

- Section 7: Test that the BookingServiceProxy sends a signed message – The student verifies that a signed message is received from DataPower.
- Section 8: Configure the BookingServiceProxy to decrypt a message – The new request rule is modified to decrypt an encrypted message before the signature is validated.
- Section 9: Test the BookingServiceProxy message decryption – A signed and encrypted message is sent from SoapUI. The student verifies that the message is decrypted and validated before the message is sent to the back end.
- Section 10: Configure BookingServiceProxy to encrypt the response message – The response rule is modified to encrypt the message after it is signed.
- Section 11: Test the BookingServiceProxy message encryption – The student uses SoapUI to verify that a signed encrypted message was sent back from DataPower.
- Section 12: Examine the rule processing by using the multi-step probe – The student uses the probe to examine the message state as the message contents are passed between the actions in the request and response rules.

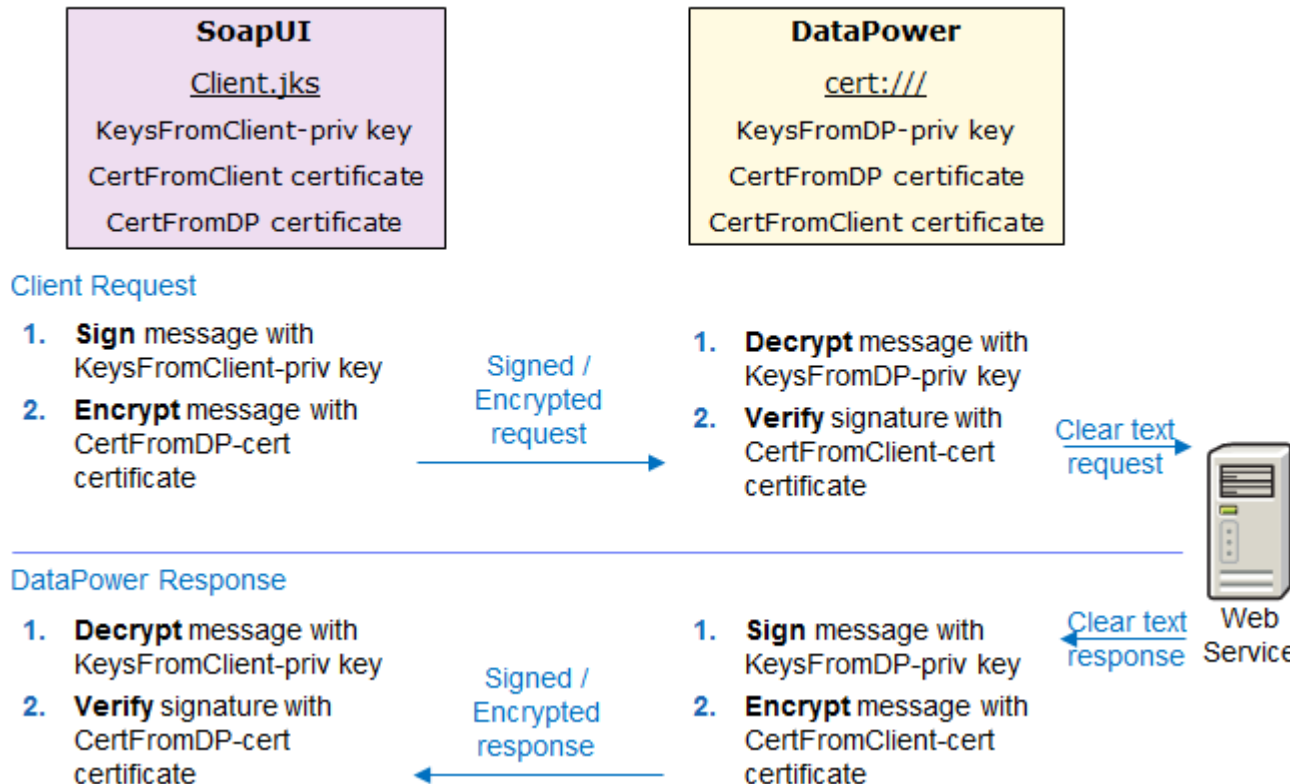
It is important to understand the key material setup in that each end of the communication does not have the private key of the other end. SoapUI has the only visibility to its private key in the Client.jks keystore. DataPower has the only visibility to its private key in its cert: directory. This arrangement mimics the real world environments.

A key-certificate pair was generated on DataPower. The certificate file was also imported into SoapUI's Client.jks. Java keytool was used to generate a key-pair in Client.jks. That certificate file is uploaded to the DataPower cert: directory.

The following graphic shows the DataPower objects and files that are in the cert: directory, and the contents of SoapUI's Client.jks keystore. It also summarizes the message processing that is configured in DataPower by the end of the exercise.

The terms “certificate” and “public key” are used interchangeably.

## Completed exercise



## Requirements

To complete this exercise, you need:

- Access to the DataPower gateway
- SoapUI to send requests to the DataPower gateway
- Client.jks, configured with SoapUI
- The BookingServiceBackend web service that runs on the DataPower gateway in the FLYServices domain
- Access to the `<lab_files>` directory

## Exercise instructions

### Preface

- Remember to use the domain and port address that you were assigned in the exercise setup. **Do not** use the default domain.
- The references in exercise instructions refer to the following values:
  - `<lab_files>`: Location of the student lab files. Default location is: `/usr/labfiles/dp/`
  - `<image_ip>`: IP address of the student image (use `/sbin/ifconfig` from a terminal window to obtain value).
  - `<dp_internal_ip>`: IP address of the DataPower gateway development and administrative functions that are used by internal resources such as developers.
  - `<dp_public_ip>`: IP address of the public services on the gateway that is used by customer and clients.
  - `<dp_WebGUI_port>`: Port of the WebGUI. The default port is 9090.
  - `<nn>`: Assigned student number. If no instructor exists, use “01”.
  - `<studentnn>`: Assigned user name and user account. If no instructor exists, use “student01”.
  - `<studentnn_password>`: Account password. In most cases, the initial value is the same as the user name. You are prompted to create a password on first use. Write it down.
  - `<studentnn_domain>`: Application domain that the user account is assigned to. If no instructor exists, use “student01\_domain”.
  - `<FLY_booking_port>`: Port number that the back-end BookingServices web services listen on. The default port is 9080.
  - `<mpgw_booking_port>`: `12nn1`, where “nn” is the two-digit student number. This number is the listener port for the MPGW that proxies the BookingService web service.

## 1.1. Initialize the lab environment

Some setup activities are required to properly configure the lab environment and determine IP addresses and ports.

- \_\_\_ 1. If you did not yet perform the setup activities, you must go to **Appendix B: Lab environment setup**. Complete those activities before proceeding.

These activities need to be done only once for the course.

## 1.2. Create cryptographic objects

In this section, you create the key and certificate objects to do the security-related tasks in this course. In particular, for this exercise, they are used to encrypt and decrypt messages. Two pairs of keys and certificates are used. One pair is from the client, and the other pair is from DataPower. The objects and keys are named to assist with the key and certificate origination.

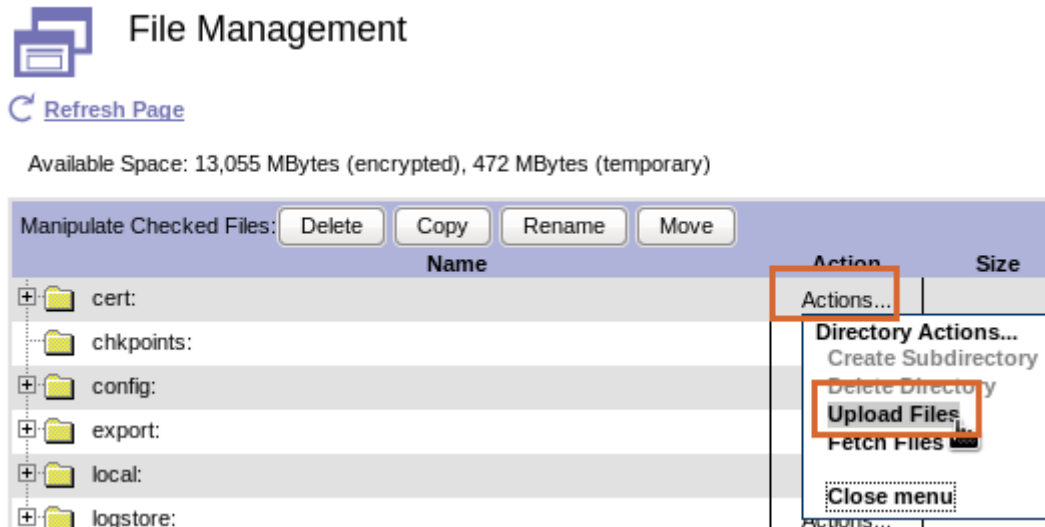
The SoapUI tool is configured with the Client.jks (Java keystore) file. This file was prepopulated with the **CertFromDP** certificate, the **KeysFromClient** private key, and the **CertFromClient** certificate.

You are uploading and configuring DataPower with the **KeysFromDP** private key, **CertFromDP** certificate, and the **CertFromClient** certificate.



- \_\_\_ 1. Log in to the IBM DataPower Gateway WebGUI.
  - \_\_\_ a. Connect to the DataPower gateway:  
`https://<dp_internal_ip>:<dp_WebGUI_port>`
  - \_\_\_ b. Log in with your student account <studentnn> and password <studentnn\_password>, and select your student domain <studentnn\_domain>.
- \_\_\_ 2. Upload the key and certificate files to DataPower.
  - \_\_\_ a. Click the **File Management** icon.
  - \_\_\_ b. Click the **Actions** link for the **cert:** directory to open the Directory Actions window.
  - \_\_\_ c. Click **Upload Files**.

- \_\_\_ d. Upload the certificate file:  
 <lab\_files>/WSecurity/KeysFromClient-sscert.pem



- \_\_\_ e. Repeat the steps to upload the private key file:  
 <lab\_files>/WSecurity/KeysFromDP-privkey.pem
- \_\_\_ f. Repeat the steps to upload the certificate file:  
 <lab\_files>/WSecurity/KeysFromDP-sscert.pem



### Note

Notice that you do not import the client private key. The client must keep its private key private! The DataPower should have no knowledge of that key.

- \_\_\_ 3. Create a crypto key object called `KeysFromDP`.
- \_\_\_ a. Click **Objects > Crypto Configuration > Crypto Key** (or use the **Search** field).
- \_\_\_ b. Click **Add** to create a crypto key object.
- \_\_\_ c. On the Configure Crypto Key page, enter the following information, and leave the remaining fields with their default values.
- o **Name:** `KeysFromDP`
  - o **File name:** `cert:/// KeysFromDP-privkey.pem`

Name

---

Administrative state ☒ enabled ☐ disabled

File Name

- \_\_\_ d. Click **Apply**.
- \_\_\_ e. Verify that the `KeysFromDP` object operational state is **up**.



### Information

The `KeysFromDP` object provides a reference to the private key file that the DataPower gateway uses to decrypt messages that were encrypted by using the associated certificate. This associated certificate is set up in the next step. A private key is also used to sign messages.

- \_\_\_ 4. Create a certificate object called `CertFromDP`.
  - \_\_\_ a. Click **Objects > Crypto Configuration > Crypto Certificate**.
  - \_\_\_ b. Click **Add** to create a crypto certificate object.
  - \_\_\_ c. On the Configure Crypto Certificate page, enter the name: `CertFromDP`  
 Select the `KeysFromDP-sscert.pem` file from the `cert:` directory. Leave the remaining fields with their default values.
  - \_\_\_ d. Click **Apply**.
  - \_\_\_ e. Verify that the `CertFromDP` operational status is **up**.



### Information

The `CertFromDP` object provides a reference to the certificate that is used to encrypt and verify signed messages that are sent to the DataPower gateway.

- \_\_\_ 5. Create a certificate object called `CertFromClient`.
  - \_\_\_ a. Click **Objects > Crypto Configuration > Crypto Certificate**.
  - \_\_\_ b. Click **Add** to create a crypto certificate object.
  - \_\_\_ c. On the Configure Crypto Certificate page, enter the name: `CertFromClient`  
 Select the `KeysFromClient-sscert.pem` file from the `cert:` directory, and use that file for the **File Name** field. Leave the remaining fields with their default values.
  - \_\_\_ d. Click **Apply**.
  - \_\_\_ e. Verify that the `CertFromClient` operational status is **up**.



### Information

The `CertFromClient` object provides a reference to the certificate that is used to verify a message that the client signed by using its own private key.

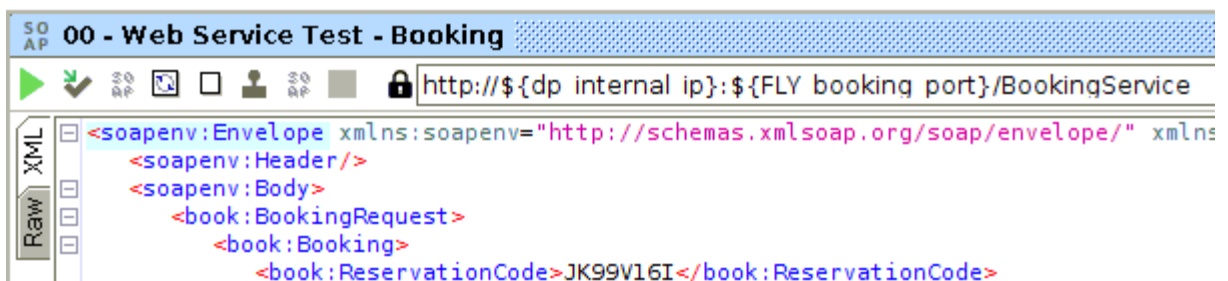
- \_\_\_ f. Click **Save Configuration**.



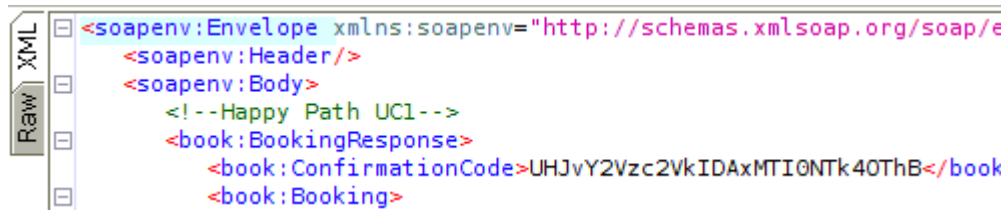
## 1.3. Import the BookingServiceProxy service

If you do not have the BookingServiceProxy MPGW already defined in your domain from the prerequisite Essentials course, then you need to import this service as your starting point.

- \_\_\_ 1. Verify that the host aliases are defined. The imported service references the `dp_internal_ip` alias.
  - \_\_\_ a. Switch to the default domain. You should have “read” access to this domain.
  - \_\_\_ b. Open **Network > Interface > Host Alias**.
  - \_\_\_ c. Verify that the **dp\_internal\_ip** alias refers to the `<dp_internal_ip>` address.
  - \_\_\_ d. Verify that the **dp\_public\_ip** alias refers to the `<dp_public_ip>` address.
  - \_\_\_ e. Switch back to your student domain.
- \_\_\_ 2. Use **Administration > Configuration > Import Configuration** to import `<lab_files>/WSecurity/BookingServiceProxyExport.zip` into your domain.
- \_\_\_ 3. Update the MPGW to use your student variables.
  - \_\_\_ a. Edit the BookingServiceProxy MPGW.
  - \_\_\_ b. Verify that the **Default Backend URL** points to `http://dp_internal_ip:9080/BookingService`. This URL is the Booking Service web service back end that runs in the FLYServices domain.
  - \_\_\_ c. Edit the **HTTP\_12nn1** front side handler.
  - \_\_\_ d. Set the **Local IP Address** to: `<dp_public_ip>`
  - \_\_\_ e. Set the **Port** to: `<mpgw_booking_port>`
  - \_\_\_ f. Click **Apply** to save the front side handler configuration.
  - \_\_\_ g. If necessary, click **Apply** to save the MPGW configuration.
  - \_\_\_ h. Save the configuration.
- \_\_\_ 4. Use SoapUI to test the service.
  - \_\_\_ a. Open SoapUI.
  - \_\_\_ b. Expand **BookingServices > BookingServicesSOAP > BookTravel**. Double-click **Web Service Test – Booking**.
  - \_\_\_ c. Observe that it is an HTTP POST request (since it is a web services request) that passes a SOAP payload in the HTTP body.



- \_\_\_ d. Click the green Submit arrow.
- \_\_\_ e. The panel repaints and the response pane gets larger. The **XML** tab should be selected. The SOAP response for the booking is displayed.



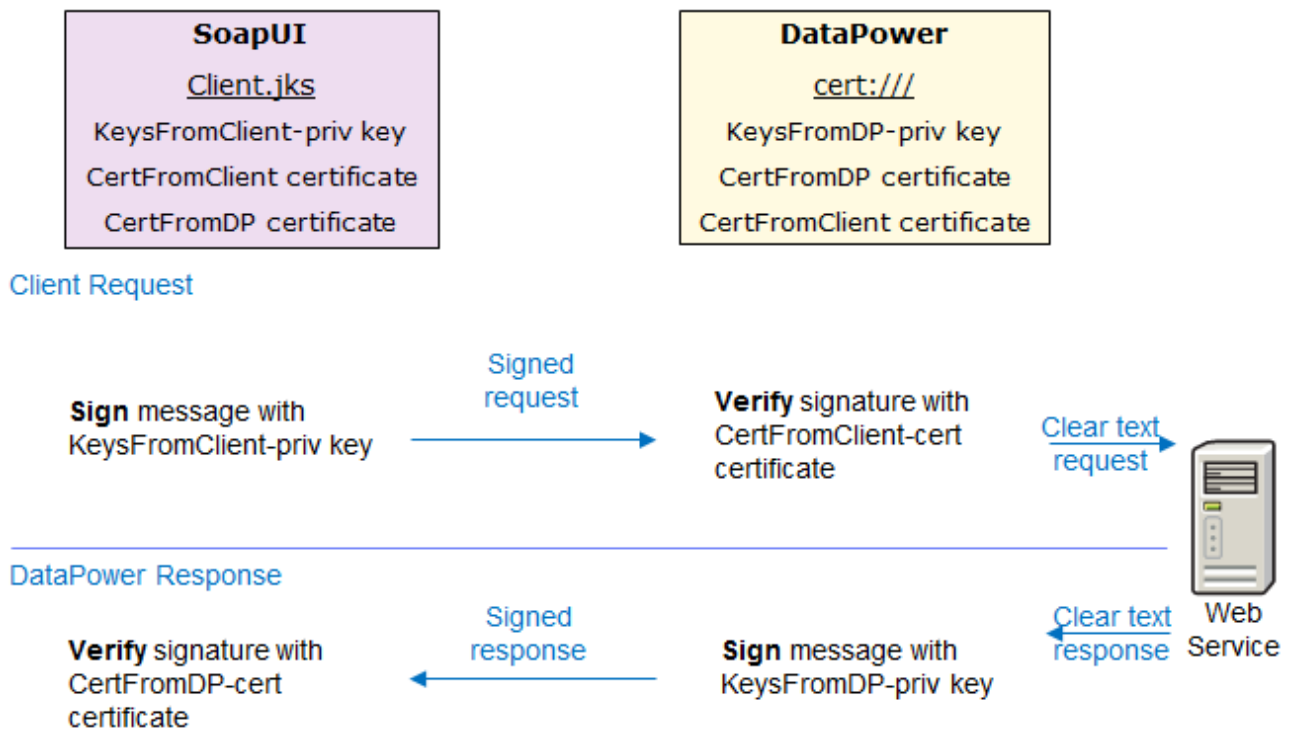
```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <!--Happy Path UC1-->
    <book:BookingResponse>
      <book:ConfirmationCode>UHJvY2Vzc2VkdAxtMTI0NTk4OThB</book:ConfirmationCode>
    </book:BookingResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

- \_\_\_ f. The base MPGW is set up. Close the “Web Service Test – Booking” pane. You can leave SoapUI open for later use.

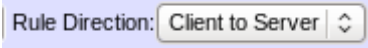
## 1.4. Configure BookingServiceProxy to verify a signed message

The client (SoapUI) and DataPower are configured to sign and verify messages. The client is signing the incoming message and DataPower verifies the signature. Then, DataPower signs the outgoing message and the client verifies it. This process is built within the next few sections.

### Sign and verify steps



- \_\_\_ 1. On the **Control Panel**, click the **Multi-Protocol Gateway** icon.
- \_\_\_ 2. On the "Configure Multi-protocol Gateway catalog list" page, click **BookingServiceProxy**.
- \_\_\_ 3. On the Configure Multi-protocol Gateway page, edit the **BookingServicePolicy** policy.
- \_\_\_ 4. Several rules exist. Click **New Rule** to add a rule.
- \_\_\_ 5. Set the Rule Direction to **Client to Server**.



- \_\_\_ 6. Double-click the **Match** action for configuration.
  - \_\_\_ a. Click new (+) to create a matching rule definition.
  - \_\_\_ b. Enter `Sign` as the name for the matching rule.
  - \_\_\_ c. Click **Add** to define the rule.
  - \_\_\_ d. Select **URL** as the **Matching type**.

- \_\_\_ e. Enter `/BookingService/sign` as the **URL match**. The URL match is case-sensitive.

Matching type  \*

URL match

- \_\_\_ f. Click **Apply** to save the rule condition.
- \_\_\_ g. Click **Apply** to save the matching rule and **Done** to save the Match action.
- \_\_\_ 7. Drag a **Verify** action after the **Match** action on the rule.
- \_\_\_ 8. Double-click the **Verify** action for configuration.
- \_\_\_ a. Leave the Standard at the default value **XML Security**.
- \_\_\_ b. You need a validation credential for this action. Click new (+) to create a Validation Credential.

**Verify**

**Standard** ☒ XML Security ☐ JSON Web Security \*

**Asynchronous** ☐ on ☒ off

**Signature Verification Type**  ☐

**Optional Signer Certificate**

**Validation Credential**   ...

- \_\_\_ c. Enter `CertFromClient` as the **Validation Credential** name.
- \_\_\_ d. Using the drop-down box, select **CertFromClient** as the **Certificates**.
- \_\_\_ e. Click **add** to add the Certificate to the validation credential.

**Name**

**Administrative state** ☒ enabled ☐ disabled

**Certificates**    ...

**Certificate Validation Mode**

- \_\_\_ f. Click **Apply**.

- \_\_\_ g. Confirm that the **Signature Verification Type** is RSA/DSA Signatures, and the **Validation Credential** is CertFromClient.

Signature Verification Type: RSA/DSA Signatures ☐ Save

Optional Signer Certificate:  ☐ Save

Validation Credential: CertFromClient   ☒ Save

- \_\_\_ h. Click **Done**.

- \_\_\_ 9. After the signature is verified, the WS-Security headers in the message are no longer needed. DataPower provides a stylesheet to strip the signature headers. To use this behavior, drag a **Transform** action after the Verify action on the rule.
- \_\_\_ 10. Double-click the Transform action for configuration.
- \_\_\_ a. Select **Transform with XSLT style sheet** as the **Document Processing Instructions**.
- \_\_\_ b. Select **store:///** as the **Transform File** directory.
- \_\_\_ c. Select **strip-wssec-signature.xsl** as the file.

Transform with XSLT style

Use Document Processing Instructions

☐ Transform binary

☐ Transform with a processing control

☐ Transform with embedded processing

☒ Transform with XSLT style sheet

Transform File

store:/>

strip-wssec-signature.xsl

URL Rewrite Policy

(none)

- \_\_\_ d. Click **Done**.

- \_\_\_ 11. This rule is at the end of the processing order of all the configured rules. Move this newly created request rule above the existing request rule that is configured to accept all incoming traffic. This move allows the **/sign** traffic to match to the appropriate rule.



















### Important

In most cases, the **name** of the rule is not important. It is the contents of the rule and the order it appears in the Configured Rules section. The names that you see in the following screen capture are the DataPower generated names; your names might differ. The order of the rules determines the order in which the Match actions are evaluated.

The only time that the rule name is important is when you use a **Call Processing Rule** action that invokes another rule in the service policy by its name.

- \_\_\_ a. Click the **Up arrow** key in the Order column for the Validate rule to move up the rule in the list of the Configured Rules. Repeat this process until the rule with the Verify action is above the existing request rule.

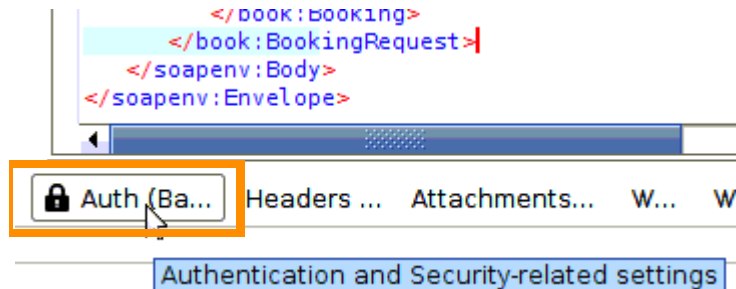
Configured Rules				
Order	Rule Name	Direction		
 	BookingServicePolicy_rule_2	Client to Server		 
 	BookingServicePolicy_rule_0	Client to Server		  
 	BookingServicePolicy_rule_1	Server to Client		 

- \_\_\_ b. Click **Apply Policy** to save the service policy and close the policy editor window.
- \_\_\_ 12. If necessary, click **Apply** at the MPGW level.

## 1.5. Test the BookingServiceProxy signature verification

Send a signed message from SoapUI to the BookingServiceProxy to see whether it can verify the signature.

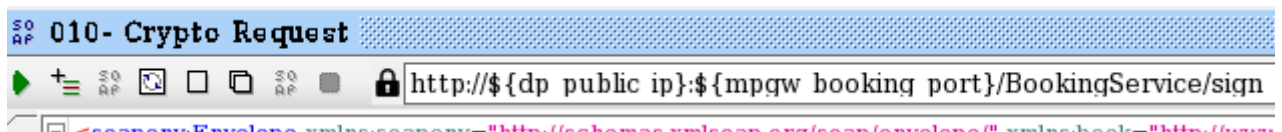
- \_\_\_ 1. If required, Open SoapUI by clicking the **SoapUI** icon on the desktop.
- \_\_\_ 2. In SoapUI, expand the BookingServices tree and open **10 – Crypto Request**.
- \_\_\_ 3. Click the **Auth** tab that is in the bottom portion of the 10 – Crypto Request.



- \_\_\_ 4. An Authorization section appears in the request tab. In this section, set the **Outgoing WSS** to **Sign**.
- \_\_\_ 5. Set the **Incoming WSS** to a “blank” entry.

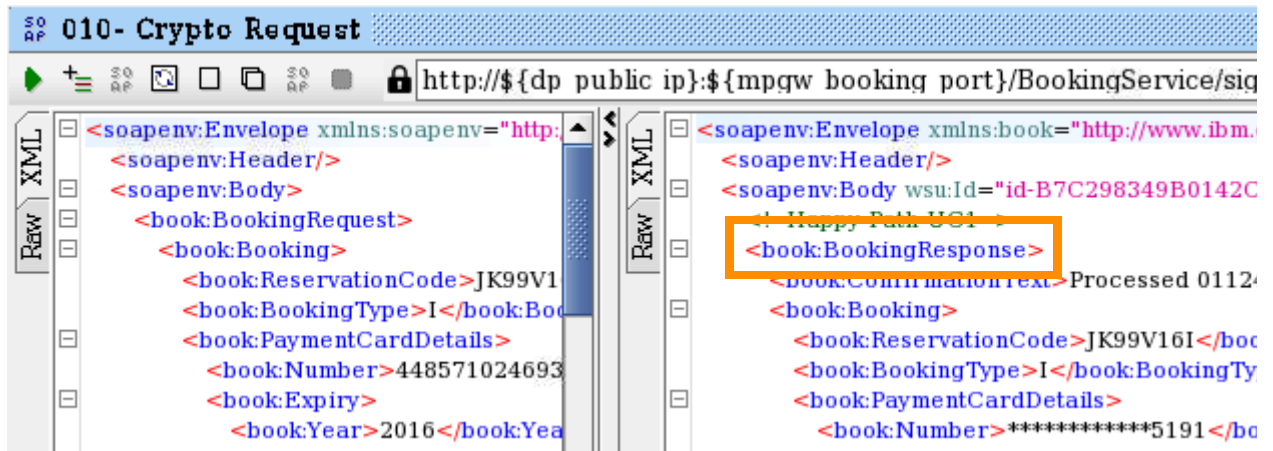


- \_\_\_ 6. Ensure that the endpoint URL appears as:  
`http://${dp_public_ip}:${mpgw_booking_port}/BookingService/sign`



- \_\_\_ 7. Click the green **Submit** arrow to POST the message to BookingServiceProxy. The request should return a successful booking reservation.

- \_\_ 8. Confirm that the <book:BookingResponse> is received.



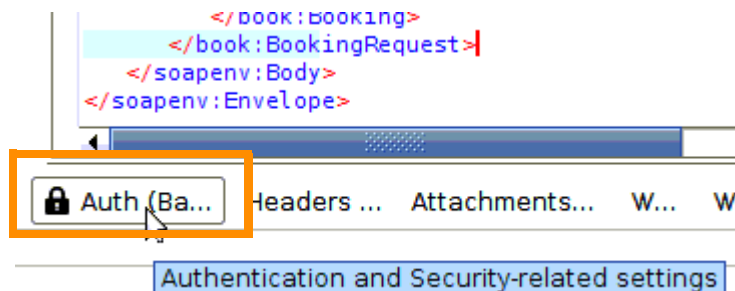
Because the Validate action in the DataPower rule verified the signature that SoapUI sent, the request was sent to the back end.



## Troubleshooting

If the verification fails on the DataPower side, check the system log to see whether the failure is due to an expired signature. If so, check the times on the gateway and on the Ubuntu image. If they are not close, then you must reset the time on one of them and retest.

- \_\_ 9. Right-click inside the response pane and select **Select All**.
- \_\_ 10. Right-click again and select **Clear**. This action empties the response pane so that you do not get confused when you do repetitive requests. You should use this technique throughout the exercise.
- \_\_ 11. Send a non-signed message that confirms the DataPower configuration of accepting only correctly signed messages.
- \_\_ a. Click the **Auth** tab that is in the bottom portion of the 10 – Crypto Request.

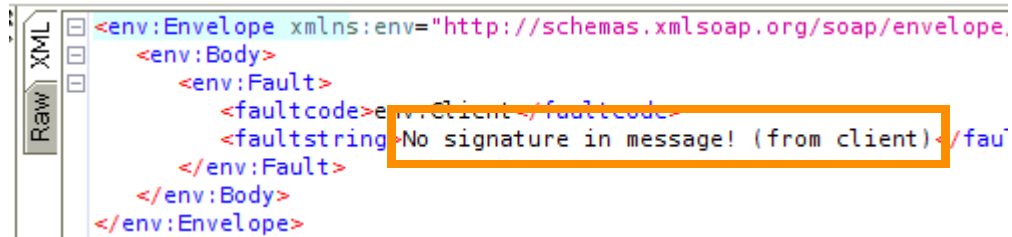


- \_\_ 12. Set the **Outgoing WSS** to have no value.

- \_\_ 13. Click the green **Submit** arrow to POST the message to BookingServiceProxy.



- \_\_\_ 14. Confirm that an error message is received with a status of No signature in message.



```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope">
  <env:Body>
    <env:Fault>
      <faultcode>env:Client</faultcode>
      <faultstring>No signature in message! (from client)</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

- \_\_\_ 15. Set the **Outgoing WSS** back to **Sign**.

Outgoing WSS:	<input type="text" value="Sign"/>
Incoming WSS:	<input type="text"/>

## 1.6. Configure BookingServiceProxy to sign a message


In this section, you create a response rule that signs the message that is being returned to the client.

- \_\_\_ 1. In the **BookingServicePolicy** policy editor, click **New Rule**.
- \_\_\_ 2. Set the Rule Direction to **Server to Client**.



- \_\_\_ 3. Double-click the **Match** action for configuration.
  - \_\_\_ a. Select the existing **Sign** matching rule.
  - \_\_\_ b. Click **Done**.
- \_\_\_ 4. Drag a **Sign** action after the **Match** action on the rule.
- \_\_\_ 5. Double-click the **Sign** action for configuration.
  - \_\_\_ a. Select **KeysFromDP** as the Key.

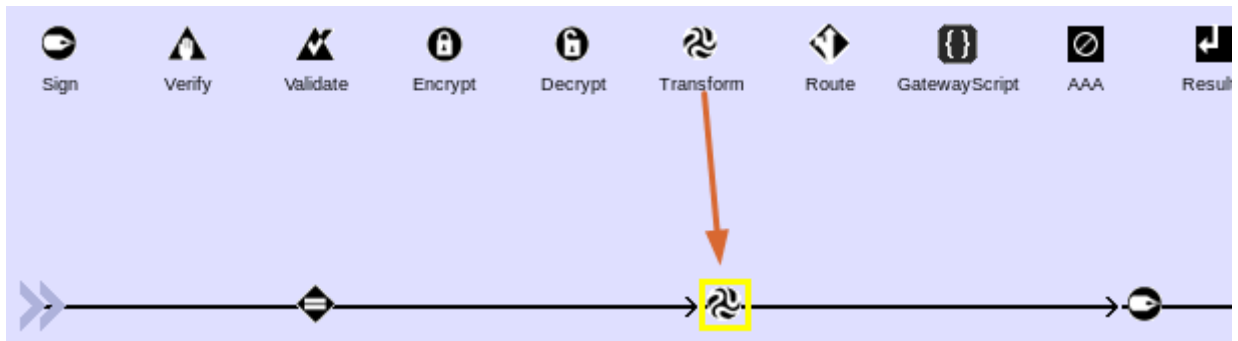
- \_\_\_ b. Select **CertFromDP** as the Certificate.

 **Sign**

<b>Standard</b>	<input checked="" type="radio"/> XML Security <input type="radio"/> JSON Web Security *
<b>Envelope Method</b>	<input type="radio"/> Enveloped Method <input type="radio"/> Enveloping Method <input type="radio"/> SOAPSec Method <input checked="" type="radio"/> WSSec Method <input type="radio"/> Advanced *
<b>Message Type</b>	<input checked="" type="radio"/> SOAP Message <input type="radio"/> SOAP With Attachments <input type="radio"/> Raw XML Document, including SAML <input type="radio"/> Selected Elements (Field-Level) <input type="radio"/> Advanced
<b>Asynchronous</b>	<input type="radio"/> on <input checked="" type="radio"/> off
<b>Use Asymmetric Key</b>	<input checked="" type="radio"/> on <input type="radio"/> off <input type="checkbox"/> Save
<b>Signing algorithm</b>	rsa <input type="button" value="Save"/>
<b>Key</b>	KeysFromDP <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/>
<b>Certificate</b>	CertFromDP <input type="button" value="+"/> <input type="button" value="..."/> <input type="checkbox"/>
<b>WS-Security Version</b>	1.0 <input type="button" value="Save"/>

- \_\_\_ c. Leave the remaining fields at their default values.
- \_\_\_ d. Click **Done**.

- \_\_\_ 6. Before you sign the message, you want to modify the response message as you did before. Drag a **Transform** action after the **Match** action and before the **Sign** action on the rule. This stylesheet works with the Expiry, CVV, ConfirmationCode, and Number elements in the response. You can view the stylesheet to see the behavior.



- \_\_\_ 7. Configure the **Transform** action.
- \_\_\_ a. Double-click the **Transform** action for configuration.
  - \_\_\_ b. Select **local:///** as the transform file type.
  - \_\_\_ c. Select **BookingResponse\_Transform.xsl** as the transform file.

This file might already be in the local subdirectory. If the file is not in the local subdirectory, upload the file from the `<lab_files>/WSecurity` directory.



### Information

In the successful response that you received earlier, all of the request details were included in the response, and the `<book: ConfirmationCode>` was base64 encoded.

The `BookingResponse_Transform.xsl` template looks for a few different patterns:

- When `book:Expiry` or `book:CVV` tags are found, no additional steps are taken and these tags and any children of these tags are removed from the output.
- When a `book:ConfirmationCode` tag is encountered, it changes it into a `<book:ConfirmationText>` tag and then decodes the original tag's value. `dp:decode()` is an extension function that does the base64 decoding.
- When a `book:Number` tag is encountered, it uses the `substring()` function to get the last four numbers and replace the other numbers with asterisks.
- For anything else that does not match, an identity transform is used to copy it to the output document.

- \_\_\_ d. Click **Done**.
- \_\_\_ 8. The response rule is complete. Now you must correctly order the configured rules. Press the **Up arrow** for the new signed response rule as many times as required to move the rule above the other response rule. Recall that the other response rule matches on "all URLs", so it would get control before your new rule would be evaluated.

- \_\_\_ 9. Confirm the first response rule (**Server to Client**) listed in the configured rules is the rule that contains the **Sign** action.
- \_\_\_ 10. Click **Apply Policy** and close the policy editor window.
- \_\_\_ 11. Click **Apply** in the Configure Multi-Protocol Gateway window.

## 1.7. Test that the BookingServiceProxy sends a signed message

In this section, you verify in the SoapUI response tab that the DataPower service signs the message.

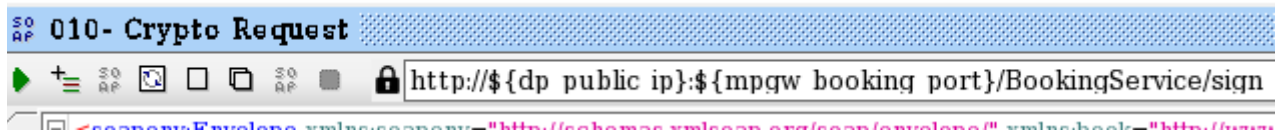
- \_\_\_ 1. Return to **SoapUI**.
- \_\_\_ 2. If necessary, click the **Auth** tab that is in the bottom portion of the 10 – Crypto Request.
- \_\_\_ 3. Set the **Incoming WSS** to **Verify**.



Outgoing WSS: Sign

Incoming WSS: Verify

- \_\_\_ 4. Ensure that the endpoint URL appears as:  
`http://${dp_public_ip}:${mpgw_booking_port}/BookingService/sign`



- \_\_\_ 5. Click the green **Submit** arrow to POST the message to BookingServiceProxy. The request should return a successful booking reservation. You must scroll down toward the bottom of the message to see the SOAP body with the `<book:BookingResponse>` element.

- \_\_\_ 6. Confirm that the top of the response message displays the signature information.

```

<soapenv:Envelope xmlns:book="http://www.ibm.com/datapower/FLY/Booking"
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2002/01/wss-schema"
      <wsu:Timestamp wsu:Id="Timestamp-8ffe1e6a-8d35-4318-98c1-b398ff"
        <wsu:Created>2015-02-10T21:15:02Z</wsu:Created>
        <wsu:Expires>2015-02-10T21:20:02Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken wsu:Id="SecurityToken-994b7fe2-f3b5-48"
        <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
          <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
              <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
                <Reference URI="#Timestamp-8ffe1e6a-8d35-4318-98c1-b398ff"
                  <Transforms>
                    <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
                      </Transform>
                  </Transforms>
                  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha-1"
                    <DigestValue>3jnABaxyhgL5k/OUL+bV49K+hQ=</DigestValue>
                  </DigestMethod>
                </Reference>
                <Reference URI="#id-741AB4A339C6C8C73A1423603007559151"
                  <Transforms>
                    <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
                      </Transform>
                  </Transforms>
                  <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha-1"
                    <DigestValue>JbL/s0Ji8FVhi3BRv5yw22/SI4=</DigestValue>
                  </DigestMethod>
                </Reference>
              </SignedInfo>
              <SignatureValue>wvCkyR5jh2owIWwP+sBdY5PRQ93ropRCP0baaAk
            </SignatureValue>
            <KeyInfo>
              <wsse:SecurityTokenReference xmlns=""
                <wsse:Reference URI="#SecurityToken-994b7fe2-f3b5-48bf-b0"
              </wsse:SecurityTokenReference>
            </KeyInfo>

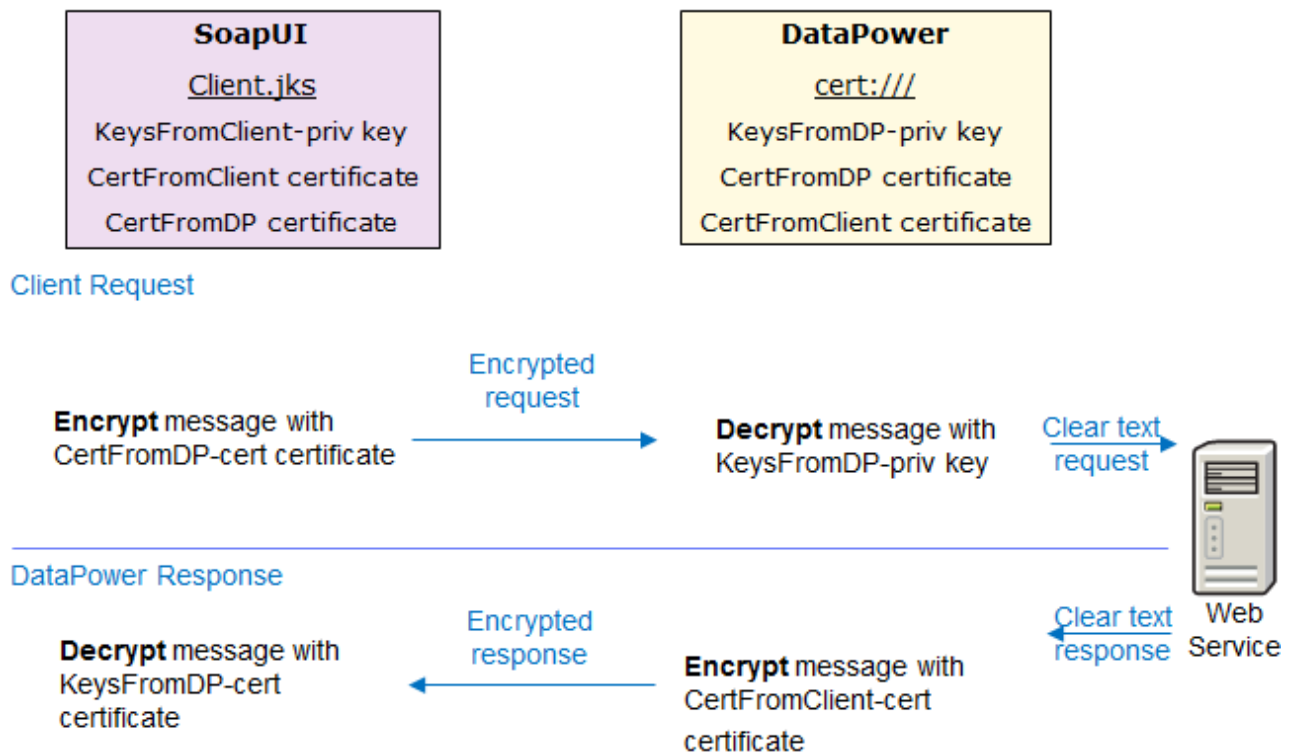
```

- \_\_\_ 7. Click **Save Configuration** in the WebGUI.




## 1.8. Configure BookingServiceProxy to decrypt a message

The client (SoapUI) and DataPower are configured to encrypt and decrypt messages. SoapUI encrypts the request message and DataPower decrypts it before sending it to the back end. Then, DataPower encrypts the response message from the back end before sending it back to SoapUI. SoapUI decrypts the response message from DataPower. This process is built within the next few sections.

### Encrypt and decrypt steps

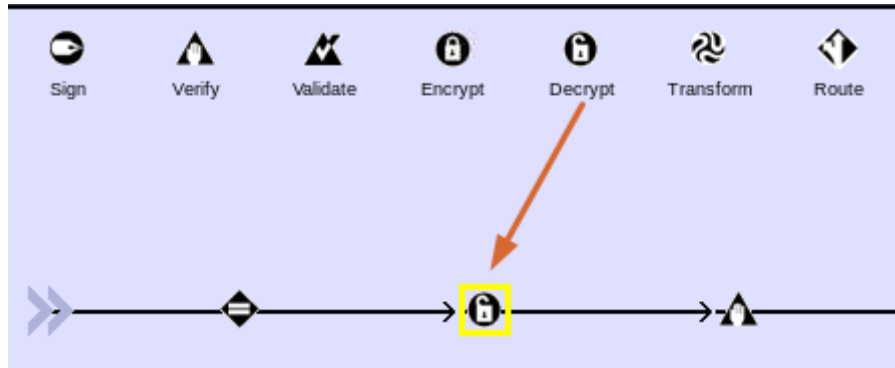


- \_\_\_ 1. Open the **BookingServicePolicy** policy editor.
- \_\_\_ 2. Modify the request rule that verifies the signature to decrypt the message first.
  - \_\_\_ a. Click the request rule that contains the Verify action to make the rule active in the policy editor.

Configured Rules		
Rule Name	Direction	
BookingServicePolicy_rule_4	Client to Server	  



- \_\_\_ b. Drag a **Decrypt** action before the **Verify** action.



- \_\_\_ 3. Configure the **Decrypt** action.

- \_\_\_ a. Double-click the **Decrypt** action for configuration.
- \_\_\_ b. Ensure that **Entire Message/Document** is selected as the **Message Type**.
- \_\_\_ c. Select **KeysFromDP** as the **Decrypt Key**.

**Decrypt**

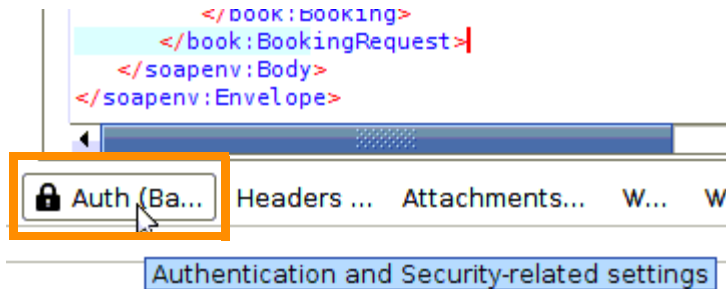
<b>Standard</b>	<input checked="" type="radio"/> XML Security <input type="radio"/> JSON Web Security *
<b>Message Type</b>	<input checked="" type="radio"/> Entire Message/Document <input type="radio"/> Selected Elements (Field-Level) <input type="radio"/> Advanced
<b>Asynchronous</b>	<input type="radio"/> on <input checked="" type="radio"/> off
<b>Decrypt Key</b>	<div>KeysFromDP ▾</div> <div>+</div> <div>...</div>

- \_\_\_ d. Click **Done**.
- \_\_\_ 4. Click **Apply Policy**.
- \_\_\_ 5. Close the policy window.
- \_\_\_ 6. Click **Apply** to apply the multi-protocol gateway.

## 1.9. Test the BookingServiceProxy message decryption

Send a signed and encrypted message from SoapUI to the BookingServiceProxy. The service should decrypt the message, and then validate the signature before sending it to the back end.

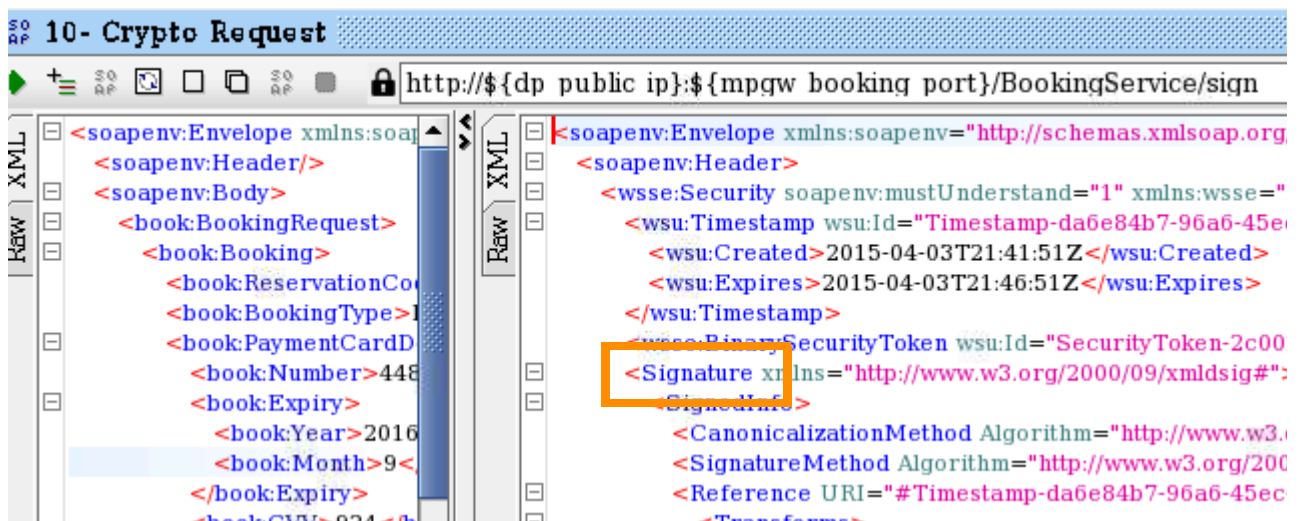
- \_\_\_ 1. In **SoapUI**, ensure that the **10 – Crypto Request** is open.
- \_\_\_ 2. If necessary, click the **Auth** tab that is in the bottom portion of the 10 – Crypto Request.



- \_\_\_ 3. Set the **Outgoing WSS** to **SignAndEncrypt**.















- \_\_\_ 4. Ensure that the endpoint URL appears as:  
http://\${dp\_public\_ip}:\${mpgw\_booking\_port}/BookingService/sign
- \_\_\_ 5. Click the green **Submit** arrow to POST the message to BookingServiceProxy.
- \_\_\_ 6. Confirm that the BookingResponse is received. Because SoapUI does not remove the WS-Security headers, the <Signature> element is in the SOAP header. You must scroll down to see the BookingResponse in the SOAP body.



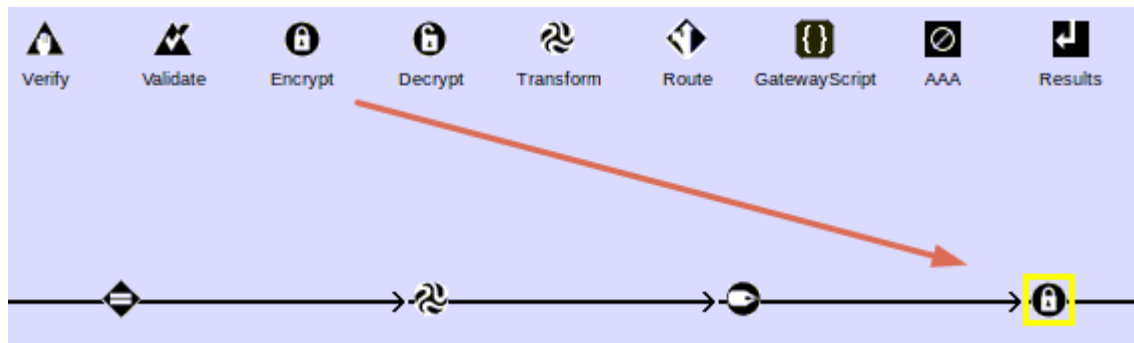
## 1.10. Configure BookingServiceProxy to encrypt the response message

In the response rule that signs the message, add an Encrypt action. SoapUI needs to decrypt and verify the signature on the response message.

- \_\_\_ 1. Open the **BookingServicePolicy** policy editor.
- \_\_\_ 2. Configure the response rule for encryption.
  - \_\_\_ a. Click the **response** rule that contains the Sign action so that it becomes the active rule in the policy editor.

Configured Rules		
Rule Name	Direction	Actions
BookingServicePolicy_rule_2	Client to Server	  
BookingServicePolicy_rule_3	Server to Client	  
BookingServicePolicy_rule_0	Client to Server	   
BookingServicePolicy_rule_1	Server to Client	  

- \_\_\_ b. Drag an **Encrypt** action after the **Sign** action.



- \_\_\_ 3. Configure the **Encrypt** action.
  - \_\_\_ a. Double-click the **Encrypt** action for configuration.
  - \_\_\_ b. Select **WSSEC Encryption** as the **Envelope Method**.
  - \_\_\_ c. Select **SOAP Message** as the **Message Type**.

- \_\_\_ d. Select **CertFromClient** as the **Recipient Certificate**.

**Encrypt**

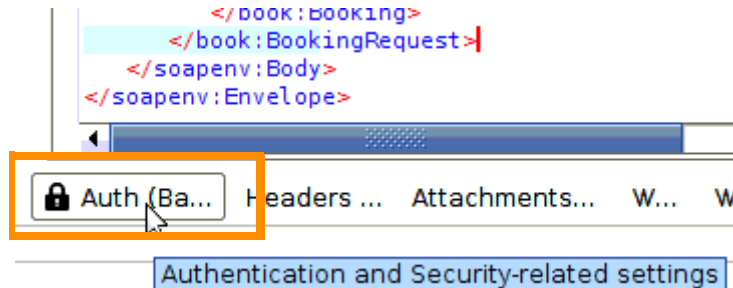
<b>Standard</b>	<input checked="" type="radio"/> XML Security <input type="radio"/> JSON Web Security *
<b>Envelope Method</b>	<input checked="" type="radio"/> WSSec Encryption <input type="radio"/> Standard XML Encryption <input type="radio"/> Advanced *
<b>Message Type</b>	<input checked="" type="radio"/> SOAP Message <input type="radio"/> Raw XML Document <input type="radio"/> Selected Elements (Field-Level) <input type="radio"/> Advanced *
<b>Asynchronous</b>	<input type="radio"/> on <input checked="" type="radio"/> off
<b>Message and Attachment Handling</b>	Message Only <input type="button" value="Save"/>
<b>Encryption Key Type</b>	Use Ephemeral Key Transported by Asy
<b>Use Dynamically Configured Recipient Certificate</b>	<input type="radio"/> on <input checked="" type="radio"/> off <input type="button" value="Save"/>
<b>One Ephemeral Key</b>	<input type="radio"/> on <input checked="" type="radio"/> off <input type="button" value="Save"/>
<b>Recipient Certificate</b>	CertFromClient <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/>

- \_\_\_ e. Click **Done**.
- \_\_\_ 4. Click **Apply Policy**.
- \_\_\_ 5. Close the policy editor.
- \_\_\_ 6. Click **Apply** to apply the multi-protocol gateway.
- \_\_\_ 7. Enable the probe for the MPG. You will examine the rule execution in the next section.

## 1.11. Test the BookingServiceProxy message encryption

Configure SoapUI to decrypt and verify the signature of the message it receives from DataPower.

- \_\_\_ 1. Return to **SoapUI**.
- \_\_\_ 2. If necessary, click the **Auth** tab that is in the bottom portion of the 10 – Crypto Request.



- \_\_\_ 3. Set the **Incoming WSS** to **DecryptAndVerify**.



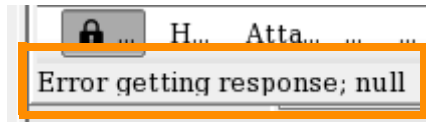
- \_\_\_ 4. Ensure that the endpoint URL appears as:  
`http://${dp_public_ip}:${mpgw_booking_port}/BookingService/sign`
- \_\_\_ 5. Click the green **Submit** arrow to POST the message to BookingServiceProxy. The request should return a successful booking reservation.
- \_\_\_ 6. Confirm that the top of the response message displays the encryption information. If you see nothing in the right pane, check at the bottom of the request window to see whether the message “Error getting response” is present. If so, read the warning that follows this step.

```
<soapenv:Envelope xmlns:book="http://www.ibm.com/datapower/FLY/Booki
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1" xmlns:wsse="http://docs.o
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/x
        <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/ws
          </wsse:SecurityTokenReference>
        </dsig:KeyInfo>
        <xenc:CipherData xmlns:dsig="http://www.w3.org/2000/09/xmldsig:
          <xenc:CipherValue>EFiSuQ1c8ofboio45o1DOmQVs2VcF9Pw9!
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#body"/>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
```



## Attention

The SoapUI tool frequently fails on the “DecryptAndVerify”. If you get this error message at the bottom of the request window, then this failure probably occurred:



An issue exists with this specific Java code that underlies SoapUI; it has nothing to do with the DataPower operation.

The best way to see what your response message contains is to click **http log** on the bottom of the SoapUI window:



This pane displays the HTTP data flow. At the bottom of the scrollable pane, you are looking at the end of the HTTP response, which is signed and encrypted. If you scroll right (it is not a formatted pane), you can see the <EncryptedData> element within the SOAP body. This element contains the signed and encrypted BookingResponse. If you look further up in the pane, you can see the <Signature> element. If you want, you can compare these HTTP contents to the OUTPUT contents of the probe.

You do not have the processed XML response visible in the response tab to do the next two steps.

- \_\_\_ 7. Scroll below the encryption section of the response message, and confirm that the response message contains the signature information.

```

</wsu:Timestamp>
<wsse:BinarySecurityToken wsu:Id="SecurityToken-69bd0213-c53
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/2001,
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xml
    <Reference URI="#Timestamp-d3f26cd4-fc22-4a3f-9e3e-52f4
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-e3
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
      <DigestValue>+ucI1vWk28oakTDJABc9mTNHCA=</Digest
    </Reference>
    <Reference URI="#id-741A14A339C6C8C73A1423603356541
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-e3
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmld
      <DigestValue>Y96TmLlrLAzq4Ru7vV1kVWqJrGE=</DigestV
    </Reference>
  </SignedInfo>
  <SignatureValue>e5yTSoeRqxsQ6W8jj9QIjJGELSPpi4cuE7Ive3

```

- \_\_\_ 8. Scroll further down the response message and confirm that the BookingResponse is received.

```

</soapenv:header>
<soapenv:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:book="http://www.ibm.com/xml/schemas/booking" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <book:BookingResponse>
    <book:ConfirmationText>Processed 0111459898A</book:ConfirmationText>
    <book:Booking>
      <book:ReservationCode>JK99V16I</book:ReservationCode>
      <book:BookingType>I</book:BookingType>
      <book:PaymentCardDetails>
        <book:Number>*****5191</book:Number>
        <book:Type>Visa</book:Type>
        <book:HolderName>James Roberts</book:HolderName>
      </book:PaymentCardDetails>
      <book:BillingDetails>
        <book:FirstName>James</book:FirstName>
        <book:LastName>Roberts</book:LastName>
        <book:Address>314 S. Wells St</book:Address>
        <book:City>Chicago</book:City>
        <book:State>IL</book:State>
        <book:ZIP>60606</book:ZIP>
        <book:Country>USA</book:Country>
      </book:BillingDetails>
    </book:Booking>
  </book:BookingResponse>
</soapenv:Body>
</soapenv:Envelope>

```

- \_\_\_ 9. You confirmed that DataPower sent an encrypted and signed response message.

## 1.12. Examine the rule processing by using the multi-step probe

In this last section, you use the probe to examine the behavior inside the request and response rules.

- \_\_\_ 1. If the Transaction List probe window is not visible, click **Show Probe**.



### Configure Multi-Protocol Gateway

[Export](#) | [View Log](#) | [View Status](#) | [Show Probe](#)

Multi-Protocol Gateway status: [up]

### General Configuration

- \_\_\_ 2. Click **Refresh**.
- \_\_\_ 3. Expand the last entry in the list. It might be the only entry.

DataPower XI52					
<input type="button" value="Refresh"/> <input type="button" value="Flush"/> <input type="button" value="Disable Probe"/> <input type="button" value="Export Capture"/> <input type="button" value="View Log"/> <input type="button" value="Send Message"/> <input type="button" value="Close"/>					
view	trans#	type	inbound-url	outbound-url	rule
	405298	request	http://172.16.78.24:12311/BookingService/sign	http://dp_internal_ip:9080/BookingService/sign	BookingS

- \_\_\_ 4. Observe that both the request rule and the response rule are displayed in the probe list. You know that no errors exist because the rules are listed in black and not red. Click the magnifying glass that is located next to the request rule.

DataPower XI52					
<input type="button" value="Refresh"/> <input type="button" value="Flush"/> <input type="button" value="Disable Probe"/> <input type="button" value="Export Capture"/> <input type="button" value="View Log"/> <input type="button" value="Send Message"/> <input type="button" value="Close"/>					
view	trans#	type	inbound-url	outbound-url	rule
	405298	request	http://172.16.78.24:12311/BookingService/sign	http://dp_internal_ip:9080/BookingService/sign	BookingS
	405298	response	http://172.16.78.24:12311/BookingService/sign	http://dp_internal_ip:9080/BookingService/sign	BookingS



- \_\_\_ 5. Click the magnifying glass that is on the request rule.
  - \_\_\_ a. You can examine the state of the message that is going through the rule. The actions are displayed sequentially. The request message from SoapUI is decrypted, validated, stripped, and then sent to the back-end web service. Click the various magnifying glasses to see the message state at each step in the rule.

Input Context '1' of Step 0



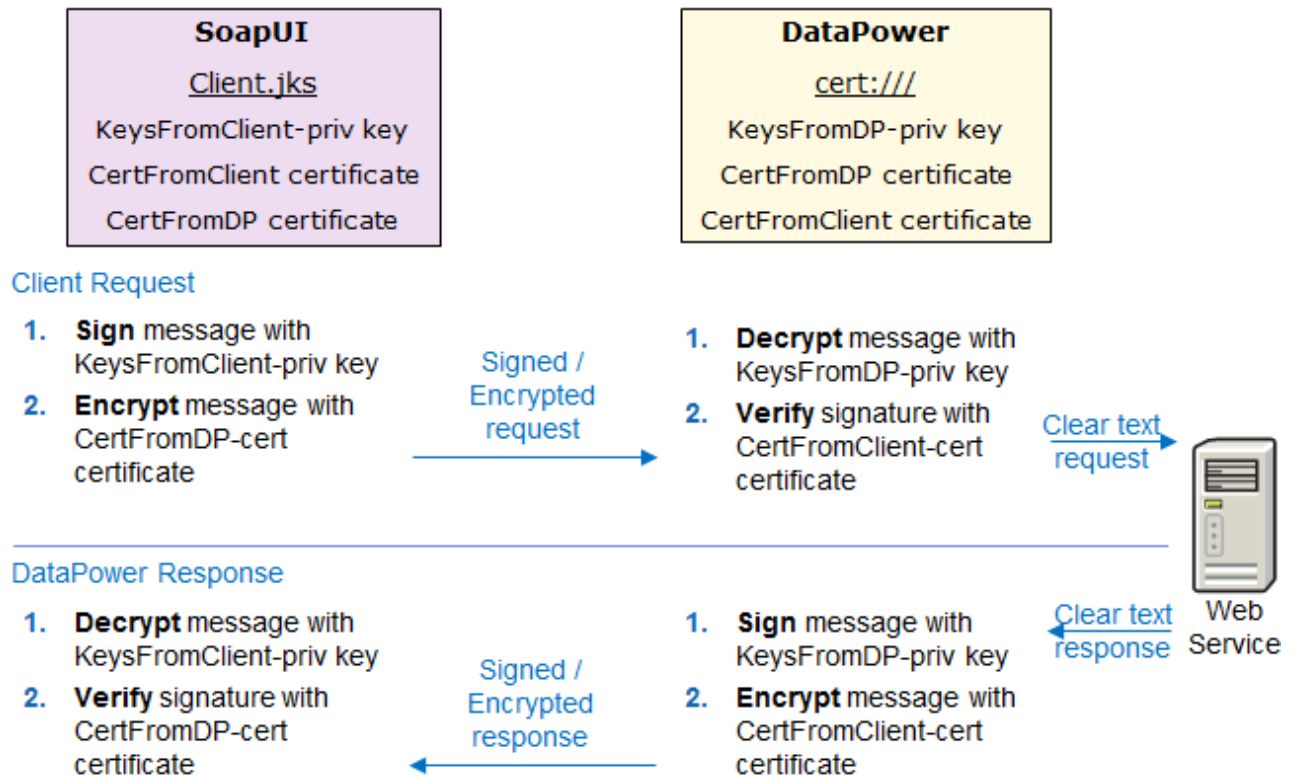
- \_\_\_ b. Close the transaction window after you complete reviewing the message details.
- \_\_\_ 6. Click the magnifying glass located next to the response rule.
  - \_\_\_ a. You can examine the state of the message that is going through the response rule. The actions are displayed sequentially. The response message from the back end is signed, encrypted, and then returned to SoapUI.



- \_\_\_ b. Close the transaction window after you complete reviewing the message details.
- \_\_\_ 7. Click **Disable Probe** to turn off the multi step probe.
- \_\_\_ 8. Click **Close**.
- \_\_\_ 9. Click **Save Configuration** to save your work.

The complete process is shown in the following figure:

## Completed exercise



## End of exercise

## Exercise review and wrap-up

In this exercise, you learned how to sign, validate, encrypt, and decrypt a message that conforms to the WS-Security specification. You defined the needed DataPower key and certificate objects from the provided key materials. You created a request rule that decrypts and validates a signed and encrypted message from SoapUI. You created a response rule that signed and encrypted a message that was returned to SoapUI.

---

# Exercise 2. Configuring a web service proxy

## Estimated time

01:00

## Overview

In this exercise, you create a web service proxy (WS-Proxy) service that virtualizes or proxies a back-end web service. A web service proxy can mask the actual endpoint of the web service. You configure the web service proxy by uploading a WSDL document for the service. After you create the web service proxy, you configure the service policy with rules and actions for each service that is defined within your proxy.

## Objectives

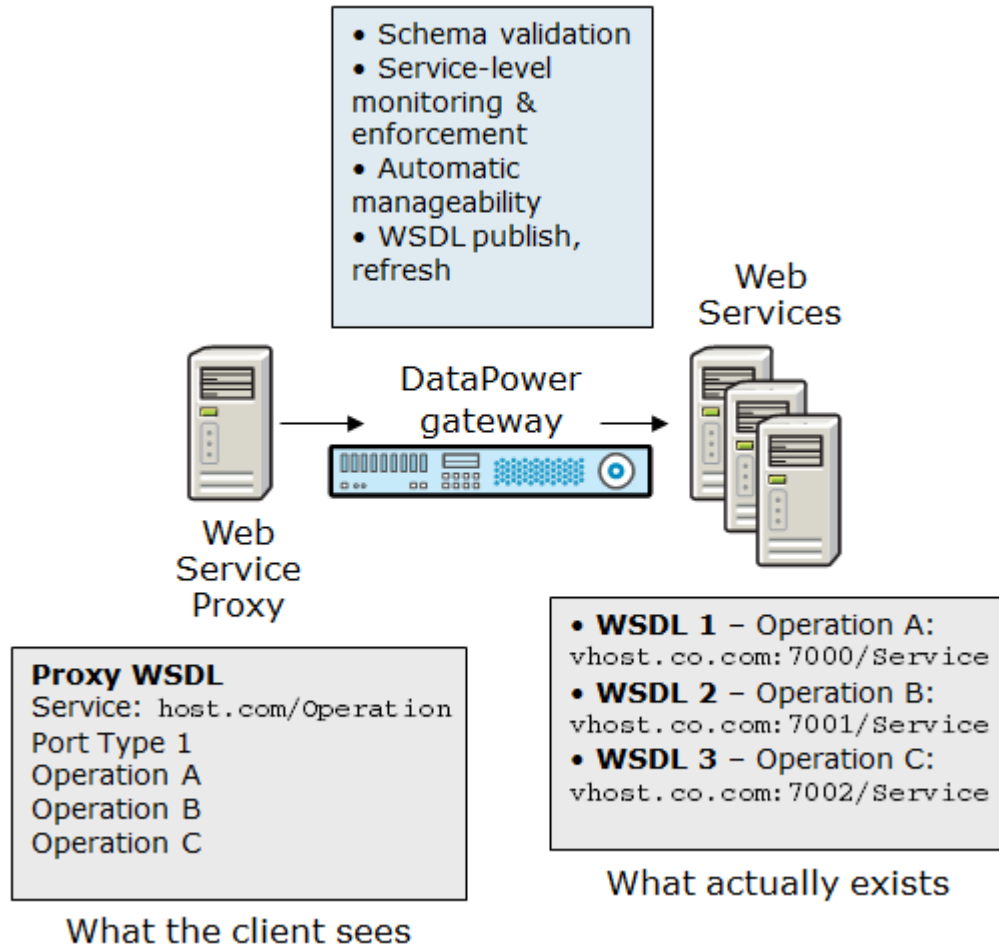
After completing this exercise, you should be able to:

- Configure a web service proxy to virtualize an existing web service
- Configure the service policy within the web service proxy

## Introduction

A web service proxy is used to externalize your web services to protect them from malicious attacks. A client cannot directly connect to your service; all requests enter through the web service proxy. You can decouple security, validation, and management from your back-end web service and do these tasks on the web service proxy.

The WS-Proxy also can proxy multiple back-end web services within a single WS-Proxy service by presenting a single virtual WSDL to the client. The client can access only the operations that are visible in the virtual WSDL regardless of their existence or location on the actual back-end systems.



The DataPower gateway supports the creation of a web service proxy by uploading a WSDL file that describes the web services.

In this exercise, you upload a WSDL file for the Booking Service web services. The WSDL files contain the endpoint address of the web service. Using the web service proxy, you create a virtual address for the service, which the client calls to invoke the web service. In addition, the web service proxy validates both request and response messages and can publish the virtual WSDL document. You can also configure a policy with rules and actions at a fine-grained level. The policy can be applied at the WSDL service, port, or operation level. In this exercise, you configure a policy on the `BookTravel` operation. The policy has a **Transform** action to convert the confirmation code into a more readable confirmation text.

You also learn to configure operational level processing by using a series of check boxes that control the web service proxy behavior.

Finally, decryption is added to the web service proxy, not by using a **Decrypt** action on a rule in a processing policy, but by using the Proxy Setting capabilities in the WS-Proxy.

You use SoapUI to send the appropriate SOAP message, with and without encryption.

## Requirements

To complete this exercise, you need:

- Access to the DataPower gateway
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- SoapUI to send requests to the DataPower gateway
- The BookingServiceBackend web service that runs on the DataPower gateway in the `FLYServices` domain
- Access to the `<lab_files>` directory

# Exercise instructions

## Preface

- Before starting this lab, complete the steps in Exercise 1: Web service encryption and digital signatures.
- Remember to use the domain and port address that you were assigned in the exercise setup. **Do not** use the default domain.
- The references in exercise instructions refer to the following values:
  - `<lab_files>`: Location of the student lab files. Default location is: `/usr/labfiles/dp/`
  - `<image_ip>`: IP address of the student image (use `/sbin/ifconfig` from a terminal window to obtain value).
  - `<dp_internal_ip>`: IP address of the DataPower gateway development and administrative functions that are used by internal resources such as developers.
  - `<dp_public_ip>`: IP address of the public services on the gateway that is used by customer and clients.
  - `<dp_WebGUI_port>`: Port of the WebGUI. The default port is 9090.
  - `<nn>`: Assigned student number. If no instructor exists, use "01".
  - `<studentnn>`: Assigned user name and user account. If no instructor exists, use "student01".
  - `<studentnn_password>`: Account password. In most cases, the initial value is the same as the user name. You are prompted to create a password on first use. Write it down.
  - `<studentnn_domain>`: Application domain that the user account is assigned to. If no instructor exists, use "student01\_domain".
  - `<FLY_booking_port>`: Port number that the back-end BookingServices web services listen on. The default port is 9080.
  - `<wsp_proxy_port>`: 12nn5, where "nn" is the two-digit student number. This port number is the listener port for the BookingServiceWSPProxy service.

## 2.1. Create a web service proxy for the Booking Service web service

Configure a new web service proxy on the IBM DataPower Gateway to forward requests to the Booking Service web service.

- \_\_\_ 1. Log on to the DataPower WebGUI.
- \_\_\_ 2. Create a web service proxy from the Control Panel.



### Information

A web service proxy can be created two ways, by using either the Control Panel icon or the vertical navigation menu. Using the Control Panel icon provides a more intuitive user interface.

- \_\_\_ a. Click the **Web Service Proxy** icon.



Web Service Proxy

- \_\_\_ b. On the Configure Web Service Proxy catalog page, click **Add**.
- \_\_\_ c. For the Web Service Proxy Name, enter `BookingServiceWSProxy` and then click **Create Web Service Proxy**.
- \_\_\_ 3. The Configure Web Service Proxy page displays. Add the `BookingService` web service endpoint to the newly created web service proxy.
  - \_\_\_ a. On the **WSDL files** tab, ensure that the **Add WSDL** option is selected.

### WSDLs

Edit WSDL or Subscription	Add WSDL	Add UDDI Subscription	Add WSRR Subscription	Add WSRR Saved Search Subscriptio
---------------------------	----------	-----------------------	-----------------------	-----------------------------------



- \_\_\_ b. You might already have **BookingService.wsdl** in the `local:` directory from a previous exercise. Select it for the **WSDL File URL**. If it is not in the list, upload it from `<lab_files>/BookingService/BookingService.wsdl`.

### WSDLs

Edit WSDL or Subscription   Add WSDL   Add UDDI Subscription   Add WSRR Subscription   Add V

**WSDL File URL**

local:///              

**BookingService.wsdl**              

**Use WS-Policy References**

☒ on ☐ off

**WS-Policy Parameter Set**

(none)     

**WS-Policy Enforcement Mode**

Enforce  

**SLA Enforcement Mode**

Allow  

- \_\_\_ c. Click **Next**.
- \_\_\_ 4. Create a front side handler to accept HTTP requests for the web service proxy.
- \_\_\_ a. The entry for **BookingService – BookingServiceSOAP** is displayed.
- \_\_\_ b. Click new (+) beside the **Local Endpoint Handler** list.

### Web Service Proxy WSDLs

BookingService - BookingServiceSOAP

Local		
Local Endpoint Handler	URI	Bind
(none) <input type="button" value="+"/> <input type="button" value="..."/>	/BookingService/	<input checked="" type="checkbox"/> S <input type="checkbox"/> F

- \_\_\_ c. Select **HTTP Front Side Handler** as the local endpoint handler type.

- \_\_\_ d. Configure the new HTTP front-end handler with the following values. Leave all other settings to the default values.
- **Name:** BookingServiceWSProxyFSH
  - **Local IP Address:** Host alias of dp\_public\_ip
  - **Port Number:** <wsp\_proxy\_port>

#### HTTP Front Side Handler

Apply Cancel

**Name** BookingServiceWSProxyFSH \*

---

**Administrative state** ☒ enabled ☐ disabled

**Comments**

**Local IP address** dp\_public\_ip

**Port** 12345 \*

**HTTP version to client** HTTP 1.1

- \_\_\_ e. Click **Apply** to save the changes that are made to the HTTP front side handler.

- \_\_\_ 5. Verify that the local URI is /BookingService/.

#### Web Service Proxy WSDLs

BookingService - BookingServiceSOAP

Local			
Local Endpoint Handler	URI	Binding (Suffix)	Edit/Remove
BookingServiceWSProxyFSH + ...	/BookingService/	<input checked="" type="checkbox"/> SOAP 1.1 <input type="checkbox"/> SOAP 1.2 <input type="checkbox"/> HTTP GET	Add +



#### Note

The row that starts with “Local Endpoint Handler” cell is the section that lists all handlers that are defined for this WS-Proxy. Any handlers in this section can be edited or deleted. The bottom row is used to create a handler. You must click **Add** to create a handler from the specifications in this row.

- \_\_\_ 6. Click **Add** (green plus sign) in the Edit/Remove column to add this definition set to the client aspects of this WSDL.

- \_\_ 7. Set the back-end destination information.
  - \_\_ a. In the Remote (outbound) section, set the Remote Endpoint Host entry for the application server to: <dp\_internal\_ip>

Remote


Protocol	Remote Endpoint Host	Port	Remote URI
HTTP	172.16.17.23	9080	/BookingService/

Published

☒ Use Local

- \_\_ b. Set the Port value to: <FLY\_booking\_port>
- \_\_ c. The Remote URI should be **/BookingService/**.
- \_\_ d. Click **Next** to continue the configuration.
- \_\_ e. Verify that the **WSDL entry** in the BookingServiceWSPProxy has one active endpoint and that it is configured (1 up / 1 configured).

WSDLs

Edit WSDL or Subscription		Add WSDL	Add UDDI Subscription	Add WSRR Subscription	Add WSRR Saved Search Subscript
WSDL Source Location		Endpoint Handler Summary		WSDL Status	WS-
	local:///BookingService.wsdl	1 up / 1 configured		Okay	Okay

## 2.2. Verify the generated components

In the last section, you used a WSDL document for the BookingService web services. The DataPower gateway creates several components as a result of this action. In this section, you examine the components that the gateway created.

- \_\_\_ 1. Examine the components in the **WSDL files** tab of the web service proxy web page.
  - \_\_\_ a. Expand the `local:///BookingService.wsdl` to examine the local and remote proxy settings. You should recognize this information from your previous activity.
  - \_\_\_ b. Under **Local** for the WSDL file, you can see the incoming URI and endpoint handlers. The **URI** field specifies the URI that the client uses to request this service. The service uses a local endpoint handler called `BookingServiceWSProxyFSH`.
  - \_\_\_ c. Under **Remote**, you see the endpoint (URI) of the back-end web service. You are virtualizing this endpoint so that clients are not required to call the web service directly.



### Note

This endpoint handler contains the proxy port number on the gateway that is listening for requests and various HTTP options for the HTTP connection.

If you decide to change the web service endpoint address, it is not necessary for you to tell the client because the local URI remains unchanged. All that you must do is update the **Remote Endpoint Host** or the **Port** field in the web service proxy configuration.

- \_\_\_ 2. Click the **Services** tab of the web service proxy. Verify that you see the service.

### Services

WSDL Name: BookingService.wsdl.dpdup.0		
Service	BookingService	Publish to UDDI



### Note

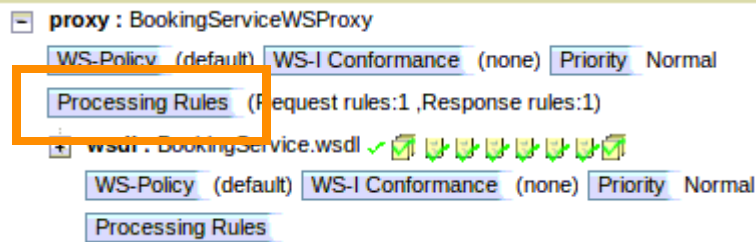
Each WSDL file contains a Services section that describes the web service endpoints that are available. For example, the `BookingServiceWSProxy.wsdl` contains the following element:

```
<wsdl:service name="BookingService">
  <wsdl:port binding="tns:BookingServiceSOAP" name="BookingServiceSOAP">
    <soap:address location="http://booking.FLY.com:9070/BookingService/" />
  </wsdl:port>
</wsdl:service>
```

Notice that the `wsdl:service` name inside the `BookingServiceWSProxy.wsdl` file is the same as the service name in the **Services** tab on the web service proxy web page.

It is possible for you to click **Publish to UDDI** to publish this service into a UDDI registry. To do so, you need the UDDI registry publish and inquiry URI, user ID, and password. The DataPower gateway itself does not contain a UDDI registry.

- \_\_\_ 3. Examine the **Policy** tab to view the BookingServiceWSProxy service policy.
  - \_\_\_ a. Click the **Policy** tab.
  - \_\_\_ b. Look in the WSDL Policy Tree Representation section of the Policy tab.
  - \_\_\_ c. Under `proxy: BookingServiceWSProxy`, click **Processing Rules**. A policy editor section opens beneath the WSDL policy tree.



The policy editor section is similar to the separate policy editor window that you see for the multi-protocol gateways or XML firewalls. The difference is that the policy editor section shows the request, response, and error rules for only the WSDL level that is selected in the policy tree.



### Information

When you create a web service proxy, the gateway generates default proxy-level request and response rules. The proxy-level request rule contains two actions: service level monitoring (SLM) and Results. The response rule is just a Results action.

- \_\_\_ d. The request rule should be initially selected. Hover your mouse pointer over the SLM action. Notice that the SLM policy has the value `BookingServiceWSProxy`. This SLM policy object was created for you. It can be modified in the **SLM** tab. The **SLM** tab is used to configure service level monitoring.
- \_\_\_ e. Hover your mouse pointer over the **Results** action. This action moves the INPUT context to the OUTPUT context.



- \_\_\_ f. Under Configured Rules in the lower part of the section, click the response rule to view it.



## Information

These two rules are proxy-level rules that are applied to every service, port, and operation in the proxy. You can override these rules by defining policies at a fine-grained level. For example, you can have a policy for each operation. This operation-level policy overrides the proxy-level policy.

- g. Under WSDL Policy Tree Representation, expand the hierarchical view until the port-operation is exposed. Notice that each service, port, and operation has an identical set of icons. These sets represent the user policy. Each of these icons represents more validation that is done and the publishing of the WSDL document to the web service proxy.

☐ **proxy : BookingServiceWSProxy**

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules (Request rules:1 ,Response rules:1)

☐ **wsdl : BookingService.wsdl.dpdup.0** ✓ 

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

☐ **service : {http://www.ibm.com/datapower/FLY/BookingService/}BookingService**

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

☐ **port : {http://www.ibm.com/datapower/FLY/BookingService/}BookingService**

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

☐ **port-operation : BookTravel** ✓ 

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules



© Copyright IBM Corp. 2017

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

2-11


- \_\_\_ h. Click any of the icons that are located under the **port-operation**: BookTravel to see the options.

#### Processing Rules

**service** : {http://www.ibm.com/datapower/FLY/BookingService/}BookingService ✓   











WS-Policy (default) WS-I Conformance (none) Priority Normal









#### Processing Rules

**port** : {http://www.ibm.com/datapower/FLY/BookingService/}BookingServiceSOAP ✓ 









WS-Policy (default) WS-I Conformance (none) Priority Normal

#### Processing Rules

**port-operation** : BookTravel ✓          

WS-Policy (default)	WS-I Conformance	Effective Value	Local Value
Processing Rules		✓	<input checked="" type="checkbox"/> Enable this component
			<input type="checkbox"/> Publish in WSDL
			<input checked="" type="checkbox"/> Schema validate faults messages
			<input checked="" type="checkbox"/> Schema validate request messages
			<input checked="" type="checkbox"/> Schema validate response messages
			<input checked="" type="checkbox"/> Do not schema validate SOAP headers
			<input checked="" type="checkbox"/> Use WS-Addressing
			<input checked="" type="checkbox"/> Use WS-ReliableMessaging
			<input checked="" type="checkbox"/> Accept MTOM / XOP Optimized Messages

- \_\_\_ i. Clear **Schema validate response messages**.

Effective Value	Local Value
✓	<input checked="" type="checkbox"/> Enable this component
	<input type="checkbox"/> Publish in WSDL
	<input checked="" type="checkbox"/> Schema validate faults messages
	<input checked="" type="checkbox"/> Schema validate request messages
	<input type="checkbox"/> Schema validate response messages
	<input checked="" type="checkbox"/> Do not schema validate SOAP headers
	<input checked="" type="checkbox"/> Use WS-Addressing
	<input checked="" type="checkbox"/> Use WS-ReliableMessaging
	<input checked="" type="checkbox"/> Accept MTOM / XOP Optimized Messages

Done

CLOSE

**Information**

The `BookingService.wsdl` has two elements in the response message in a different sequence from what is used in the actual response message. Therefore, the WSDL validation of the returning response message fails to pass the web service proxy's validation. You disable this validation so the web service proxy accepts the incoming response message. In this case, the differing order of elements has no impact. This situation demonstrates a possible reason to modify the user policy. Most likely, in production you would not allow a message to be processed that does not match the schema.

---

- \_\_\_ j. Click **Done** to save the change to the user policy. This interaction should close the user policy window.
- \_\_\_ k. Click **Apply** at the top of the page to save the change to the WS-Proxy.
- \_\_\_ l. Under the web service proxy policy, notice that you can create a rule at each level of the WSDL file, service, port, and operation by clicking the **Processing Rules** link. In a later section of this exercise, you create an operation-level rule.



## 2.3. Test the web service proxy processing

A SOAP message is sent to the `BookingServiceWSPProxy` web service proxy by using the HTTP protocol. The `BookingServiceWSPProxy` web service proxy was built by using a WSDL. The developer did not need to manually create the processing policy and all the appropriate rules. This configuration was done automatically by the web service proxy during the WSDL import. If your service is oriented around a WSDL and its operations, you should use a web service proxy rather than a multi-protocol gateway due to the simplicity of the service creation. Also, a web service proxy can be dynamically updated when linked to an external repository. In such a case, if one updated the WSDL or Concept (group of linked WSDLs) in the repository, those changes would automatically be reflected in the web service proxy service on the gateway.

- \_\_\_ 1. In SoapUI, open **15 – WSP Request**.

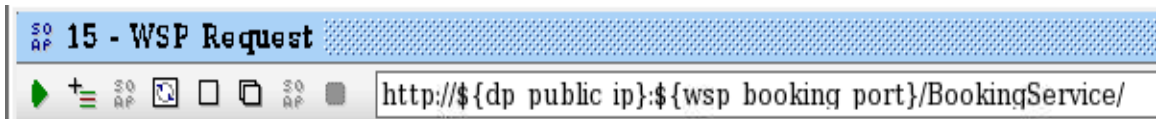


### Attention

In the request tab, do *not* “Format XML”.

- \_\_\_ 2. Ensure that the endpoint URL appears as:

`http://${dp_public_ip}:${wsp_booking_port}/BookingService/`



- \_\_\_ 3. Click the green **Submit** arrow to POST the message to `BookingServiceWSPProxy`.

- \_\_\_ 4. The response tab should show a booking response. Scroll down in the response to see the BookingResponse.

```

<soapenv:Body>
  <!--Happy Path UC1-->
  <book:BookingResponse>
    <book:ConfirmationText>Processed 0112459898A</book:ConfirmationText>
    <book:Booking>
      <book:ReservationCode>JK99V16I</book:ReservationCode>
      <book:BookingType>I</book:BookingType>
      <book:PaymentCardDetails>
        <book:Number>*****5191</book:Number>
        <book:Type>Visa</book:Type>
        <book:HolderName>James Roberts</book:HolderName>
      </book:PaymentCardDetails>
      <book:BillingDetails>
        <book:FirstName>James</book:FirstName>
        <book:LastName>Roberts</book:LastName>
        <book:Address>314 S. Wells St</book:Address>
        <book:City>Chicago</book:City>
        <book:State>IL</book:State>
        <book:ZIP>60606</book:ZIP>
        <book:Country>USA</book:Country>
      </book:BillingDetails>
    </book:Booking>
  </book:BookingResponse>
</soapenv:Body>
:/soapenv:Envelope>

```



### Optional

If you want, you can enable the user policy option for **Schema validate response messages** and resend the request. Use the system log and probe to see the details of the error.

Be sure to clear the **Schema validate response messages** check box and apply the WS-Proxy before you proceed in the exercise.

## 2.4. Add an operation-level rule to BookingServiceWSProxy

In this section, you configure a response rule for the BookTravel operation for the BookingServiceWSProxy web service (that is, an operation-level policy). The response rule reformats one of the returned elements.

- \_\_\_ 1. Generate an operation-level proxy policy for the BookingServiceWSProxy web service.
  - \_\_\_ a. Switch to the DataPower WebGUI. Make sure that you are still in the **Policy** tab of the BookingServiceWSProxy web service proxy configuration.



### Note

Recall that the gateway generates a default proxy-level request rule and response rule. You can override these rules at a fine-grained level. You override the default proxy-level request policy with the BookingServiceWSProxy operation-level policy.

- \_\_\_ b. In the WSDL policy tree view, expand and locate the BookTravel operation for BookingServiceWSProxy.

**proxy : BookingServiceWSProxy**

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules (Request rules:1 ,Response rules:1)

**wsdl : BookingService.wsdl.dpdup.0** ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

**service : {http://www.ibm.com/datapower/FLY/BookingService/}BookingService**

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

**port : {http://www.ibm.com/datapower/FLY/BookingService/}BookingService**

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

**port-operation : BookTravel** ✓ ✗ ✓ ✓ ✓ ✗ ✓ ✓ ✓ ✓

WS-Policy (default) WS-I Conformance (none) Priority Normal

Processing Rules

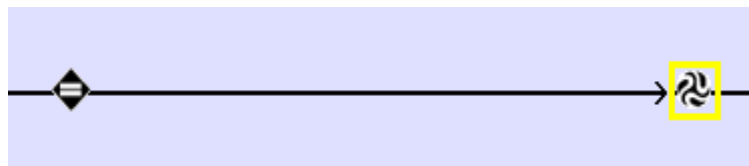
- \_\_\_ c. Click **Processing Rules** underneath **port-operation: BookTravel**. The policy editor section is displayed.

- \_\_\_ d. Under Policy Configuration, click **New Rule**.
- \_\_\_ e. Enter a rule name (for example, `BookingServiceWSProxy_rule_2`).
- \_\_\_ f. Set the Rule Direction to **Server to Client** from the list.
- \_\_\_ g. On the policy rule line, you see an automatically generated **Match** action.

**Note**

For WS-Proxies, the **Match** action is automatically generated.

- \_\_\_ 2. Add a **Transform** action to modify the response message.
  - \_\_\_ a. Drag the **Transform** action after the **Match** action.



- \_\_\_ b. Double-click the **Transform** action to open the configuration page.
- \_\_\_ c. Ensure that **Transform with XSLT style sheet** is selected in the Use Document Processing Instructions.
- \_\_\_ d. Select `BookingResponse_Transform.xsl` from the local files. If the file does not exist, then click **Upload** to upload the stylesheet from `<lab_files>/BookingService`.

**Note**

The `BookingResponse_Transform.xsl` stylesheet transforms the response message element `<book:ConfirmationCode>` to a `<book:ConfirmationText>` element with its related contents.

- \_\_\_ e. Click **Done** to close the Configure Transform page.
- \_\_\_ f. Click **Apply** at the top of the page to save the policy.

## 2.5. Test the web service proxy processing with a Transform rule

A SOAP message is sent to the `BookingServiceWSPProxy` web service proxy by using the HTTP protocol.

- \_\_\_ 1. Return to the **15 – WSP Request** request window in SoapUI.



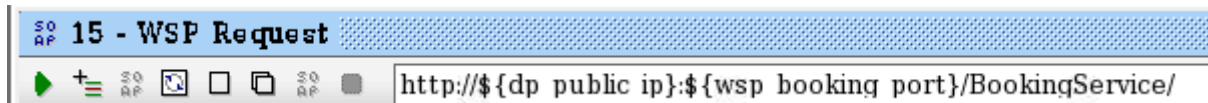
### Attention

In the request tab, do *not* “Format XML”.

- \_\_\_ 2. Before sending another request from SoapUI, review the response message from the previous section. Notice the `ConfirmationCode` element and its contents.

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:book="http://www.ibm.com/datapower/FLY/BookingService/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <!--Happy Path UC1-->
    <book:BookingResponse>
      <book:ConfirmationCode>UHJvY2Vzc2VkdAxtMTI0NTk4OThB</book:ConfirmationCode>
      <book:Booking>
        <book:ReservationCode>JK99V16I</book:ReservationCode>
        <book:BookingType>I</book:BookingType>
      </book:Booking>
    </book:BookingResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

- \_\_\_ 3. Ensure that the endpoint URL remains as:  
`http://${dp_public_ip}:${wsp_booking_port}/BookingService/`



- \_\_\_ 4. Click the green **Submit** arrow to POST the message to `BookingServiceWSPProxy`.
- \_\_\_ 5. The response tab should show a booking response. Notice the `ConfirmationText` element and its contents.

```
<?xml version='1.0' encoding='UTF-8'>
<book:BookingResponse>
  <book:ConfirmationText>Processed 0112459898A</book:ConfirmationText>
  <book:Booking>
    <book:ReservationCode>JK99V16I</book:ReservationCode>
    <book:BookingType>I</book:BookingType>
  </book:Booking>
</book:BookingResponse>
```

## 2.6. Configure message decryption on the web service proxy

In this section, you configure the WS-Proxy to handle encrypted messages.

Using XML encryption to encrypt the message body also encrypts the operation name. When the web service proxy receives an encrypted message body, it cannot determine the operation to call unless the message is decrypted on receipt. A **Decrypt Key** specification is available within the WS-Proxy that can decrypt the message within certain conditions. This decryption occurs *before* any validation or policy is processed.

- \_\_\_ 1. Configure the web service proxy to decrypt the request message body by using a crypto key object.
  - \_\_\_ a. Click the **Proxy Settings** tab on the Configure Web Service Proxy page.
  - \_\_\_ b. This page contains general configuration information. From the **Decrypt Key** list, select **KeysFromDP**. This object references the private key that you created in an earlier exercise.

### General Configuration

Comments

Type

- ☐ Dynamic Backend  
☐ Static Backend  
☒ Static from WSDL \*

Decrypt Key

KeysFromDP



EncryptedKeySHA1 Cache Lifetime

0



### Note

This key attempts to decrypt any encrypted messages that enter the web service proxy. If either the root node of the message or the first child of the SOAP body is `<EncryptedData ...>`, the proxy automatically decrypts the encrypted payload.

- \_\_\_ c. Click **Apply**.
- \_\_\_ 2. Use a stylesheet to remove the `<wsse:signature>` header from the `BookTravel` message after it is decrypted.
  - \_\_\_ a. Click the **Policy** tab.
  - \_\_\_ b. Expand the WSDL policy tree to expose the **BookTravel** operation.
  - \_\_\_ c. Click **Processing Rules** underneath the operation.

- \_\_\_ d. In the policy editor section of the page, click **New Rule**. A default Match action appears on the rule configuration path.
- \_\_\_ e. Set the Rule Direction to **Client to Server**.
- \_\_\_ f. Drag a **Transform** action to the path.
- \_\_\_ g. Double-click the **Transform** icon to configure it.
- \_\_\_ h. Select `store:///` as the subdirectory in which the transform file is located.
- \_\_\_ i. Select `strip-wssec-signature.xml` as the transform file to be used.



- \_\_\_ j. Click **Done**.
- \_\_\_ k. Click **Apply** for the web service proxy.

## 2.7. Test the web service proxy processing with decryption added

An encrypted SOAP message is sent to the `BookingServiceWSPProxy` web service proxy by using the HTTP protocol.

- \_\_\_ 1. In SoapUI, open **17 – WSP Crypto Request**.

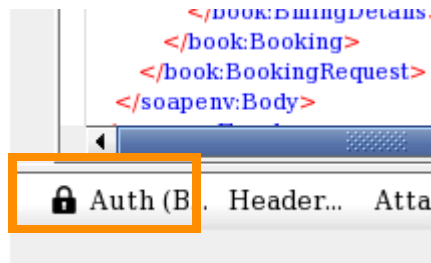


### Attention

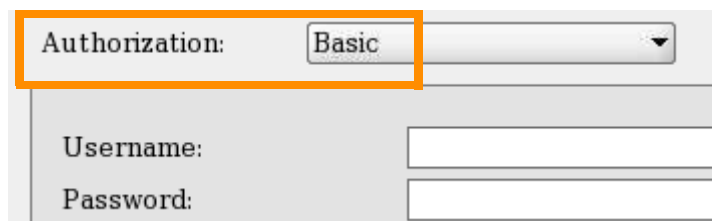
In the request tab, do *not* “Format XML”.

- \_\_\_ 2. Before sending the request, review the authorization header to verify that message encryption is on.

- \_\_\_ a. Click the **Authorization** tab that is at the bottom on the request tab.



- \_\_\_ b. Confirm that **Authorization** is set to Basic.



- \_\_\_ c. Scroll down the Authorization window, and confirm **Outgoing WSS:** is set to `Encrypt`.

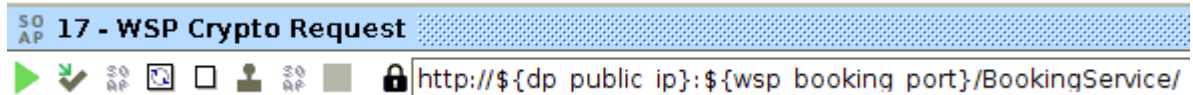




**Note**

The **Booking Request that is sent** to the `BookingServiceWSPProxy` is encrypted with the gateway's public key or certificate. DataPower receives the message and decrypts the content by using its private key.

- \_\_\_ 3. Ensure that the endpoint URL appears as:  
`http://${dp_public_ip}:${wsp_booking_port}/BookingService/`



- \_\_\_ 4. Click the green **Submit** arrow to POST the message to `BookingServiceWSPProxy`.  
 \_\_\_ 5. The response tab shows a valid booking response.

## 2.8. Use the probe to observe the encrypted request message processing

You can use the probe to see what processing occurs within the request and response rule processing.

- \_\_ 1. Within the `BookingServiceWSPProxy`, click **Show Probe**.
- \_\_ 2. Click **Enable Probe**.
- \_\_ 3. Click **Close** to close the confirmation window.
- \_\_ 4. In SoapUI, resend **17 – WSP Crypto Request**.
- \_\_ 5. Click **Refresh** on the probe transaction list window.
- \_\_ 6. You should see an entry for the submitted request.
- \_\_ 7. Expand the entry, and click the magnifying glass for the **request rule**.
- \_\_ 8. In the probe window, click the **Content** tab.



p 1: Transform with XSLT style sheet Action:Input=INPUT, Transform=store:///strip-wsse  
 Output=OUTPUT, NamedInOutLocationType=default, OutputType=default, Transactional  
 SourceType=static, Asynchronous=off, ResultsMode=first-available, RetryCount=0, RetryI  
 IteratorType=XPATH, Timeout=0, MethodRewriteType=GET, MethodType=POST,

**Content**

Headers

Attachments

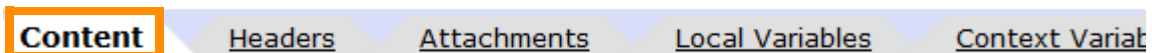
Local Variables

Context Variables

**Content of context 'INPUT':**

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope
  xmlns:book="http://www.ibm.com/datapower/FLY/BookingService/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
>
  <soapenv:Header>
    <wsse:Security
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-ws
      1.0.xsd"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-ws
      1.0.xsd"
    />
  </soapenv:Header>
  <soapenv:Body>
    <book:BookingRequest>
      <book:Booking>
        <book:ReservationCode>TK00W16T</book:ReservationCode>
      </book:Booking>
    </book:BookingRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

- \_\_\_ 9. Notice that the message contains the **wsse:Security** header. This header is present because of the WS-Security encryption that occurred inside SoapUI during the request processing. Why does the message body content appear unencrypted? Recall that in the web service proxy, the decrypt specification is configured outside of the request processing rule. Therefore, the web service proxy already decrypted the message before the request rule is executed.
- \_\_\_ 10. Click the second magnifying glass that is located after the **Transform** action.



#### Content of context 'OUTPUT':

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:book="http://www.ibm.com/datapower/FLY/BookingService/"
>
  <soapenv:Header />
  <soapenv:Body>
    <book:BookingRequest>
      <book:Booking>
        <book:ReservationCode>JK99V16I</book:ReservationCode>
        <book:BookingType>I</book:BookingType>
        <book:PaymentCardDetails>

```

- \_\_\_ 11. Observe that the message no longer contains the **wsse:Security** header. This fact is because the transform action stripped off the header.
- \_\_\_ 12. Close the probe transaction window.
- \_\_\_ 13. In the probe transaction list window, click **View Log**.
- \_\_\_ 14. Scan the entries in the log to find anything that is related to decrypting the message on receipt.
- \_\_\_ 15. No entries detail that service-level decryption occurred. How can you verify that you even received an encrypted message in DataPower? An option is to enable **XML File Capture** in the **default** domain. This facility captures the XML messages as they arrive at the gateway before they any service processes them. Because it can capture much traffic, the XML File Capture is usually limited to specific personnel. Check with your DataPower administrator for details on your gateways.
- \_\_\_ 16. Close the log window.
- \_\_\_ 17. Click **Disable Probe**.
- \_\_\_ 18. Click **Close** to close the disable probe confirmation window.
- \_\_\_ 19. Click **Close** to close the probe transaction list window.

- \_\_\_ 20. You can see the encrypted message as it leaves SoapUI. Click **http log** at the bottom of the SoapUI window.
- \_\_\_ 21. Scroll to the bottom of the log. These entries show the response message that is received from DataPower. Notice that it is a `<book:BookingResponse>`.
- \_\_\_ 22. Scroll further up to the next set of entries. These entries are for the request that is sent from SoapUI to DataPower. You cannot see the `<book:BookingRequest>` element because it is encrypted. Instead, the `<xenc:EncryptedData>` element replaces it.

```

< ?xml version="1.0" encoding="UTF-8" ?>
> "POST /BookingService/ HTTP/1.1[\r][\n]"
> "Accept-Encoding: gzip,deflate[\r][\n]"
> "Content-Type: text/xml;charset=UTF-8[\r][\n]"
> "SOAPAction: \"http://www.ibm.com/datapower/FLY/BookingService/BookTravel\""
> "Content-Length: 3786[\r][\n]"
> "Host: 172.16.78.24:12325[\r][\n]"
> "Connection: Keep-Alive[\r][\n]"
> "User-Agent: Apache-HttpClient/4.1.1 (java 1.5)[\r][\n]"
> "[\r][\n]"
> "<soapenv:Envelope xmlns:book=\"http://www.ibm.com/datapower/FLY/BookingService/BookTravel\" xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\">"
> "  <soapenv:Header><wsse:Security xmlns:wsse=\"http://docs.oasis-open.org/wss/2004/01/ws-security.xsd\">"
> "    <soapenv:Body><xenc:EncryptedData Id="ED-9A5DDE9001B47421E614260" Type="http://www.w3.org/2001/04/xmlenc#data" Base64="base64:...">"
> "</soapenv:Body></wsse:Security></soapenv:Header>"
> "</soapenv:Envelope>"
< "HTTP/1.1 200 OK[\r][\n]"

```

- \_\_\_ 23. Close the SoapUI window.
- \_\_\_ 24. In the WebGUI, click **Save Configuration**.

## End of exercise

## Exercise review and wrap-up

In this exercise, you added a WSDL file for the `BookingServiceWSProxy` web services. The WSDL file contains the endpoint address of the web service. Using the web service proxy, you created a virtual address for the service, which the client calls to invoke the web service. You also added rules to a processing policy to convert the confirmation code into a more readable confirmation text.

You also configured operational level processing by using a series of check boxes that control the web service proxy behavior.

Finally, decryption was added to the web service proxy, not by using a **Decrypt** action on a rule in a processing policy, but by specifying it at the service level.

You used SoapUI to send up the appropriate SOAP message, with and without encryption.

# Appendix A. Exercise solutions

This appendix describes:

- The dependencies between the exercises and other tools.
- How to load the sample solution configurations for the various exercises. The solutions were exported from the gateway into a .zip file. You can import a sample solution into your domain.

## Part 1: Dependencies

Certain exercises depend on previous exercises and on other resources, such as the need for the back-end application server to support service calls. The back-end application server is a multi-protocol gateway that runs in another application domain within the DataPower gateway.

*Table 1. Dependencies*

Exercise	Depends on exercise	Uses cURL	Uses SoapUI	Uses Baggage web service	Uses Booking web service
1: Web service encryption and digital signatures		No	Yes	No	Yes
2: Configuring a web service proxy	1	No	Yes	No	Yes

If the class is using the standard images and setup, the LDAP server is running on the student image. The Baggage and Booking services are running as services on the DataPower gateway. Therefore, each student is using a different IP address for the student image. Assuming that each student has a separate DataPower virtual gateway, each student also has different IP addresses for the gateway.

If the exercises are run in the IBM remote lab environment, like Skytap, the IP addresses might be the same for each student because each student has a unique entry point into the virtualized environment.

## Part 2: Importing solutions



### Note

The solution files use port numbers that might already be in use. You must change the port numbers of the imported service. You might also find it necessary to update the location of the back-end application server that provides the web services.

\_\_ 1. Determine the .zip file to import from the following table:

Table 2. Exercise solution files

Exercise	Compressed solution file name
1: Web service encryption and digital signatures	XML_WSSecurity.zip
2: Configuring a web service proxy	XML_WSProxy.zip

- \_\_ a. The .zip file names begin with the naming convention XML\_xyz, where “xyz” represents the particular lab exercise.
- \_\_ b. To import a solution to begin a new exercise, import the solution for the previous exercise. Import the .zip solution file into your application domain.
- \_\_ c. From the **Control Panel**, in the vertical navigation bar, click **Administration > Configuration > Import Configuration**.
- \_\_ d. Make sure that the selection for **From** is **ZIP Bundle** and the selection for **Where** is **File**.
- \_\_ e. Click **Browse** and navigate to your respective .zip solution file.
- \_\_ f. Click **Next**.
- \_\_ g. In the next page, leave the files selected. Scroll down and click **Import**.
- \_\_ h. Make sure that the import is successful. Click **Done**.
- \_\_ 2. *Be sure to update the port numbers and application server location to your local values. Because private keys (key files) are not exported, you also must create keys and certificates.* In some exercise solutions, the key files are exported in the `local:` directory. After import, you move those files into the `cert:` directory.
- \_\_ 3. The lab exercises call one back-end web service, **Booking Service**. This web service is in the FLY service domain. To do the labs on another DataPower gateway, be sure to import the `dev_FLYservices_domain.zip` file into the **FLYServices** domain.

# Appendix B. Lab environment setup

This appendix instructs how to set up the lab environment, including:

- Defining the literal variable values in SoapUI
- Testing the Booking and Baggage web service back ends
- Identifying the IP address of your student image
- Populating a convenient table with all the required variables that are used in this course

## ***Part 1: Configure the SoapUI variables for use***

The SoapUI tool supports specification of properties to reduce the redundant entry of the same value for testing. For these exercises, the client testing usually accesses the public interface of the student-created DataPower services. Rather than requiring the students to constantly enter the same value, the public IP address of the gateway is configured as a SoapUI property.

1. Obtain the required variables for this course. The variable information can be found in at least one of the following locations, based on the type of course you are taking:
  - On the image desktop as a background
  - On the image background from the SPVC you logged in to
  - In an email you received with instructions for this course
  - From your instructor if you are in a classroom (virtual or literal) environment
  - In the exercise guide itself

The variables are:

- The DataPower gateway's public IP address `<dp_public_ip>`
- The DataPower gateway's internal IP address `<dp_internal_ip>`
- Your student image IP address `<image_ip>`
- Your student number (it is a two-digit number) `<nn>`

2. Open SoapUI by using the icon on the desktop.

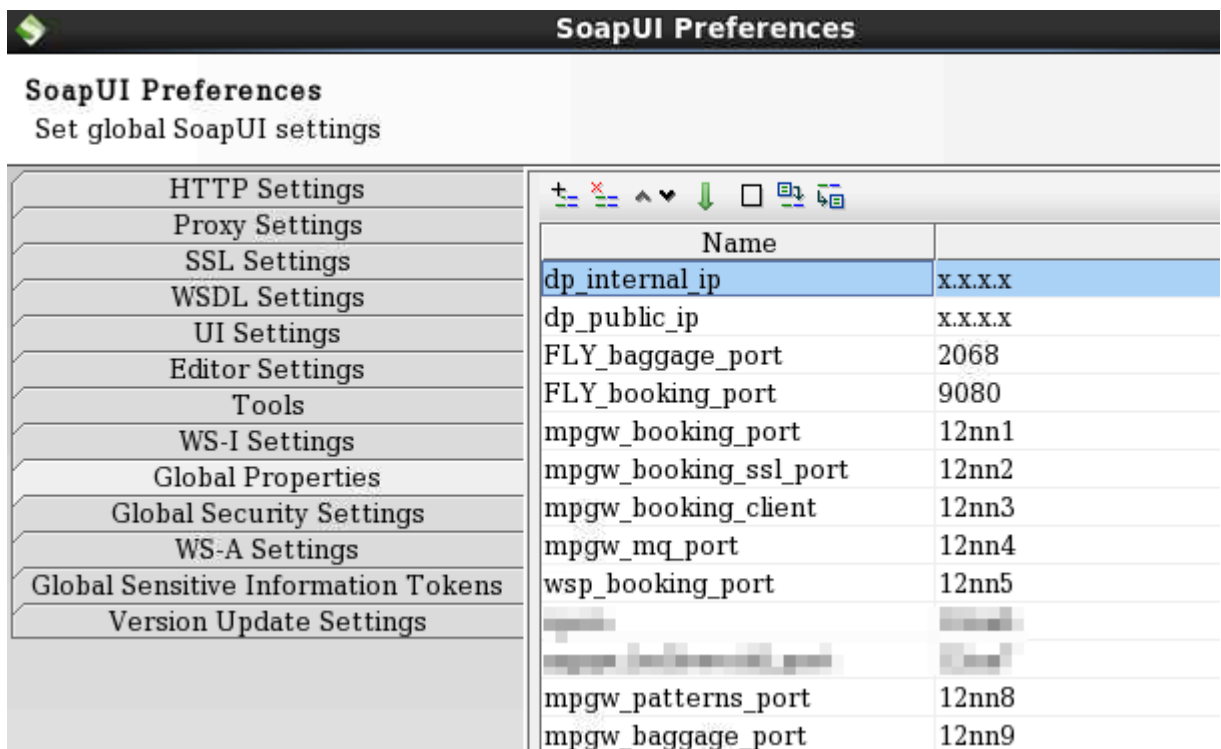


### **Attention**

If you get a “new version available” message, close the message window and do not download or install any upgrades.



- \_\_\_ 3. Click **File – Preferences**.
- \_\_\_ 4. Click the **Global Properties** choice.
- \_\_\_ 5. Update the values for the following variables.
  - \_\_\_ a. Double-click the **Value** cell for the **dp\_internal\_ip** property.
  - \_\_\_ b. Replace the value (`x.x.x.x` or `1.2.3.4`) with the literal value of the `<dp_internal_ip>` address in the cell. That is, replace `1.2.3.4` with the IP address of the DataPower gateway that is being used for your class.
  - \_\_\_ c. Click **Enter** while the cursor is in the cell. This action registers the new value.
  - \_\_\_ d. Double-click the **Value** cell for the **dp\_public\_ip** property.
  - \_\_\_ e. Replace the value (`x.x.x.x` or `1.2.3.4`) with the literal value of the `<dp_public_ip>` address in the cell. That is, replace `1.2.3.4` with the IP address of the DataPower gateway that is being used for your class.
  - \_\_\_ f. Click **Enter** while the cursor is in the cell. This action registers the new value.
  - \_\_\_ g. Double-click the **Value** cell for the **mpgw\_booking\_port** property.
  - \_\_\_ h. Replace “`nn`” with your appropriate student number. For example, if you are student 01, the value for `mpgw_booking_port` of `12nn1` is updated to `12011`.
  - \_\_\_ i. Click **Enter** while the cursor is in the Value cell. This action registers the new value.



- \_\_\_ j. Repeat the previous steps g – i for the remaining values.
- \_\_\_ k. Click **OK**.
- \_\_\_ l. Click **File > Save Preferences**.

SoapUI is now configured for all exercises in this course. The messages that are sent to DataPower when using SoapUI reference these variables. No further SoapUI configuration is required, unless stated in the specific exercise.

When SoapUI recognizes the `dp_public_ip` reference in a request ("`${dp_public_ip}`"), it substitutes the correct IP address into the URL.



## Part 2: Confirm that the Booking and Baggage web services are up

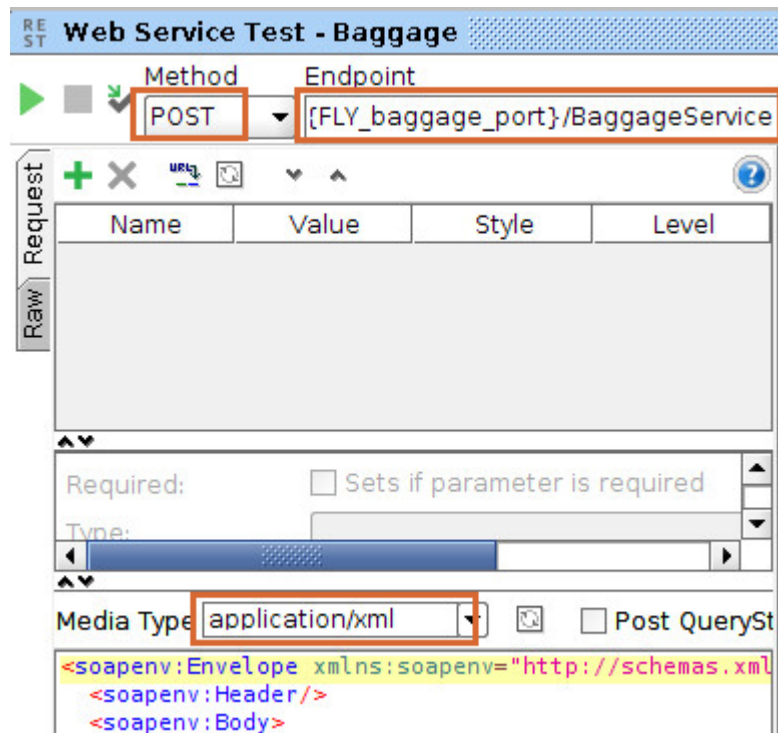
Test the Booking web service and the Baggage web service. The following steps ensure that the back-end web services are operational. In addition to testing the availability of the web service, it is also a useful troubleshooting technique to verify network connectivity to the back-end web service.

- \_\_\_ 1. In the project tree, expand the **BaggageServices** project until **Web Service Test – Baggage** is visible.

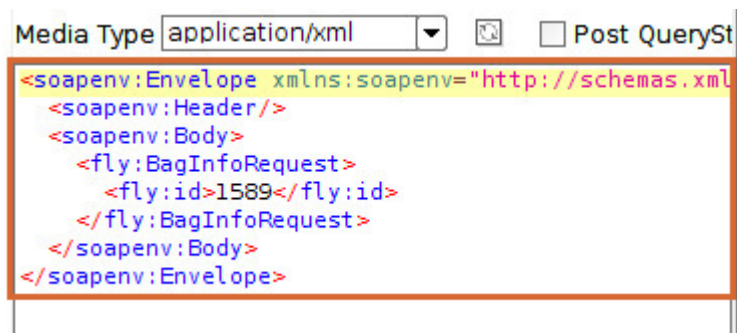


- \_\_\_ 2. Double-click **Web Service Test – Baggage** to open the request window. If a double-click does not work, right-click the request and click **Show Request Editor**.

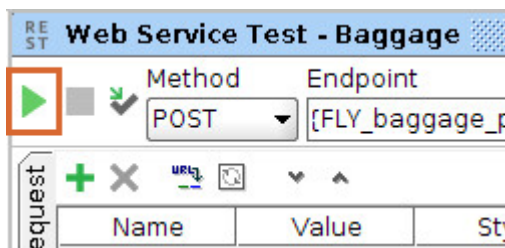
- \_\_\_ 3. Ensure that the following information is preconfigured in the request message:
- Method: **POST**
  - Endpoint: `http://${dp_internal_ip}:${FLY_baggage_port}/BaggageService`
  - Media Type: **application/xml**



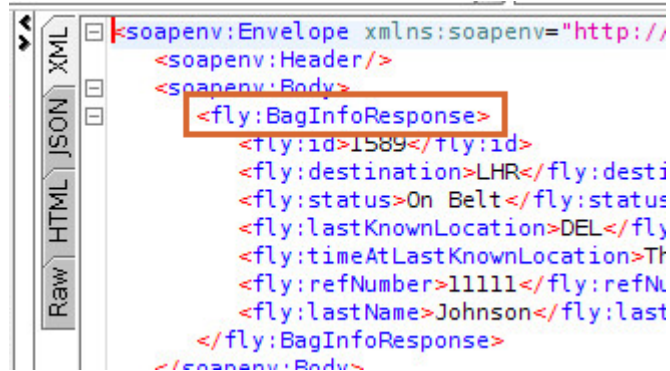
- SOAP message:



- \_\_\_ 4. Click the green **Submit** arrow to send the request message directly to the **BaggageService** web service for FLY airlines.



- \_\_\_ 5. Confirm that a successful **BagInfoResponse** response is returned in the response tab.



```

<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <fly:BagInfoResponse>
      <fly:id>1589</fly:id>
      <fly:destination>LHR</fly:desti
      <fly:status>On Belt</fly:status
      <fly:lastKnownLocation>DEL</fly
      <fly:timeAtLastKnownLocation>Th
      <fly:refNumber>11111</fly:refN
      <fly:lastName>Johnson</fly:last
    </fly:BagInfoResponse>
  </soapenv:Body>
</soapenv:Envelope>
  
```



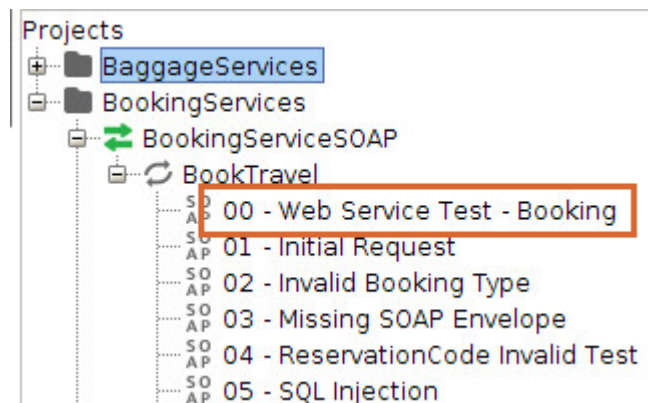
### Important

If you do not get the correct response, the failure can be due to several reasons:

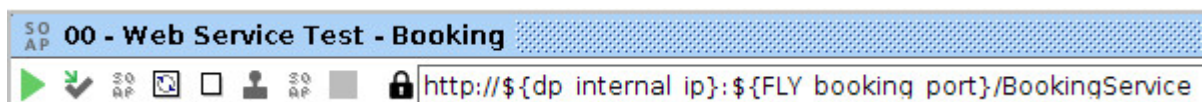
- The variables that are entered in SoapUI General Preferences are not installed on the DataPower gateway.
- The DataPower gateway is unreachable from your student image due to some network connectivity issue.

Verify that you entered the correct values for the SoapUI variables. If the values are correct, escalate for assistance.

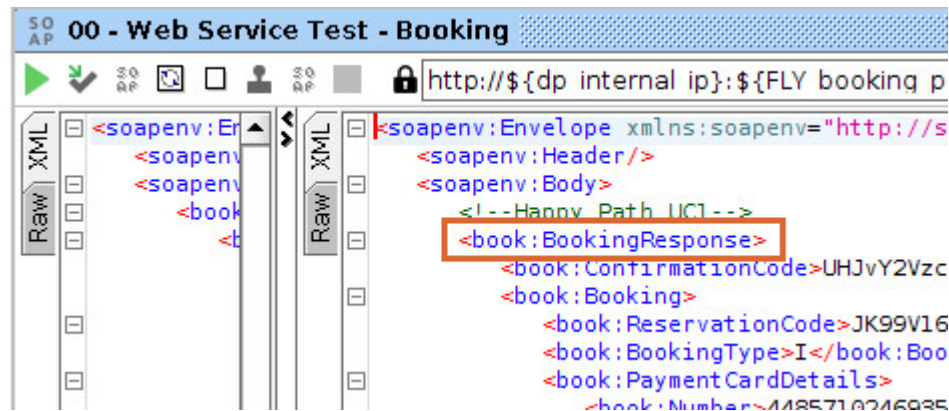
- \_\_\_ 6. Close the **Web Service Test – Baggage** window.
- \_\_\_ 7. In the project tree, expand the **BookingServices** project until **00 – Web Service Test – Booking** is visible.



- \_\_\_ 8. Double-click **00 – Web Service Test – Booking** to open the request window. If a double-click does not work, right-click the request and click **Show Request Editor**.
- \_\_\_ 9. Confirm that the URL address field contains:  
`http://${dp_internal_ip}:${FLY_booking_port}/BookingService`



- \_\_\_ 10. Click the green **Submit** arrow that is to the left of the URL address field to send the SOAP request test message directly to the FLY Airlines Booking web service.
- \_\_\_ 11. If everything worked properly, you should see the `<book:BookingResponse>` XML tree in the Response tab.



### Important

If you do not get the correct response, the failure can be due to several causes:

- The variables that are entered in SoapUI General Preferences are wrong.
- The FLYService domain that contains the web services is not installed on the DataPower gateway.
- The DataPower gateway is unreachable from your student image due to a network connectivity issue.

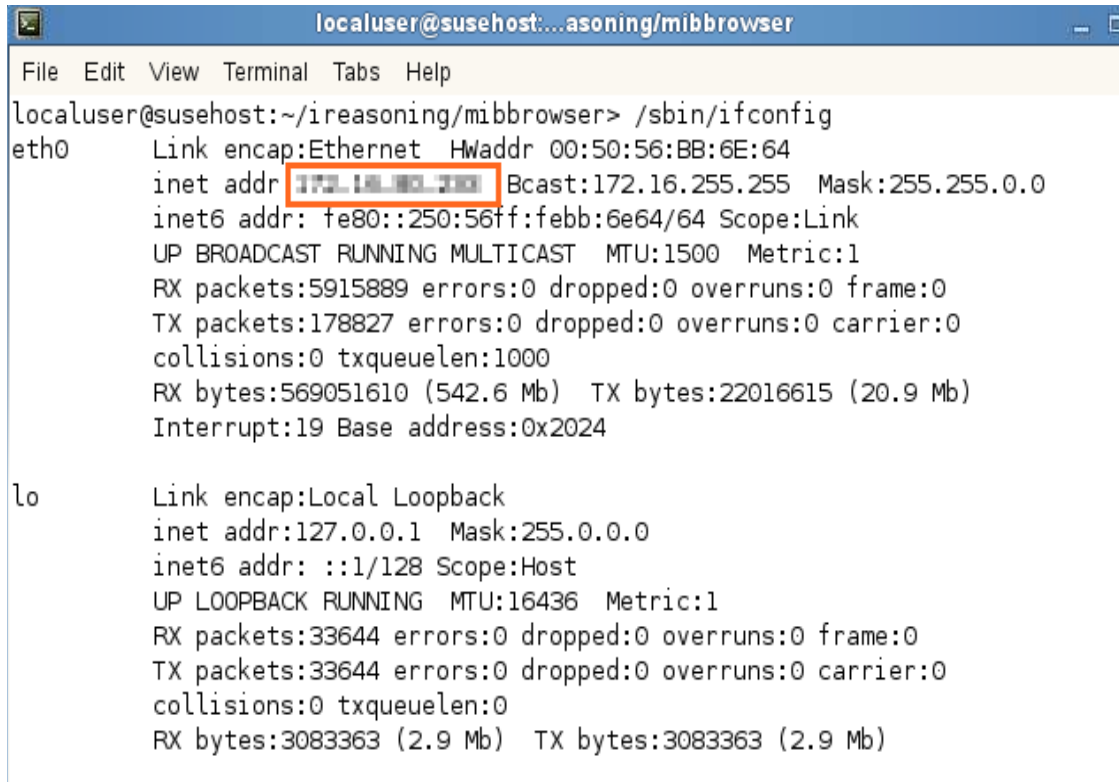
Verify that the values that you entered for the SoapUI variables are correct. If you still have problems, you must contact whatever support you have for the class.

- \_\_\_ 12. Close the **00 – Web Service Test – Booking** window.

## Part 3: Identify the student image IP address

On Linux, you can discover the IP address by running the `ifconfig` command. Open a terminal window. The terminal window is available from the icon on the desktop. From within a terminal window, run the `ifconfig` command that includes the full path, as follows:

```
> /sbin/ifconfig
```



```

localuser@susehost:~/soneing/mibbrowser
File Edit View Terminal Tabs Help
localuser@susehost:~/soneing/mibbrowser> /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:BB:6E:64
          inet addr: 172.16.10.200  Bcast:172.16.255.255  Mask:255.255.0.0
          inet6 addr: fe80::250:56ff:febb:6e64/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5915889 errors:0 dropped:0 overruns:0 frame:0
          TX packets:178827 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:569051610 (542.6 Mb)  TX bytes:22016615 (20.9 Mb)
          Interrupt:19 Base address:0x2024

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:33644 errors:0 dropped:0 overruns:0 frame:0
          TX packets:33644 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3083363 (2.9 Mb)  TX bytes:3083363 (2.9 Mb)

```

When the IP address of the local student image is obtained, update the information in table B1 for the variable `<image_ip>`.

Close the terminal window.

## Part 4: Port and variable table values

If you want to have a single reference for all variables that are used in this course, the following table is supplied. You might want to tear these two pages out of your book, or if you have a PDF file, you can print both pages as a quick reference point.

\_\_\_ 1. Complete the following table with the values that are supplied by the instructor.

Table B-1. Developers course variable and port assignments table

Object	Value (default)
<b>Lab files location</b>	
<lab_files> Location of student lab files for this course	/usr/labfiles/dp
<b>Student information</b>	
<nn>	
<studentnn>	
<studentnn_domain>	
<studentnn_password>	student<nn>
<studentnn_updated_password>	
<image_ip> IP address of the student image	
<b>Logins that are not DataPower</b>	
<linux_user>	localuser
<linux_user_password>	passw0rd
<linux_root_user>	
<linux_root_password>	passw0rd
<b>DataPower information</b>	
<dp_public_ip> IP address of the public services on the gateway	
<dp_internal_ip> IP address of the DataPower gateway development and administrative functions	
<dp_WebGUI_port> Port number for the WebGUI	9090
<dp_FLY_baggage_port>	2068
<dp_FLY_booking_port>	9080
<b>Server information</b>	
<SoapUI_keystores>	/usr/labfiles/dp/WSecurity/Client.jks

<SoapUI_keystores_password>	myjkspw
<ldap_password>	passw0rd
<ldap_server_root_dir>	/var/lib/ldap/ibm-com/
<ldap_user_name>	cn=admin,dc=ibm,dc=com
<http_server_port>	80
<http_server_root_dir>	/var/www/html/
<logger_app_port>	1112
<b>Student-built DataPower services</b>	
<mpgw_booking_port>	12nn1
<mpgw_booking_ssl_port>	12nn2
<mpgw_ssl_booking_port>	12nn3
<mpgw_mq_port>	12nn4
<wsp_booking_port>	12nn5
<mpgw_helloworld_port>	12nn7
<mpgw_patterns_port>	12nn8
<mpgw_baggage_port>	12nn9
<oidc_provider_port>	7nn5
<oidc_client_port>	7nn4

## End of appendix





IBM Training

