



*Accelerate, Secure and  
Integrate with WebSphere  
DataPower SOA Appliances  
V5*

(Course code WE401)

**Student Exercises**

ERC 3.0

Authorized

**IBM | Training**

WebSphere Education

## Trademarks

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

Approach®	BladeCenter®	DataPower®
DB™	DB2®	developerWorks®
Domino®	IMS™	Lotus®
RACF®	Rational®	RDN®
Redbooks®	Tivoli®	U®
WebSphere®	z/OS®	zSeries®

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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.

### March 2014 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

# Contents

<b>Trademarks .....</b>	vii
<b>Exercises description .....</b>	xi
<b>Verifying the image and course materials .....</b>	xv
<b>Exercise 1. Exercise setup .....</b>	1-1
1.1. Define variables and ports .....	1-3
1.2. Upload the crypto key and certificate files .....	1-6
1.3. Upload the WSDL files to the appliance .....	1-8
1.4. Launch Eclipse .....	1-9
1.5. Import the East and West Address WSDL files into Eclipse .....	1-10
1.6. Verify cURL and Open SSL installation .....	1-14
<b>Exercise 2. Creating XSL transformations .....</b>	2-1
2.1. Examine the original XML file .....	2-3
2.2. Create an XSLT style sheet to print entries in a DN element .....	2-8
2.3. Modify domain.xsl to selectively print based on attributes .....	2-15
2.4. Use a substring function to print only the CN .....	2-17
2.5. Test a style sheet by using the Interoperability Test Service .....	2-19
2.6. (Optional) Modify the templates to ignore namespace .....	2-20
<b>Exercise 3. Creating a simple XML firewall .....</b>	3-1
3.1. Validation: XML firewall creation .....	3-5
3.2. Validation: XML firewall examination and configuration .....	3-10
3.3. Validation: XML firewall testing .....	3-18
3.4. Transformation: XML firewall creation .....	3-21
3.5. Transformation: XML firewall configuration .....	3-22
3.6. Transformation: XML firewall testing .....	3-27
3.7. Integration: Edit the MyBasicFirewall policy .....	3-31
3.8. Integration: XML firewall testing .....	3-32
<b>Exercise 4. Creating an advanced multi-protocol gateway .....</b>	4-1
4.1. Create a basic MPGW to validate SOAP messages .....	4-5
4.2. Create a multi-protocol gateway for WestAddressSearch .....	4-29
4.3. Configure an HTTP front side protocol handler .....	4-35
4.4. Create an MPGW for content based routing .....	4-37
4.5. Perform end-to-end content based routing scenario testing .....	4-45
<b>Exercise 5. Adding error handling to a service policy .....</b>	5-1
5.1. Adding an error rule to the policy .....	5-3
<b>Exercise 6. Creating cryptographic objects .....</b>	6-1
6.1. Generate a certificate-key pair on the DataPower appliance .....	6-3

6.2. Creating cryptographic objects .....	6-8
6.3. Import a certificate-key pair onto the DataPower appliance .....	6-12
<b>Exercise 7. Using SSL to secure connections .....</b>	<b>7-1</b>
7.1. Verify web service behavior .....	7-4
7.2. Add an HTTPS handler to the EastAddressSearch service .....	7-5
7.3. Test the EastAddressSearch HTTPS access .....	7-7
7.4. Create the SSL client support in the AddressRouter service .....	7-8
7.5. Test the SSL connection from the AddressRouter to the EastAddressSearch ..	7-11
<b>Exercise 8. Protecting against XML threats .....</b>	<b>8-1</b>
8.1. Recursive entity attack simulation .....	8-5
8.2. Recursive entity threat protection test .....	8-8
8.3. Test the multi-protocol gateway .....	8-15
8.4. Excessive attribute count threat protection .....	8-16
8.5. SQL injection attack prevention .....	8-19
<b>Exercise 9. Configuring a web service proxy .....</b>	<b>9-1</b>
9.1. Create a web service proxy for EastAddressSearch web service .....	9-5
9.2. Add a WSDL to the web service proxy for the WestAddressSearch web service ..	9-9
9.3. Verify the generated components .....	9-10
9.4. Test the EastAddressSearch web service .....	9-15
9.5. Test the WestAddressSearch web service .....	9-20
9.6. Add an operation-level rule to West Address Search web service proxy .....	9-22
<b>Exercise 10. Web service encryption and digital signatures .....</b>	<b>10-1</b>
10.1. Create cryptographic objects .....	10-4
10.2. Examine the East Address Search web service proxy .....	10-6
10.3. Create a multi-protocol gateway to encrypt messages .....	10-7
10.4. Create a rule to sign messages .....	10-15
10.5. Configure message decryption on the web service proxy .....	10-17
10.6. Configure field-level message decryption on the web service proxy .....	10-20
10.7. Configure message verification on the web service proxy .....	10-24
10.8. Send a signed and encrypted message to the web service proxy .....	10-28
<b>Exercise 11. Web service authentication and authorization .....</b>	<b>11-1</b>
11.1. Test the East Address Search web service .....	11-4
11.2. Configure a AAA policy action for a web service operation .....	11-7
11.3. Test the access control policy for a web service operation .....	11-14
11.4. Test the access control policy with a web services security user name token	11-17
11.5. Configure the access control policy to handle SAML assertions .....	11-19
11.6. Test the access control policy with a SAML assertion .....	11-23
<b>Exercise 12. Creating message counter monitors for a AAA policy .....</b>	<b>12-1</b>
12.1. Test the East Address Search web service .....	12-4
12.2. Configure a AAA policy action for message counter .....	12-5
12.3. Test the East Address Search web service count monitor .....	12-11

12.4. Configure a AAA policy action for message rejection that is based on message counter .....	12-12
12.5. Test the East Address Search web service reject count monitor .....	12-17
<b>Exercise 13. Creating a AAA policy by using LDAP .....</b>	<b>13-1</b>
13.1. Obtain the IBM Tivoli Directory Server connectivity information .....	13-4
13.2. Add directory entries into IBM Tivoli Directory Server .....	13-5
13.3. Use LDAP to create a AAA policy .....	13-10
<b>Exercise 14. Implementing an SLM monitor in a web service proxy .....</b>	<b>14-1</b>
14.1. Test the existing AddressSearchProxy by using the test script .....	14-4
14.2. Create a log target for SLM log messages .....	14-8
14.3. Add SLM criteria to the web service proxy .....	14-10
14.4. Run the test script with SLM criteria in effect .....	14-12
14.5. Add an SLM action to a port-operation request rule .....	14-15
14.6. Run the test script with the operation-level SLM action .....	14-17
14.7. View the SLM policy object .....	14-18
14.8. Examine the graph behavior (optional) .....	14-19
<b>Exercise 15. Configuring a multi-protocol gateway service with WebSphere MQ</b>	<b>15-1</b>
15.1. Obtain the WebSphere MQ configuration .....	15-4
15.2. Create a WebSphere MQ front side handler (FSH) .....	15-5
15.3. Create a multi-protocol gateway that completes one-way messaging .....	15-9
15.4. Use the WebSphere MQ FSH to call the EastAddressSearch web service .....	15-17
15.5. Configuring transaction capability on the DataPower WebSphere MQ queue manager .....	15-19
<b>Appendix A. Creating a firewall and HTTP proxy for a web application.....</b>	<b>A-1</b>
A.1. Examine existing East Address web application .....	A-4
A.2. Create a web application firewall .....	A-6
A.3. Examine the objects that the web application firewall generated .....	A-13
A.4. Configure the web application security policy .....	A-16
<b>Appendix B. Configuring WebSphere JMS .....</b>	<b>B-1</b>
B.1. Using the WebSphere JMS transport .....	B-4
B.2. Create a multi-protocol gateway to invoke a web service over JMS .....	B-6
<b>Appendix C. Exercise solutions .....</b>	<b>C-1</b>



# 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® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

Approach®	BladeCenter®	DataPower®
DB™	DB2®	developerWorks®
Domino®	IMS™	Lotus®
RACF®	Rational®	RDN®
Redbooks®	Tivoli®	U®
WebSphere®	z/OS®	zSeries®

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

## Address case study

The exercises in this course build upon a common case study: the AddressSearch web service. Two web services that are running on WebSphere Application Server are provided, East Address web service and West Address web service.

Both services are identical except for their data. The East Address web service contains address data for people in the eastern United States, and the West Address web service contains data for people in the western United States.

Each service supports the following operations:

- `findByName(name)` returns a single mailing address that is based on an exact match on a person's social title, given name, and surname. Mr., Mrs., and Ms. are examples of social titles.
- `findByLocation(String city, String state)` takes in two parameters and returns an array of addresses that match all of the fields exactly. The city field is optional, but the state is mandatory. For example, if all fields are left blank other than `<state>CA</state>`, all entries within California are returned, regardless of street, city, or zip code.
- `findByDetails(city, state)` returns a list of mailing addresses that match the city and state. If the city field is omitted, this operation retrieves all mailing addresses in that state.
- `retrieveAll()` returns the entire list of mailing addresses that the service maintains.

The two endpoints are:

- East Address web service:  
`http://server:9080/EastAddress/services/AddressSearch/`

- West Address web service:  
`http://server:9080/WestAddress/services/AddressSearch/`

Technically, the AddressSearch application is a self-contained web application capable of generating a list of fictitious but convincing names and addresses for any city in the United States. The seed values for the name, street, city, state, and zip code are based on the most common values that are found in previous United States Census surveys.

This application minimizes its dependencies on data sources by relying on a list of addresses from a flat file.

The list of addresses can represent a healthcare provider, a financial institution, or a government agency. In most countries, privacy laws and regulations stipulate that such data must be kept confidential. The overall purpose of these exercises is to use IBM WebSphere DataPower SOA Appliance to restrict access to this web service.

This course includes the following exercises:

1. **Exercises setup:** Assigns a set of port numbers to each student and sets up the environment for the subsequent exercises in the course.
2. **Creating XSL transformations:** Shows students how to create an XSLT file to do XML transformations. The XSLT file is then tested within an XML firewall.
3. **Creating a simple XML firewall:** Creates an XML firewall that does validation and transformation actions.
4. **Creating an advanced multi-protocol gateway:** Creates a multi-protocol gateway (MPGW) for both the East and West Address web services. Students also create a third MPGW that does content-based routing to the East or West Address MPGW. Steps for reviewing the log and for using the multistep probe are included.
5. **Adding error handling to a service policy:** Uses either the **On Error** action or the error rule to prepare for problems.
6. **Creating cryptographic objects:** Creates the private and public key objects that are used in cryptographic functions. The cryptographic objects that are used in SSL communications are created and configured from the public and private key objects.
7. **Using SSL to secure connections:** Uses the cryptographic objects that are created in the previous exercise to secure communications to and from the DataPower appliance.
8. **Protecting against XML threats:** Simulates a series of XML-based attacks on a newly created MPGW. Malformed XML documents and hidden SQL statements are examples of attacks on an XML application.
9. **Configuring a web service proxy:** Virtualizes both the East and West Address search web services.
10. **Web service encryption and digital signatures:** Demonstrates the support for web service security, XML encryption, and XML digital signatures through processing rule actions that are applied to the web service proxy.
11. **Web service authentication and authorization:** Restricts access to the `retrieveAll` web service operation by using a security token in a web services security header and a security assertion that is written in the Security Assertion Markup Language (SAML).
12. **Creating message counter monitors for a AAA policy:** Demonstrates the use of a message counter.

13. **Creating a AAA policy by using LDAP:** Uses an LDAP server to create a AAA policy for authentication and authorization of clients.
14. **Implementing an SLM monitor in a web service proxy:** Shows the use of an SLM monitor to track and manage login requests.
15. **Configuring a multi-protocol gateway service with WebSphere MQ:** Adds the use of a front side protocol handler that uses WebSphere MQ, and a back-end transport that calls WebSphere MQ.

The following are optional exercises in an appendix section:

1. **Creating a firewall and HTTP proxy for a web application:** Creates a web application firewall service to offload authentication from the back-end application server. SSL is used to secure the communication between the client and web application firewall service. The service also validates incoming client requests.
2. **Configuring WebSphere JMS:** Creates a service that PUTs and GETs to a JMS queue on WebSphere Application Server.

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 if you have sufficient time and want an additional challenge.



# Verifying the image and course materials

The student books and VMware image for this course display a release number that is called the edition revision code (ERC). To verify that the books and image are at the same level, compare the ERC of the VMware image to the ERC of the student books.

- \_\_\_ 1. Determine the ERC number of your course materials.
  - \_\_\_ a. Open all of the books you received (either printed or PDF).
  - \_\_\_ b. Note the ERC listed on the front page of your books. The ERC number is listed under the course title on the first page of the books as in the following example:

*Accelerate, Secure and  
Integrate with WebSphere  
DataPower SOA Appliances  
V5*

(Course code WE401)

## Student Exercises

ERC 2.0

- \_\_\_ 2. Determine the ERC number of the VMware image.
  - \_\_\_ a. On the image desktop, open the **readme.txt** file.
  - \_\_\_ b. Note the ERC listed in the file. The ERC number is indicated on the “ERC number” line as in the following example:

```
-----
Student Workstation Virtual Machine Information
WebSphere Education

Copyright (C) 2013 IBM Corporation
Course materials, including this virtual machine, may not be reproduced
in whole or in part without prior written permission of IBM.
-----
```

```
Course code: WE401/VE401
Course title: Accelerate, Secure and Integrate with
              WebSphere DataPower SOA Appliances
ERC number: 2.0
Last modified date: July 4, 2013
```

```
Operating System: Linux SUSUE 11
Primary account username: localuser
Primary account password: web1sphere
```

- \_\_\_ 3. Verify the image and course materials.
  - \_\_\_ a. If the ERC number on the image does not match the ERC number on the printed materials, notify your instructor that the materials are not synchronized.



**Stop**

Do NOT proceed with the exercises if the ERC numbers on the course materials and course image do not match; ask your instructor for further direction.

# Exercise 1. Exercise setup

## What this exercise is about

In this exercise, you do work that is used in subsequent exercises. You determine the assigned variables and port numbers, import key and certificate crypto files, import WSDL files into Eclipse and the appliance, and verify that cURL and OpenSSL are installed.

## What you should be able to do

At the end of the exercise, you should be able to:

- Import the files that are used in the exercises
- Verify cURL installation
- Populate the table that contains all of the port numbers

## Introduction

In this exercise, you set up for later exercises. With instructor guidance, you determine port and other variable assignments. In a subsequent exercise, you need the Alice private key and certificate and some WSDL files; you import them into the appliance now. When testing some of the later exercises, you use Eclipse. To run the test, you now import some WSDL files into the Eclipse workspace. Later exercises require the use of cURL to make HTTP and HTTPS calls to the services you configure on the appliance; you verify its installation.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- **Eclipse**, to contain the WSDLs used in a later exercise
- **WebSphere Application Server**, if running it on the local machine
- 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 value:
  - *<lab\_files>*: location of the student lab files
  - *<dp\_internal\_ip>*: IP address of the DataPower appliance development and administrative functions
  - *<dp\_WebGUI\_port>*: port number for the WebGUI

## 1.1. Define variables and ports

The instructor assigns you a student number **nn**, which you substitute as needed in the port numbers that you define for your services on the appliance.

The instructor also supplies the values for the appliance URL, application server URL, and lab files location. You can tear this sheet from the student exercise book to use as reference for the exercises in this course.

Enter your assigned values in the table.

**Table 1-1: Variable and port assignments**

Object	Value (default)
<b>Lab files location</b>	
<lab_files>	/home/localuser/dplabs/dev
Location of student lab files for WE401	
<b>Student information</b>	
<nn>	
<studentnn>	
<studentnn_domain>	
<studentnn_password>	student<nn>
<studentnn_updated_password>	
<studentnn_image_ip>	
<b>Logins that are not DataPower</b>	
<linux_user>:	localuser
<linux_user_password>	websphere
<linux_root_user>	root
<linux_root_password>	
<b>DataPower information</b>	
<dp_public_ip>	
IP address of the public services on the appliance	
<dp_internal_ip>	
IP address of the DataPower appliance development and administrative functions	
<dp_WebGUI_port>	9090
Port number for the WebGUI	
<dp_its_http_port>	9990
Port for the HTTP interface to the Interoperability Test Service	

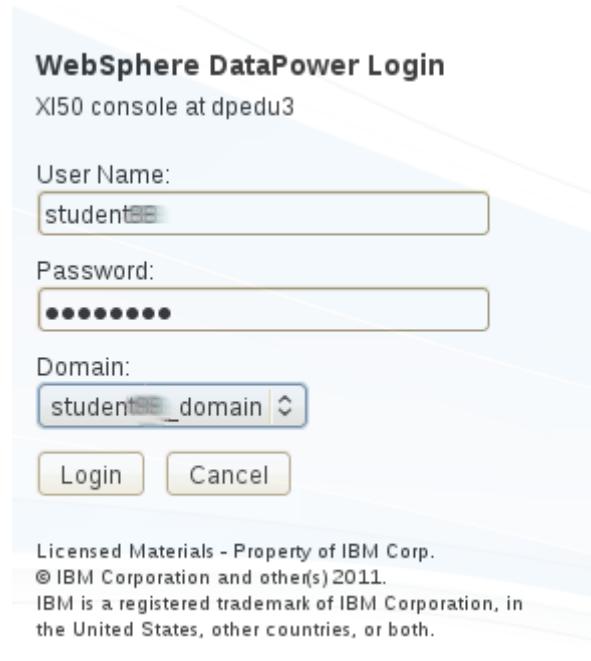
<b>Server information</b>	
<backend_server_ip>	
IP address for the services that run in WebSphere Application Server (AddressSearch web services, LDAP, registry)	
<localhost_address>	127.0.0.1
< ldap_password>	websphere
< ldap_server_port>	9080
Port number for the LDAP administrative console	
< ldap_user_name>	cn=root
<was_server_port>	9080
Port number for the services that are running in WebSphere Application Server (AddressSearch web services, LDAP, registry)	
<b>Application ports: Dev</b>	
<lab_files>	/home/localuser/dplabs/dev
Directory path of the student lab files	
<mpgw_jms_client_port>	8nn1
<mpgw_mq_client_port>	8nn0
<waf_http_port>	7nn4
<waf_https_port>	7nn5
<wsp_proxy_port>	6nn5
Port number for the AddressSearchProxy	
<mpgw_content_based_routing_port>	6nn9
Port number for the service that provides the routing function	
<mpgw_east_port>	6nn7
Port number for the service that handles East Address web services	
<mpgw_east_ssl_port>	7nn1
Port number for the service that handles East Address web services over HTTPS	
<mpgw_crypto_port>	7nn2
<mpgw_threat_protect_port>	6nn2
<xmlfw_transform_port>	6nn1
<xmlfw_validation_port>	6nn0

<mmpgw_west_port> Port number for the service that handles West Address web services	6nn8
<xmlfw_xslt_port> Port for the XSLT XML firewall	6nn6

## 1.2. Upload the crypto key and certificate files

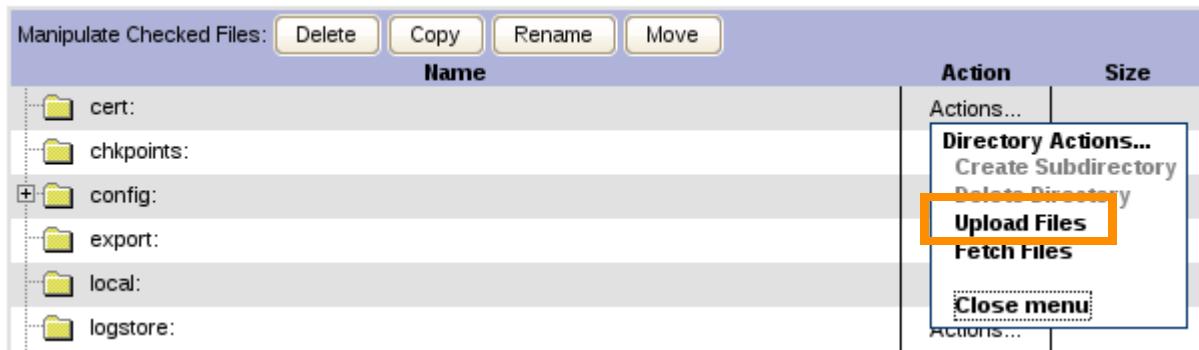
In this section, you upload a crypto private key and certificate that are used in the exercises in this course.

- \_\_\_ 1. Verify with the instructor that the user accounts and domains are already created. If not, you must wait for the creation of those objects before you can proceed.
- \_\_\_ 2. Recall your assigned user account, password, and domain.
- \_\_\_ 3. Use your assigned student user account and domain to log in to the WebGUI ([https://<dp\\_internal\\_ip>:<dp\\_WebGUI\\_port>](https://<dp_internal_ip>:<dp_WebGUI_port>)).



- \_\_\_ 4. You might be prompted to create a password. Follow the prompts, and write the new password in the previous table. After you create the password, you are directed back to the login page. Log in again with the new password.
- \_\_\_ 5. Upload the Alice private key file to the DataPower appliance.
  - \_\_\_ a. Click the **Control Panel** link at the top of the WebGUI.
  - \_\_\_ b. Click the **File Management** icon.
  - \_\_\_ c. Select the folder that represents the `cert:` directory and click the **Actions** link.

- \_\_\_ d. Click **Upload Files** to open the “Upload files” panel.



- \_\_\_ e. In the “File Management” window, click **Browse** to navigate to the Alice private key at: <lab\_files>/setup/Alice-privkey.pem
- \_\_\_ f. Click **Upload**.
- \_\_\_ g. Click **Continue** on the “Upload successful” page.
- \_\_\_ 6. Upload the Alice certificate file to the DataPower appliance.
- \_\_\_ a. Select the folder that represents the `cert:` directory and click the **Actions** link.
- \_\_\_ b. Click **Upload Files** to open the “Upload files” panel.
- \_\_\_ c. In the “File Management” window, click **Browse** to navigate to the Alice certificate at: <lab\_files>/setup/Alice-sscert.pem
- \_\_\_ d. Click **Upload**.
- \_\_\_ e. Click **Continue** on the “Upload successful” page.

### 1.3. Upload the WSDL files to the appliance

In this section, you upload two WSDL files to the `local:///` directory for use in later exercises. Continue to use the File Management part of the WebGUI.

- \_\_\_ 1. Upload the EastAddressSearch WSDL file to the `local:///` directory on the appliance.
  - \_\_\_ a. Select the folder that represents the `local:` directory and click the **Actions** link.
  - \_\_\_ b. Click **Upload Files** to open the “Upload files” panel.
  - \_\_\_ c. In the “File Management” window, click **Browse** to navigate to the East Address Search WSDL file at: `<lab_files>/setup/EastAddressSearch.wsdl`
  - \_\_\_ d. Click **Upload**.
  - \_\_\_ e. Click **Continue** on the “Upload successful” page.
- \_\_\_ 2. Upload the WestAddressSearch WSDL file to the `local:///` directory on the appliance.
  - \_\_\_ a. Select the folder that represents the `local:` directory and click the **Actions** link.
  - \_\_\_ b. Click **Upload Files** to open the “Upload files” panel.
  - \_\_\_ c. In the “File Management” window, click **Browse** to navigate to the West Address Search WSDL file at: `<lab_files>/setup/WestAddressSearch.wsdl`
  - \_\_\_ d. Click **Upload**.
  - \_\_\_ e. Click **Continue** on the “Upload successful” page.
- \_\_\_ 3. **Logout** of the WebGUI.

## 1.4. Launch Eclipse

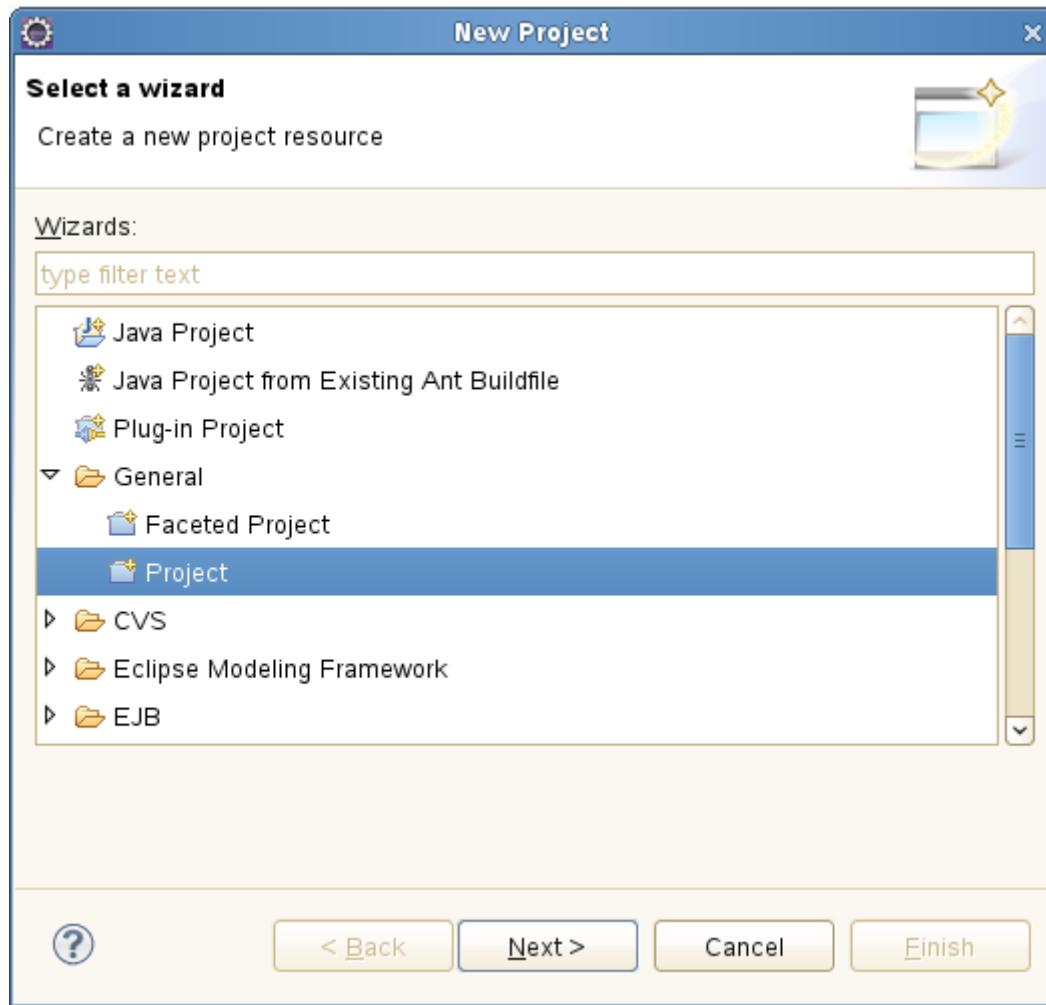
- \_\_\_ 1. Check the desktop for a **Link to Eclipse** icon, or verify that Eclipse is installed at /opt/eclipse. If it is installed, skip the following steps, and go to the next section.
- \_\_\_ 2. Download the Eclipse integrated development environment (IDE)
  - \_\_\_ a. Open a web browser and enter the URL:  
<http://www.eclipse.org/downloads/>
  - \_\_\_ b. Select the **Linux 32-bit version of Eclipse IDE for Java EE Developers**.
  - \_\_\_ c. Select any of the download sites that are listed on the page to begin the download. Save the compressed archive file on your workstation.
- \_\_\_ 3. Extract the file by using File Roller, or your preferred utility.
  - \_\_\_ a. To extract a file by using File Roller, double-click the File Roller desktop icon to launch it.
  - \_\_\_ b. Navigate to the location where you downloaded the compressed file.
  - \_\_\_ c. Double-click the file, select it, and then click **Extract**.
  - \_\_\_ d. Specify a location to extract the file – for example, /home/localuser – and click **Extract**.
  - \_\_\_ e. Optionally, create a link to Eclipse on your desktop.

## 1.5. Import the East and West Address WSDL files into Eclipse

In this section, you import two WSDL files that are used to send SOAP messages to the DataPower appliance.

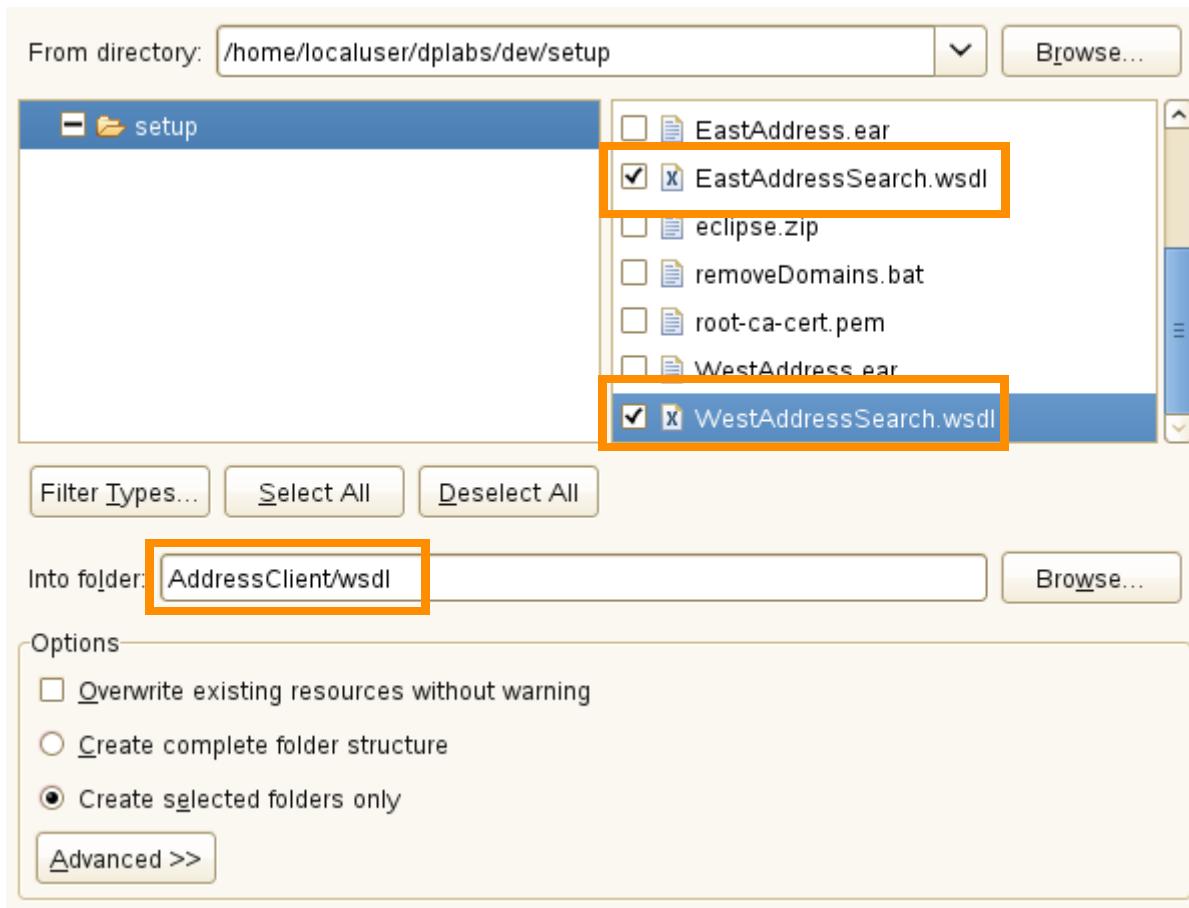
- \_\_\_ 1. Open the Eclipse EE integrated development environment (IDE).
  - \_\_\_ a. Double-click the Link to Eclipse icon on your desktop.
  - \_\_\_ b. In the workspace Launcher dialog window, enter the path:  
/home/localuser/workspace.
  - \_\_\_ c. Click **OK**. Eclipse initializes the workspace and starts up.
  - \_\_\_ d. Close the Welcome page.
- \_\_\_ 2. Switch to the Resource perspective.
  - \_\_\_ a. Select **Window > Open Perspective > Resource**.
  - \_\_\_ b. Click **OK**.
- \_\_\_ 3. Create a simple project called **AddressClient**.
  - \_\_\_ a. Select **File > New > Project**.

- \_\_\_ b. In the “New Project” dialog window, expand **General** and click **Project**.



- \_\_\_ c. Click **Next**.
- \_\_\_ d. Enter the project name: **AddressClient**
- \_\_\_ e. Click **Finish**.
- \_\_\_ 4. Import the Address WSDL files.
- \_\_\_ a. In the Project Explorer view, the **AddressClient** project is highlighted.
- \_\_\_ b. Select **File > Import**.
- \_\_\_ c. In the Import dialog window, expand **General** and select **File System**.
- \_\_\_ d. Click **Next**.
- \_\_\_ e. In the File system wizard page, click **Browse**, navigate to `<lab_files>/setup/`, and click **OK**.
- \_\_\_ f. Select both the `EastAddressSearch.wsdl` and `WestAddressSearch.wsdl` files.

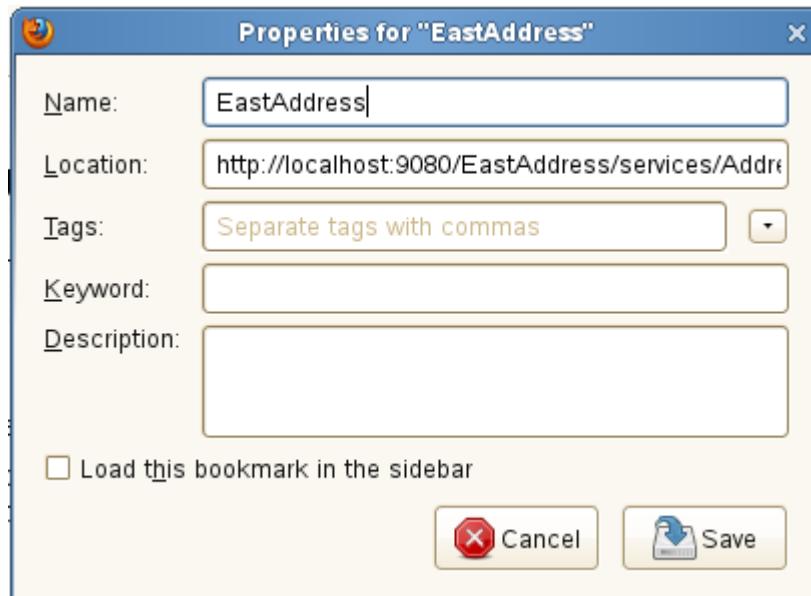
- \_\_ g. In the **Into folder** field, enter: AddressClient/wsdl



- \_\_ h. Click **Finish**.

- \_\_ 5. In the **Project Explorer** view, expand the **AddressClient** project and the **wsdl** folder. You now see both imported WSDLs.
- \_\_ 6. Close the Eclipse window.
- \_\_ 7. Update the links for **EastAddress** and **WestAddress** on the browser bookmarks toolbar.
- \_\_ a. Open Firefox.
- \_\_ b. Right-click the **EastAddress** link on the browser bookmarks toolbar.
- \_\_ c. Select **Properties** from the menu.

- \_\_\_ d. On the Properties dialog, update the Location field with the appropriate location. Your instructor can provide this information.



- \_\_\_ e. Click Save.  
\_\_\_ f. Repeat these steps to update the WestAddress link.

## 1.6. Verify cURL and Open SSL installation

cURL is used to send HTTP and HTTPS traffic to a URL. In this course, it is used to send messages to the DataPower services and ports. cURL uses OpenSSL.



### Note

Note: cURL and Open SSL are already installed. If not, they can be downloaded from the Internet, from the following URLs:

**cURL:** <http://curl.haxx.se/download.html>

**OpenSSL:** <http://www.openssl.org/source/>

- \_\_\_ 1. To verify that cURL is installed, enter the following command in a terminal window:

```
curl
```

The response looks similar to the following example:

```
curl: try 'curl --help' or 'curl --manual' for more information
```

- \_\_\_ 2. To check for installed applications, enter the following command in a terminal window:

```
rpm -qa | grep <package_name>
```

where *<package\_name>* is the name of the software package. For example:

```
rpm -qa | grep curl
```

The output looks similar to the output for *curl* (the version numbers might vary):

```
curl-7.19.0-11.22.1
```

```
libcurl4-7.19.0-11.22.1
```

The output looks similar to the output for OpenSSL (the version numbers might vary):

```
openssl-0.9.8h-30.18.1
```

```
openssl-certs-0.9.8h-25.14
```

```
libopenssl0_9_8-0.9.8h-30.18.1
```

If these applications are not installed, notify your instructor.

- \_\_\_ 3.

### End of exercise

## Exercise review and wrap-up

You determined all of the port number and variable assignments that are used in subsequent exercises. You also uploaded the Alice private key and certificate for later crypto work, and the WSDL files for later web services work. Last, you imported WSDLs into Eclipse for later testing.



# Exercise 2. Creating XSL transformations

## What this exercise is about

In this exercise, you examine an existing XML file, create an XSL style sheet, create an XML firewall service, and test the style sheet by using the new service. You also test the same style sheet by using the Interoperability Test Service.

## What you should be able to do

At the end of the exercise, you should be able to:

- Create an XSL style sheet
- Create an XML firewall service
- Transform an XML file by using the compiled XSL style sheet
- Test a style sheet by using the Interoperability Test Service
- Describe the use of DataPower variables and extensions in XSL style sheets

## Introduction

XSLT style sheets are used in many of the services that run on the DataPower appliance. In this exercise, you create and modify an XSLT style sheet and use different XSL coding to achieve a particular result. You also create an XML firewall service that you use to test the style sheet. An additional section introduces the Interoperability Test Service as another way to test a style sheet.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercise (see the Preface section in the exercise instructions for details)
- Eclipse for the XML-based coding
- Access to the `<lab_files>` directory

## Exercise instructions

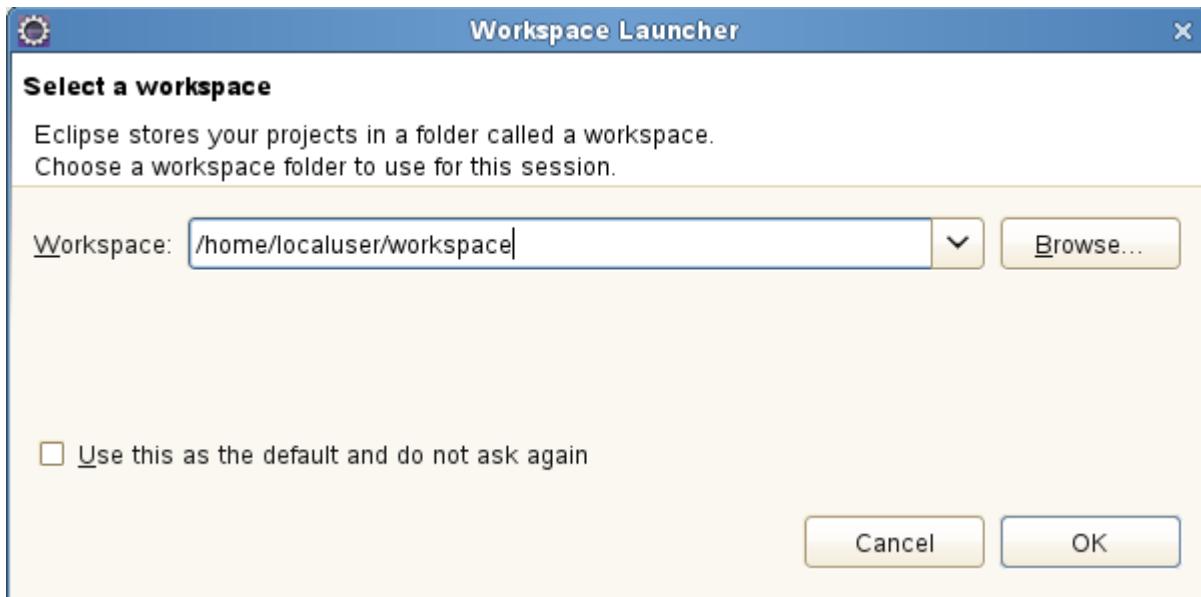
### Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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 student lab files
  - *<dp\_internal\_ip>*: IP address of the DataPower appliance development and administrative functions
  - *<dp\_WebGUI\_port>*: port number for the WebGUI
  - *<dp\_public\_ip>*: IP address of the public services on the appliance
  - *<xmfw\_xslt\_port>*: port for the XSLT XML firewall
  - *<dp\_its\_http\_port>*: port for the HTTP interface to the Interoperability Test Service

## 2.1. Examine the original XML file

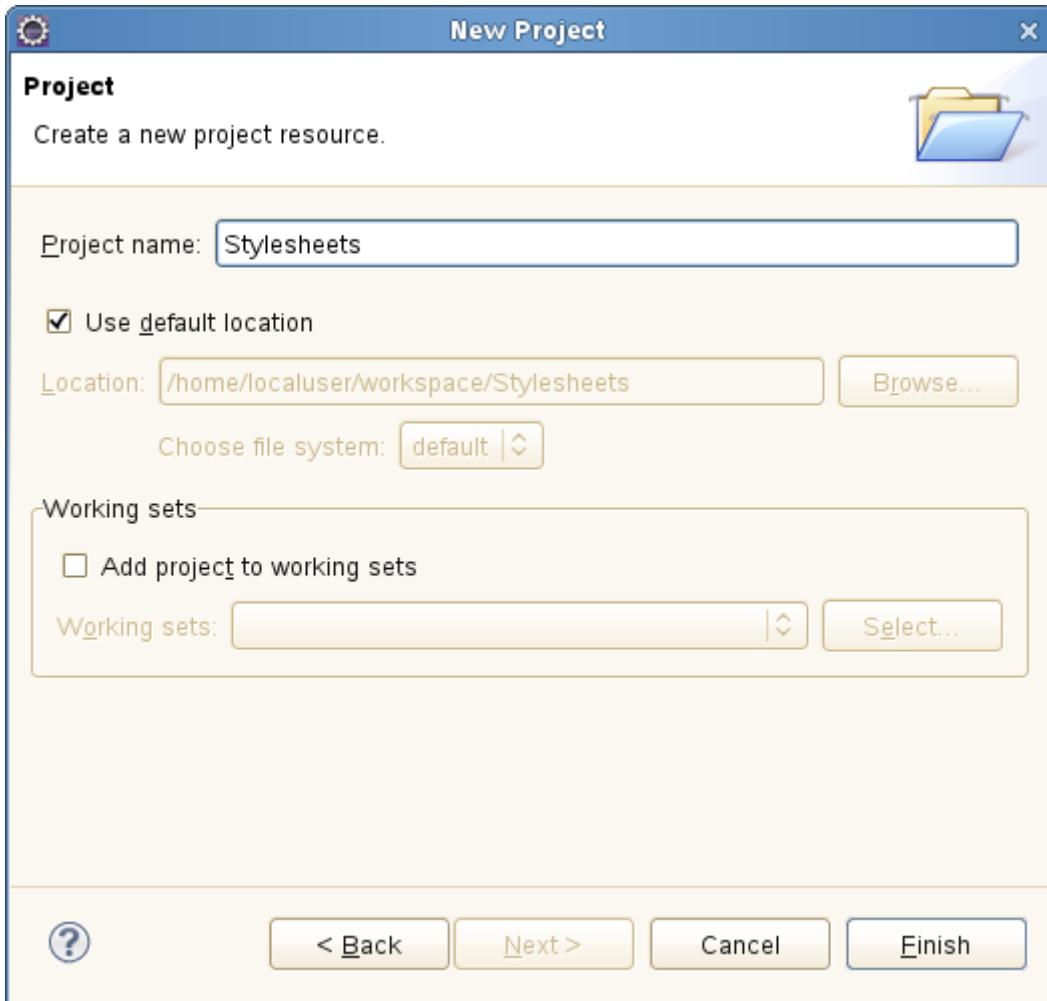
In this section, you create a project in Eclipse. You then open and examine the XML file that you transform.

- \_\_\_ 1. Double-click the **Link to Eclipse** icon on the desktop to start it.
- \_\_\_ 2. The workspace Launcher shows `/home/localuser/workspace` as the workspace directory. Click **OK**.



- \_\_\_ 3. Create a **Project** for the XSL files that is called `Stylesheets` in the Resource perspective.
  - \_\_\_ a. Close the **Welcome** tab, if it displays.
  - \_\_\_ b. Eclipse opens in the Resource perspective. If not, switch to it by using **Window > Open Perspective > Other > Resource**.
  - \_\_\_ c. Click **File > New > Project**. Expand **General** and select **Project**. Click **Next**.

- \_\_ d. Enter Stylesheets into the **Project Name** field.

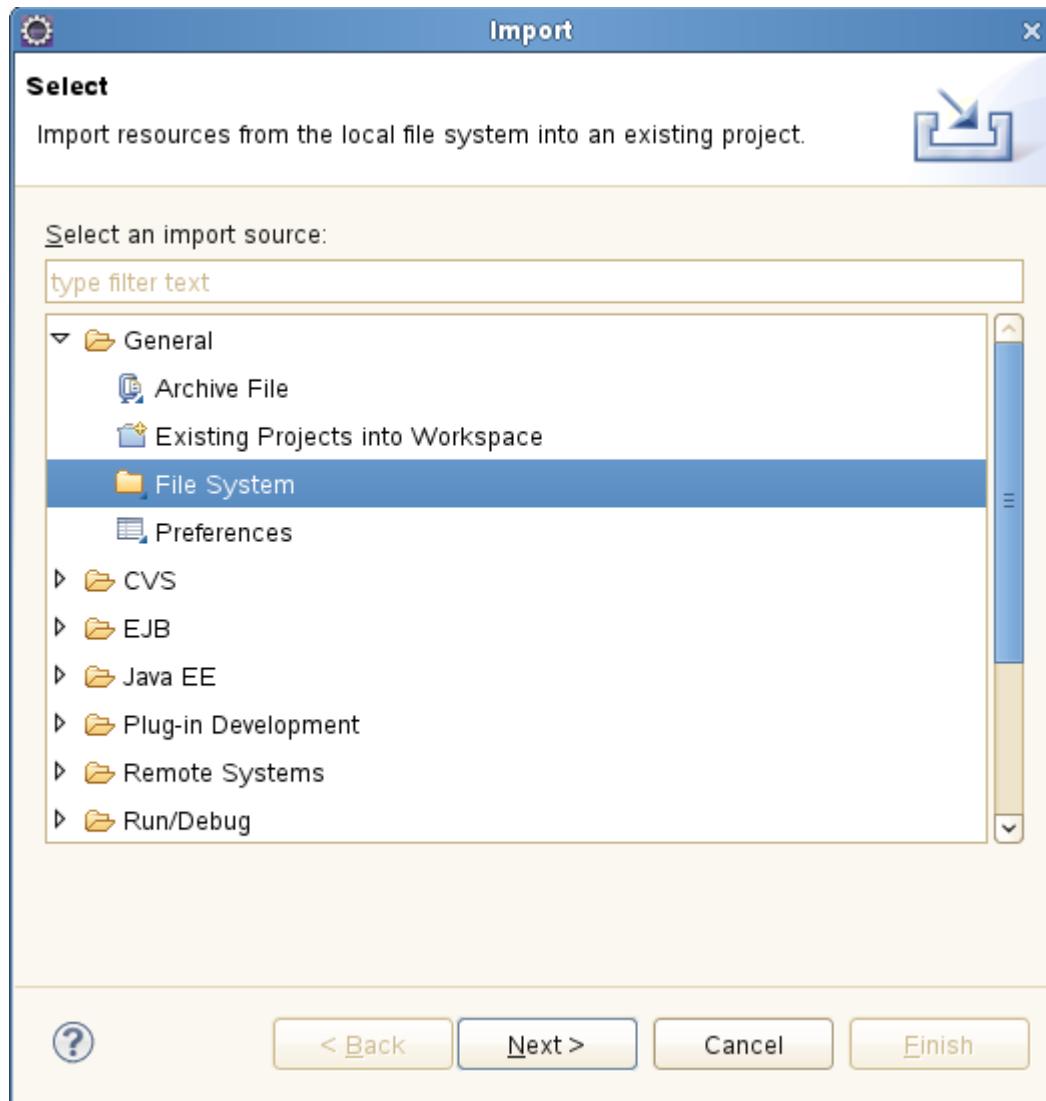


- \_\_ e. Click **Finish**. The project is created and is displayed in the Project Explorer view.

\_\_ 4. Import Token.xml.

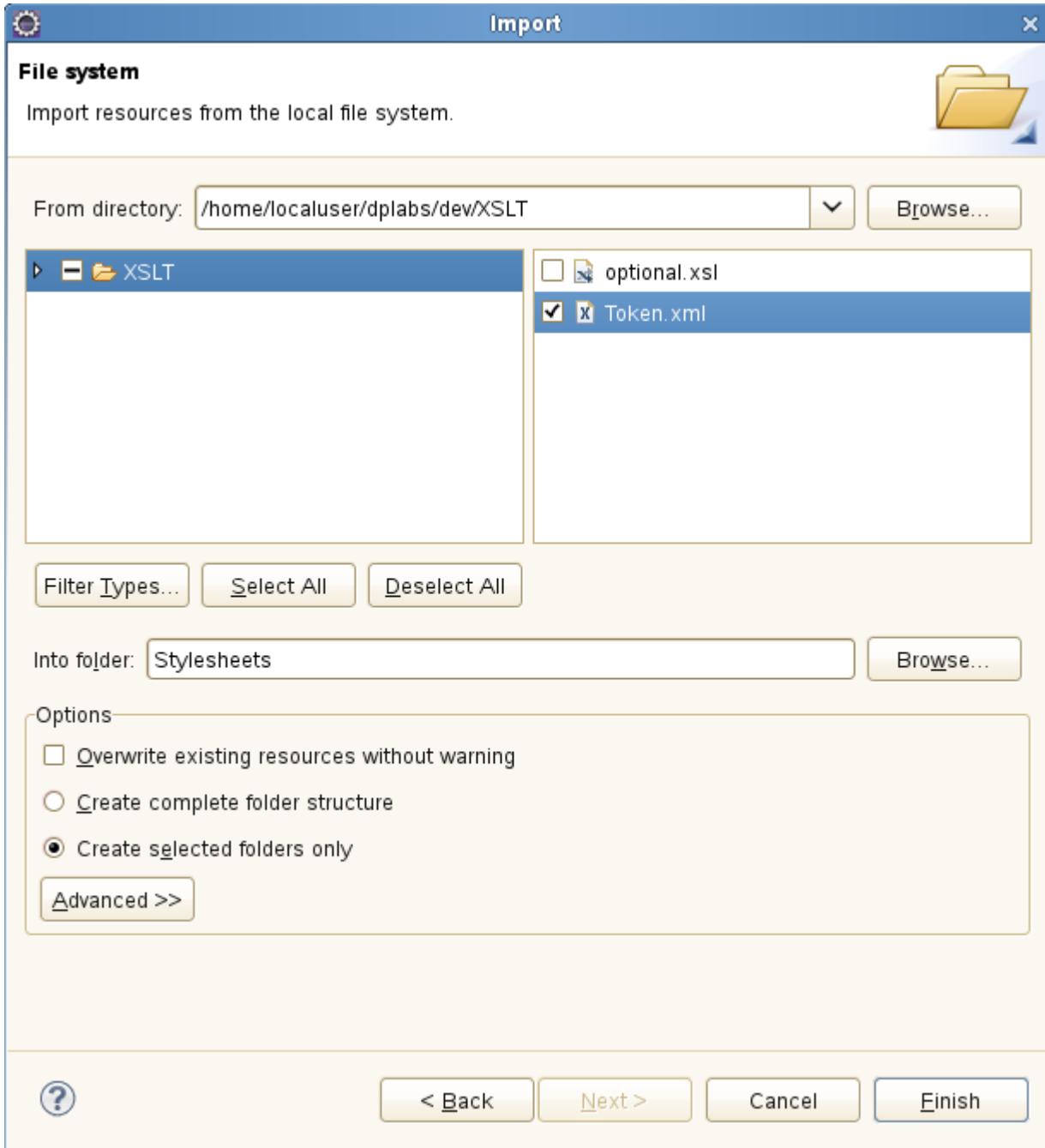
- \_\_ a. Right-click the **Stylesheets** project in the Project Explorer view and select **Import**.

- \_\_\_ b. Select General > File System.



- \_\_\_ c. Click **Next**.  
\_\_\_ d. Click **Browse** next to the From directory entry field.  
\_\_\_ e. Navigate to `<lab_files>/XSLT` and click **OK**.

- \_\_ f. Select Token.xml in the XSLT folder.



- \_\_ g. Click **Finish**.
- \_\_ h. In the Project Explorer view, expand the **Stylesheets** project.
- \_\_ i. Double-click **Token.xml** to open it. In the Editor view, select the **Source** tab.
- \_\_ 5. Examine the document structure.
- The first line defines the XML version and encoding type.  
`<?xml version="1.0" encoding="UTF-8"?>`

- The root element, `credentials`, defines and is part of the sample namespace `q0`.  
`<q0:credentials xmlns:q0="http://example.com/token">`
- The **entry** elements are sample entries. The **type** attribute would allow whatever program is parsing it to know how to deal with the body of the node.
  - LTPA stands for **Lightweight Third Party Authentication** and would notify the parser of an LDAP entry.

```
<entry type="ltpa">cn=websphere5,ou=datapower,o=IBM,c=it</entry>
<entry type="ltpa">cn=websphere6,ou=datapower,o=IBM,c=it</entry>
<entry type="ltpa">cn=websphere7,ou=datapower,o=IBM,c=it</entry>
```

- `token` represents a string, possibly stating the version of WebSphere to use.

```
<entry type="token">websphere7</entry>
```

After you are done viewing `Token.xml`, close the editor.

## 2.2. Create an XSLT style sheet to print entries in a DN element

In this section, you create an XSLT style sheet to change the names of the elements. The root element, `credentials`, is changed to `DNLlist` and the `entry` elements are changed to `DN`.

- \_\_\_ 1. Create an XSL document in Eclipse.
  - \_\_\_ a. Right-click the **Stylesheets** project in the Project Explorer view and select **New > Other**.
  - \_\_\_ b. Scroll down and select **XML > XML File**.
  - \_\_\_ c. Click **Next**.
  - \_\_\_ d. Leave the `Stylesheets` folder selected and enter `domain.xsl` as the file name.
  - \_\_\_ e. Click **Finish**. The document opens in the editor. The version and encoding are already created.
- \_\_\_ 2. Enter the style sheet tag and output format tags.
  - \_\_\_ a. Enter the **stylesheet** element. Define both the `xsl` and `http://example.com/token` namespaces in this element.

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:q0="http://example.com/token">
```
  - \_\_\_ b. Leave the **stylesheet** element open, and on the next line, define the output format of the modified document with an **output** element.

```
<xsl:output method="xml" />
```

As you begin typing this tag, Eclipse might automatically complete it.
- \_\_\_ 3. Add the logic to convert the names of the **entry** elements to **DN** and the root element to **DNLlist**.

In this step, you create the logic to change the root element name to **DNLlist** and the **entry** element to **DN**, while keeping all of the body text the same. Create a template that matches the **entry** element and creates an element **DN** with the body text of **entry**. Another template must also be created that matches the root and creates an element **DNLlist**. Within the **DNLlist** element, apply the **entry** template so that the children are transformed.

- \_\_\_ a. Create a template that matches **entry**. Ensure that it is fully qualified.

```
<xsl:template match="q0:credentials/entry">
```
- \_\_\_ b. Create an element **DN** by using the `element` tag. Put it in the `http://example.com/token` namespace with the `namespace` attribute.

```
<xsl:element name="DN" namespace="http://example.com/token">
```

- \_\_\_ c. Next, populate the **DN** element with the body text of **entry**. Use the **value-of** tag and the **select** attribute with the regular expression for the value of the current element.

```
<xsl:value-of select=". "/>
```

- \_\_\_ d. This template is complete. Add closing **element** and **template** tags.

```
</xsl:element>
</xsl:template>
```

- \_\_\_ e. Create another template that matches **credentials**. In this case, use the regular expression for the root of the document.

```
<xsl:template match="/">
```

- \_\_\_ f. Create an element that is named **DNList** in the namespace <http://example.com/token>.

```
<xsl:element name="DNList" namespace="http://example.com/token">
```

- \_\_\_ g. Populate **DNList** with the **DN** elements that the previous template created by using the **apply-templates** tag.

```
<xsl:apply-templates select="q0:credentials/entry"/>
```

- \_\_\_ h. The root template is complete. Close the **element** and **template** tags.

```
</xsl:element>
</xsl:template>
```

- \_\_\_ i. Close the **stylesheet** tag.

```
</xsl:stylesheet>
```

- \_\_\_ j. If you want to have Eclipse clean up the formatting of the source code with the `domain.xsl` window active, right-click and select **Source > Format**. The final formatted source resembles the code that is shown in this example:

```
domain.xsl ✘
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:q0="http://example.com/token">
  <xsl:output method="xml" />
  <xsl:template match="q0:credentials/entry">
    <xsl:element name="DN" namespace="http://example.com/token">
      <xsl:value-of select=". "/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="/">
    <xsl:element name="DNList" namespace="http://example.com/token">
      <xsl:apply-templates select="q0:credentials/entry" />
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

- \_\_\_ k. Save and close `domain.xsl`. Leave Eclipse open.

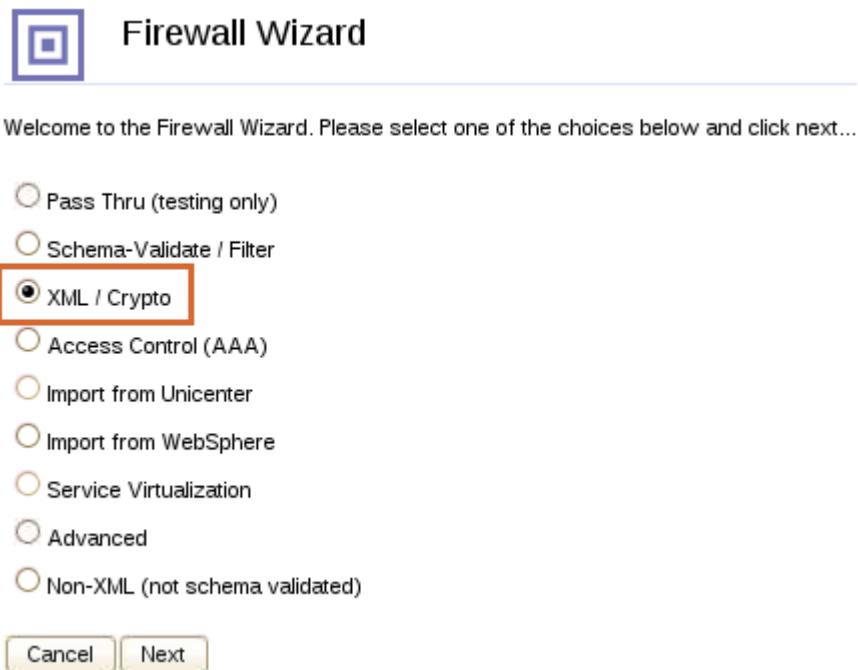
\_\_\_ 4. Create an XML firewall to test domain.xsl.

In this step, you create a loopback XML firewall with a **Transform** action that references domain.xsl.

- \_\_\_ a. Open a web browser to the DataPower WebGUI:

`https://<dp_internal_ip>:<dp_WebGUI_port>/`

- \_\_\_ b. In the login page, enter the user name, password, and custom domain that were assigned in the exercise setup. Click **Login**.
- \_\_\_ c. The DataPower WebGUI Control Panel displays. Click **XML Firewall** from the Control Panel.
- \_\_\_ d. Click **Add wizard**.
- \_\_\_ e. Select **XML/Crypto** and click **Next**.



- \_\_\_ f. Enter the **Firewall Name** XSLTFirewall and click **Next**.
- \_\_\_ g. Set the **Firewall Type** to **loopback-proxy** and click **Next**.



**Information**

In the Add wizard, the Firewall Type is listed as **loopback-proxy**. On most other pages in the WebGUI, it is listed as **Loopback**.

- \_\_\_ h. Click **Select Alias**.
- \_\_\_ i. If **dp\_public\_ip** is displayed in the Select Host Alias dialog, select it and click **Apply**. The dialog closes, and **dp\_public\_ip** shows in the Device Address field.
- \_\_\_ j. If it does not show in the list, click **Cancel**, and enter **<dp\_public\_ip>** for the **Device Address**.

**Information**

A “host alias” can be created to provide a variable-type reference for the appliance IP address. The developer can specify the variable, rather than a specific address, so the service configuration is more portable. The appliance administrator defines the host alias name and the IP address it refers to.

The screen captures in this exercise use a host alias.

- \_\_\_ k. Enter **<xmlfw\_xslt\_port>** for the **Device Port**, leave the SSL default, and click **Next**.

**Device Address**  
dp\_public\_ip

**Device Port**  
6976 \*

**Do you want to use SSL?**  
 on  off

\*

- \_\_\_ l. Leave verifying and decrypting requests **off**. Click **Next**.
- \_\_\_ m. Leave the signing and encrypting requests **off** and click **Next**.
- \_\_\_ n. Look over the settings and click **Commit**.

- \_\_ o. The operational state of all components is **up**. Click **Done**.

### Create a XML/Crypto XML Firewall Service

You have successfully created XML Firewall XSLTFirewall.

*Operational state of all created and referenced objects.*

Type	Name	Operational State
XML Firewall Service	XSLTFirewall	[up]
└ XML Manager	default	[up]
└ User Agent	default	[up]
└ Processing Policy	XSLTFirewall	[up]
└ Matching Rule	XSLTFirewall	[up]
└ Processing Rule	XSLTFirewall_request	[up]
└ Processing Action	XSLTFirewall_request_get_plaintext	[up]
└ Processing Action	XSLTFirewall_request_results	[up]

[View Object Status](#)

[View Policy](#)

[Done](#)

- \_\_ 5. Change the wizard configuration of the **XSLTFirewall**.

- \_\_ a. Click **XML Firewall** on the Control Panel.  
 \_\_ b. Click **XSLTFirewall** on the “Configure XML Firewall” catalog page.

### Configure XML Firewall

XML Firewall Name	Op-State	Logs	Req-Type	Local Address	Port	Resp-Type	Remote Address	Port
XSLTFirewall	up		soap	dp_public_ip	6976	soap		

[Add Wizard](#)

[Add Advanced](#)

- \_\_ c. Scroll down and change the **Request Type** to **XML**.  
 \_\_ d. Modify the automatically generated **Processing Policy** by clicking **Edit [...]**.



- \_\_\_ e. Hover your mouse pointer over the **Match** action. It matches any incoming URL directed to the port. The URL is set to the wildcard, which is an asterisk (\*).

## Configure XML Firewall Style Policy

**Policy:**

Policy Name: XSLTFirewall \*

Apply Policy Cancel Export | View Log | View Status |

**Rule:**

Rule Name: XSLTFirewall\_request Rule Direction: Client to Server

New Rule Delete Rule

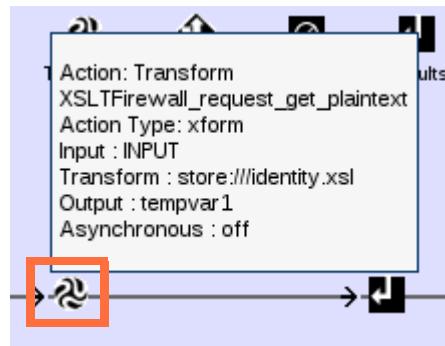
Create rule: Click New, drag action icons onto line. Edit rule: Click on rule, double-click on action

Action: Match  
XSLTFirewall  
Match Rule:  
Type : url  
Url : \*  
Method : default  
MatchWithPCRE : off  
CombineWithOr : off

Filter Sign Verify Encrypt Transform Route AAA Results Advanced

CLIENT

- \_\_\_ f. The policy automatically generates a **Transform** action. Double-click the **Transform** action to modify it.



- \_\_\_ g. In the **XSL style sheet** fields, use the drop-down list to select the local:/// directory.  
 \_\_\_ h. Click **Upload**.

- \_\_\_ i. Click **Browse** and navigate to: /home/localuser/workspace/Stylesheets. Select domain.xsl and click **Open**.
  - \_\_\_ j. Click **Upload**; then click **Continue**.
  - \_\_\_ k. Click **Done** back on the “Configure Transform Action” page.
  - \_\_\_ l. Click **Apply Policy**, and then close the policy editor window.
  - \_\_\_ m. **Apply** the XML firewall.
- \_\_\_ 6. Test domain.xsl by sending token.xml to **XSLTFirewall** with a cURL command.
- \_\_\_ a. Open a Terminal window.
  - \_\_\_ b. Navigate to: <lab\_files>/XSLT
  - \_\_\_ c. Issue the following cURL command:  

```
curl --data-binary @Token.xml  
http://<dp_public_ip>:<xmfw_xsln_port>/
```
  - \_\_\_ d. You receive the following response that contains the new XML:  

```
<?xml version="1.0" encoding="UTF-8"?>  
<DNList xmlns="http://example.com/token">  
    <DN>cn=websphere5,ou=datapower,o=IBM,c=it</DN>  
    <DN>cn=websphere6,ou=datapower,o=IBM,c=it</DN>  
    <DN>cn=websphere7,ou=datapower,o=IBM,c=it</DN>  
    <DN>websphere7</DN>  
</DNList>
```



#### Note

The output that you see in your Terminal window might not be formatted like the example shown.



#### Information

It is a good idea to keep the Terminal window open. You can recall previous cURL commands, rather than retyping the same or similar commands in later steps. Be sure that you are in the correct directory for each test.

## 2.3. Modify domain.xsl to selectively print based on attributes

In this section, you modify `domain.xsl` by using the `xsl:if` tag to convert **entry** elements to **DN** elements only if their type is LTPA. Then, you overwrite the version of the file on the DataPower appliance with the new one and send a request to the firewall by using cURL.

- \_\_\_ 1. Open `domain.xsl` in Eclipse and add the logic to print only entries of type LTPA.
  - \_\_\_ a. Switch to Eclipse and double-click `domain.xsl` to open it.
  - \_\_\_ b. In the template that matches the `q0:credentials/entry`, use an `xsl:if` tag to insert an expression to create the **DN** element only if the **entry** is of type LTPA. This condition is represented in the **test** attribute of `xsl:if`, and it uses a regular expression.

```
<xsl:template match="q0:credentials/entry">
  <xsl:if test="@type = 'ltpa'">
    <xsl:element name="DN" namespace="http://example.com/token">
      <xsl:value-of select=". />
    </xsl:element>
  </xsl:if>
</xsl:template>
```

- \_\_\_ c. Save and close `domain.xsl`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:q0="http://example.com/token">
  <xsl:output method="xml" />
  <xsl:template match="q0:credentials/entry">
    <xsl:if test="@type = 'ltpa'">
      <xsl:element name="DN" namespace="http://example.com/token">
        <xsl:value-of select=". />
      </xsl:element>
    </xsl:if>
  </xsl:template>
  <xsl:template match="/">
    <xsl:element name="DNList" namespace="http://example.com/token">
      <xsl:apply-templates select="q0:credentials/entry" />
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

- \_\_\_ 2. Upload the new `domain.xsl` on the XML firewall.
  - \_\_\_ a. Switch back to the DataPower WebGUI.
  - \_\_\_ b. Navigate to the configuration page for the XSLTFirewall.
  - \_\_\_ c. Open the Processing Policy editor and double-click the **Transform** action.
  - \_\_\_ d. Click **Upload** next to the **Processing Control File** field.
  - \_\_\_ e. Browse to `/home/localuser/workspace/Stylesheets` and select `domain.xsl`.

- \_\_\_ f. Check **Overwrite Existing File**.
- \_\_\_ g. Click **Upload**; then click **Continue**.
- \_\_\_ h. Click **Done** on the **Transform** action editor; then click **Apply Policy**, and close the policy editor.
- \_\_\_ i. Click **Apply** on the XML firewall configuration page.



### Information

Another approach to directly updating the `domain.xsl` file on the appliance is to use **File Management** from the Control Panel.

- \_\_\_ 3. Test the XSLTFirewall with cURL.

- \_\_\_ a. Open a Terminal window (or use one that is already open) and navigate to:  
`<lab_files>/XSLT`
- \_\_\_ b. Reissue the following command:  

```
curl --data-binary @Token.xml  
http://<dp_public_ip>:<xmfw_xsln_port>/
```
- \_\_\_ c. The response is the same as the previous section, except that now the `<entry type="token">websphere7</entry>` is excluded in the transformed document:  

```
<?xml version="1.0" encoding="UTF-8"?>  
<DNList xmlns="http://example.com/token">  
    <DN>cn=websphere5,ou=datapower,o=IBM,c=it</DN>  
    <DN>cn=websphere6,ou=datapower,o=IBM,c=it</DN>  
    <DN>cn=websphere7,ou=datapower,o=IBM,c=it</DN>  
</DNList>
```

## 2.4. Use a substring function to print only the CN

In this section, you modify the **xsl:value-of** tag within the **DN** element to contain only the **cn** value. For example,

```
<entry type="ltpa">cn=websphere5,ou=datapower,o=IBM,c=it</entry>
```

becomes:

```
<DN>websphere5<DN>
```

- \_\_\_ 1. Open `domain.xsl` and modify the **xsl:value-of** tag.
  - \_\_\_ a. Double-click `domain.xsl` in the Project Explorer view.
  - \_\_\_ b. Modify the select attribute of **xsl:value-of** within the **DN** element to display only the **cn** value.

Use the **substring-after** function with the **substring-before** function. They take two parameters: the string to shorten, and the string that identifies where the substring is to start or end.

- Obtain a string from the `<entry>` value before the `ou` value. Be sure to include the comma before `ou` in the identifier to indicate where to end the substring. Recall that the **xsl:value-of** tag is within the context of the `<entry>` element, so the regular expression for the value of the current node can be used to obtain the initial string.

```
substring-before( . , ' ,ou=' )
```

- Use this substring within a substring-after function, taking the text beginning after `cn=`

```
substring-after (substring-before( . , ' ,ou=' ), 'cn=' )
```

- The modified XSL file displays as follows:

```
...
<xsl:element name="DN" namespace="http://example.com/token">
  <xsl:value-of select="substring-after (substring-before( . ,
    ' ,ou=' ) , 'cn=' )"/>
</xsl:element>
...
```

- \_\_\_ c. Save and close `domain.xsl`.
- \_\_\_ 2. Upload the new `domain.xsl` to the firewall.
  - \_\_\_ a. Repeat step 2 of section 3, or try the File Management approach.
- \_\_\_ 3. Test the XSLTFirewall with cURL.
  - \_\_\_ a. Open a Terminal window (or use one that is already open) and navigate to:  
`<lab_files>/XSLT`

- \_\_\_ b. Reissue the following command:

```
curl --data-binary @Token.xml  
http://<dp_public_ip>:<xmlfw_xsslt_port>/
```

- \_\_\_ c. The response is the same as the previous section, except now the only text in the `DN` elements is the `cn` value:

```
<?xml version="1.0" encoding="UTF-8"?>  
<DNLList xmlns="http://example.com/token">  
    <DN>websphere5</DN>  
    <DN>websphere6</DN>  
    <DN>websphere7</DN>  
</DNLList>
```

## 2.5. Test a style sheet by using the Interoperability Test Service

The Interoperability Test Service (ITS) can receive a style sheet and message data, and return the transformation results. It is not necessary for a developer to create a service, such as XSLTFirewall, just to test a style sheet.

In this section, you invoke ITS to test `domain.xsl`.

- 1. Open a Terminal window (or use one that is already open) and navigate to:  
`<lab_files>/XSLT`

- 2. Issue the following command:

```
./dp-interop-client.sh -x  
/home/localuser/workspace/Stylesheet/domain.xsl -i Token.xml -h  
<dp_internal_ip> -p <dp_its_http_port>
```

`dp-interop-client.sh` is the Secure Shell script to invoke ITS that is supplied on the resource kit.

The `domain.xsl` reference is to the edited version in the Eclipse directory tree.

- 3. The response is the same as the previous section:

```
<?xml version="1.0" encoding="UTF-8"?>  
<DNList xmlns="http://example.com/token">  
    <DN>websphere5</DN>  
    <DN>websphere6</DN>  
    <DN>websphere7</DN>  
</DNList>
```

ITS works nicely for testing of style sheets and static data. However, if previous actions create or modify the input to the transform, ITS has no way to duplicate that behavior.

## 2.6. (Optional) Modify the templates to ignore namespace

In this section, you modify the template **match** attributes to ignore namespaces and to use local names. This section is optional because no new XSL processing occurs.

In this method, the style sheet format is the same, but the way the elements are specified in the template tags is different. Previously, elements were identified by using the namespace. Here, they are identified by using regular expressions that are mapped directly from the root.

- \_\_\_ 1. Modify `domain.xsl` in Eclipse.
  - \_\_\_ a. Switch back to Eclipse.
  - \_\_\_ b. Double-click `domain.xsl` to open it.
  - \_\_\_ c. Make a regular expression that points to the entry from within the root context by using local names to replace the instances of `q0:credentials/entry`. This line is as follows:

`*[local-name() = 'credentials' ]/*[local-name() = 'entry' ]`

- \_\_\_ d. Replace both the values of the `select` attribute of `xsl:apply-templates` and the `match` attribute of the `entry` `xsl:template`:

...

`<xsl:template`

`match="*[local-name() = 'credentials' ]/*[local-name() = 'entry' ]">`  
 ...  
`<xsl:apply-templates`  
`select="*[local-name() = 'credentials' ]/*[local-name() = 'entry' ]" />`  
 ...

- \_\_\_ e. Save and close `domain.xsl`.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:q0="http://example.com/token">
  <xsl:output method="xml" />
  <xsl:template match="*[local-name() = 'credentials' ]/*[local-name() = 'entry' ]">
    <xsl:if test="@type = 'ltpa'">
      <xsl:element name="DN" namespace="http://example.com/token">
        <xsl:value-of select="substring-after(substring-before(. , ',ou='),'">
      </xsl:element>
    </xsl:if>
  </xsl:template>
  <xsl:template match="/">
    <xsl:element name="DNList" namespace="http://example.com/token">
      <xsl:apply-templates select="*[local-name() = 'credentials' ]/*[local-name() = 'ent'">
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>

```

- \_\_\_ 2. Upload the new `domain.xsl` to the firewall.

- \_\_\_ a. Repeat step 2 of section 3.

- 
3. Test the XSLTFirewall with cURL.
- \_\_ a. Open a Terminal window (or use one that is already open) and navigate to <lab\_files>/XSLT.
  - \_\_ b. Reissue the following command:  
curl --data-binary @Token.xml  
http://<dp\_public\_ip>:<xmlfw\_xsln\_port>/
  - \_\_ c. The response is the same as the previous section.  

```
<?xml version="1.0" encoding="UTF-8"?>
<DNList xmlns="http://example.com/token">
    <DN>websphere5</DN>
    <DN>websphere6</DN>
    <DN>websphere7</DN>
</DNList>
```
  - \_\_ d. Back in your web browser, click **Save Config** to write your firewall definition to the flash memory in the appliance.

### End of exercise

## Exercise review and wrap-up

In this exercise, you created an XSL style sheet in several steps, testing along the way. You used several XSL tags and different ways of identifying nodes in the document structure.

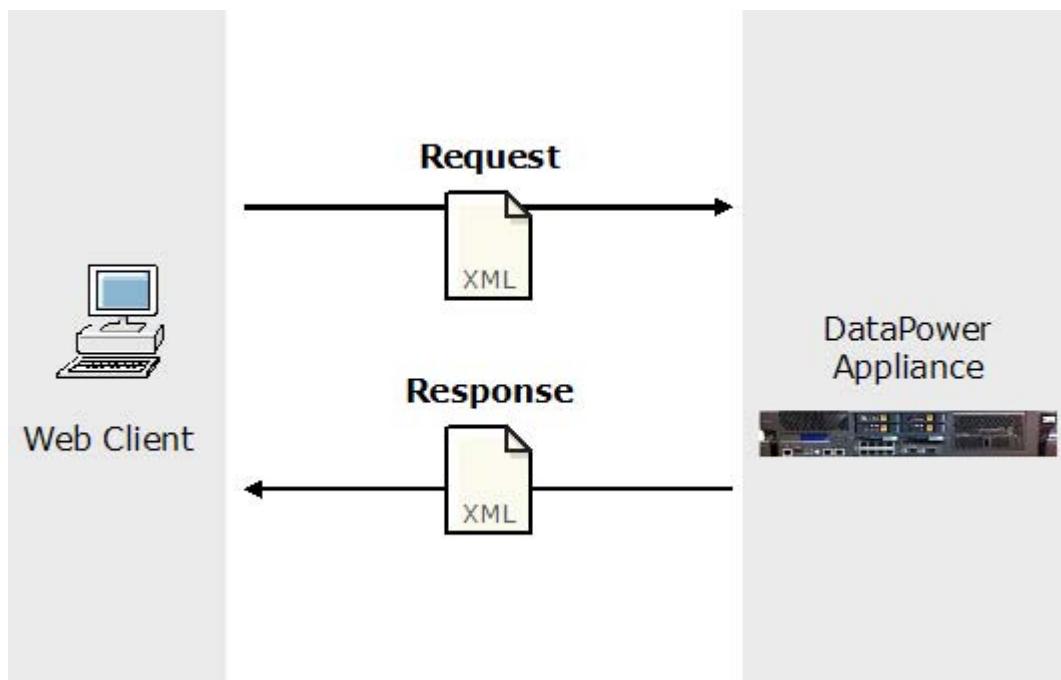
You also used the Interoperability Test Service as an alternative way to test a style sheet.

# Exercise 3. Creating a simple XML firewall

## What this exercise is about

This exercise shows you how to create a basic XML firewall that can perform schema validation and message transformation. You learn the basic steps necessary to implement a message flow within the DataPower appliance. You implement the validation and transformation by configuring an XML firewall in the loopback proxy mode. You then test the scenarios with the cURL command-line tool.

The XML firewall provides a wizard. The wizard allows for a simpler process for creating the DataPower service. Therefore, to start learning foundational concepts of DataPower services, the XML firewall is an excellent teaching tool. However, the XML firewall is more limited than the multi-protocol gateway, and less scalable. Use the XML firewall in proof-of-concepts, demonstrations, and as a teaching aid only. Use the other services for production applications.



## What you should be able to do

At the end of the exercise, you should be able to:

- Create an XML firewall

- Create a document processing policy with message schema validation and transformation
- Test the message flow by using the command-line tool cURL

## Introduction

In this exercise, you implement the scenarios that are presented during the lecture.

The exercise is organized into three sections:

1. The first section of this exercise demonstrates how to configure an XML firewall to do schema validation of incoming XML messages against an XML schema.
2. The second section of the exercise demonstrates how to configure an XML firewall to do a simple XSL transformation.
3. The third section of the exercise integrates the schema validation and transformation capabilities to demonstrate an integrated scenario that involves both schema validation and XSL transformation.

## Required materials

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercise (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- Access to the `<lab_files>` directory

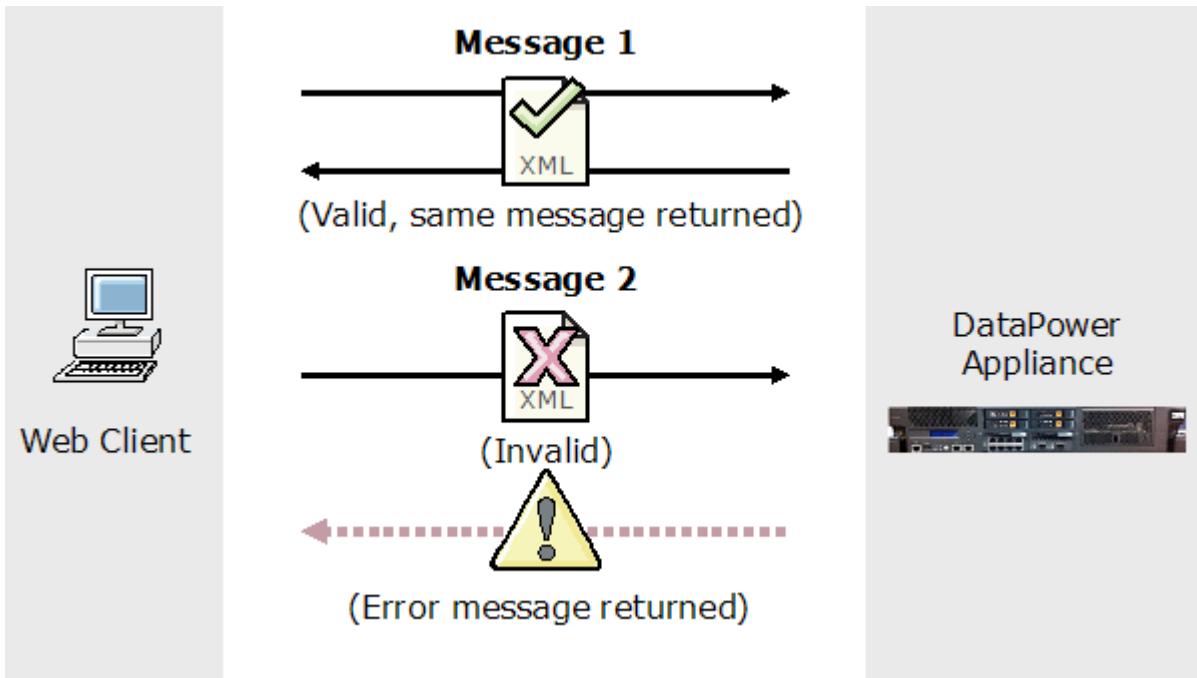
# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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
  - *<dp\_internal\_ip>*: IP address of the DataPower appliance development and administrative functions
  - *<dp\_public\_ip>*: IP address of the public services on the appliance
  - *<dp\_WebGUI\_port>*: the port that is used to access the DataPower WebGUI
  - *<xmlfw\_validation\_port>*: XML firewall port for validation (and integration)
  - *<xmlfw\_transform\_port>*: XML firewall port for transformation

## Create a basic XML firewall to do message validation

You create a basic XML firewall as a loopback proxy and configure a document processing policy that contains a schema validation action. Incoming XML requests are matched against a *match rule* that determines whether the incoming traffic is applicable to a document processing rule. The processing rule specifies the steps that are applied to incoming requests and it consists of one or more actions.



You create the XML firewall service and use the following three steps to test it:

- Validation: XML firewall creation
- Validation: XML firewall examination and configuration
- Validation: XML firewall testing

### 3.1. Validation: XML firewall creation

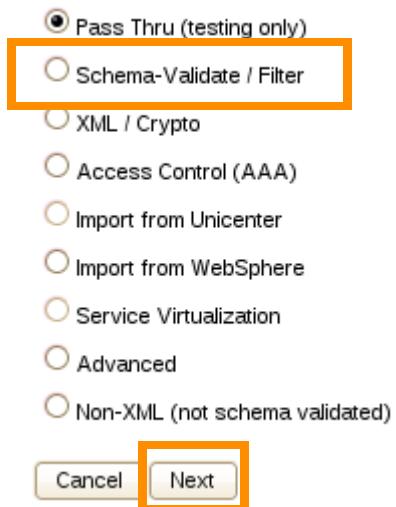
- \_\_\_ 1. Use a web browser to log on to the DataPower WebGUI.
  - \_\_\_ a. Use your assigned student user account and domain to log in to the WebGUI ([https://<dp\\_internal\\_ip>:<dp\\_WebGUI\\_port>](https://<dp_internal_ip>:<dp_WebGUI_port>)). Be sure that you are **not** in the default domain.
  - \_\_\_ b. When you are logged in, take some time to familiarize yourself with the organization of the administrative console. In the vertical navigation bar, you see three categories:
    - Control Panel quick link
    - DataPower appliance menus
    - DataPower firmware level



The five menu options are:

- **STATUS:** Contains logs and system utilization information.
  - **SERVICES:** Services such as XML firewall, HTTP service, web services Proxy service, and other services you can create with the DataPower appliance.
  - **NETWORK:** Network interfaces, settings, and network-level services.
  - **ADMINISTRATION:** System setup options, such as file management, import, and export configuration.
  - **OBJECTS:** The components that the DataPower services use. This menu is used for advanced configuration of the appliance.
- \_\_\_ 2. Review the files `AddressMsg.xsd` and `AddressReq.xml` in the `<lab_files>/simpleFW` directory. The XML file is validated against the XSD file in this service. You can use Eclipse (**File > Open File**) to edit these files, and it uses an XSD and an XML editor to display the contents.

- \_\_\_ 3. From the Control Panel, click **XML Firewall**.
- \_\_\_ 4. On the Configure XML Firewall catalog page, you see a list of XML firewalls that are currently defined in your domain. To use a wizard to start the creation of the XML firewall, click **Add wizard**.
  - \_\_\_ a. The Firewall wizard page lists various types of XML firewalls that this wizard can create. Select **Schema-Validate / Filter**, and click **Next**.



- \_\_\_ b. On the next page, you are asked to supply the name of the new service. Enter a name of: MyBasicFirewall

A screenshot of a Windows-style dialog box titled 'Firewall Name'. It has a text input field containing 'MyBasicFirewall' with a red asterisk (\*) to its right. Below it is a section titled 'Attempt to use streaming mode.' with a radio button group: 'on' (unchecked) and 'off' (checked). At the bottom are 'Cancel' and 'Next' buttons.

- \_\_\_ c. Click **Next** to continue.
- \_\_\_ d. The following page in the wizard asks you to specify the type of firewall. Since all you want to do is send the XML message to the service, have it validated, and get the message back, you want to have the service act as a loopback. Select **loopback-proxy** from the list, and click **Next**.

A screenshot of a Windows-style dialog box titled 'Firewall Type'. It shows a dropdown menu with 'loopback-proxy' selected and a red asterisk (\*) to its right. Below the dropdown are 'Back', 'Cancel', and 'Next' buttons.

- \_\_\_ e. The next page has you configure the Front End (Client) Information, or services interface to the client. The **Device Address** specifies the IP addresses on the appliance to which this service responds. The **Device Port** indicates the specific port on the specified device address on which this service listens. Enter a port number of: <xmlfw\_validation\_port>  
Leave the **SSL** option at its default of **off**. Click **Next**.

**Device Address**  
 Select Alias \*

**Device Port** \*

**Do you want to use SSL?**  
 on  off

- \_\_\_ f. Although you selected the **Schema-Validate / Filter** option for this wizard, you are not going to use a filter. The next page asks if you want to use a filter, so leave it at its default of **off**. Click **Next**.
- \_\_\_ g. Since you specified a validating wizard, it now wants to know how to validate. Specify a **schema Validation Method of URL**.



### Note

The next five steps show the generic way that files are uploaded from the workstation to the appliance.

- \_\_\_ h. When you click the radio button, the page reloads, and includes entry fields for schema information. Since the schema file exists on the workstation, upload it to the appliance. Click **Upload**.

#### Validation Information

Select the validation method.

##### Schema Validation Method

- Validate Document via Attribute Rewrite Rule
- Validate Document via Schema Attribute
- Validate Document via Schema URL
- Validate Document via WSDL URL

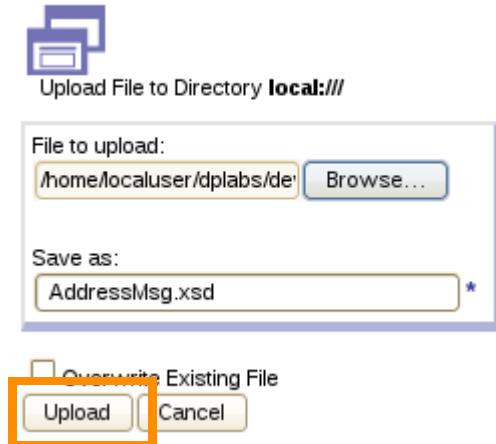
##### Schema URL

local:///

(none)  Fetch... Edit... View... \*

Back Cancel Next

- \_\_ i. On the File Management dialog, select **File** and click **Browse**.
- \_\_ j. From the file selection dialog, select <lab\_files>/simpleFW/AddressMsg.xsd, which is the intended schema.
- \_\_ k. The File Management dialog window updates the file name in the **File to upload** field. Click **Upload**.



An upload confirmation opens.

- \_\_ l. Click **Continue**. The schema Validation Method page now has the schema file information completed.



#### Note

The uploaded file is in the `local:` directory on the appliance file system. The `local:` directory is local to the domain.

- \_\_ m. Click **Next**. You get a “Confirm Your Changes and Commit” page.

\_\_\_ n. Click **Commit**.

**Confirm Your Changes and Commit**

Firewall Name:	MyBasicFirewall
Firewall Type:	loopback-proxy
Server Address:	
Server Port:	
SSL Server Crypto Profile:	
Device Address:	dp_public_ip
Device Port:	6950
SSL Client Crypto Profile:	
Do you want to use a filter?:	off
Schema Validation Method:	Validate Document via Schema URL
Schema URL:	local:///AddressMsg.xsd
Max. Message Size:	0
Override XML Manager parser limits:	off
Enable MMXDoS Protection:	off
Enable Message Tampering Protection:	off
Enable SQL Injection Protection:	off
Enable X-Virus Scanning:	off
Enable Dictionary Attack Protection:	off

**Back** **Cancel** **XML Threat Protection** **Commit**

\_\_\_ o. Click **Done**. You are returned to the Control Panel.

## 3.2. Validation: XML firewall examination and configuration

Now you examine the newly created XML firewall. Changes are made that are required for the test scenario.

- \_\_\_ 1. On the Control Panel, click **XML Firewall**.
- \_\_\_ 2. The returned XML firewall list is changed from the last time you displayed it; it now contains your new firewall. Click **MyBasicFirewall**.

The new panel is the general editor panel for an XML firewall. The wizard filled out much of the detail for you. In general, the client-facing details are on the lower right, and the server-facing details are on the lower left. General information is usually on the top.



### Note

The Firewall Name and Firewall Type are on the upper left, with the values you supplied in the wizard. On the upper right is the Firewall Policy. The firewall policy controls many of the processing steps to which a message might be subject as it enters or leaves the service or appliance. The wizard generated a policy with the same name as your firewall service.

- \_\_\_ 3. Examine the policy by clicking **Edit** (the [...] button) next to the **Firewall Policy** field.



- 4. The policy editor window opens. More is covered later, but now is a good time to examine some of the policy editor window areas.

### Configure XML Firewall Style Policy

**Policy:**

Policy Name: MyBasicFirewall \*

Apply Policy Cancel Export | View Log | View Status | Close Window |

**Rule:**

Rule Name: MyBasicFirewall\_request Rule Direction: Client to Server

New Rule Delete Rule

Create rule: Click New, drag action icons onto line. Edit rule: Click on rule, double-click on action

Filter Sign Verify Validate Encrypt Decrypt Transform Route AAA Results Advanced

CLIENT ORIGIN SERVER

Create Reusable Rule

Configured Rules		Order	Rule Name	Direction	Actions	
		MyBasicFirewall_request	Client to Server			

[Scroll to top](#)

- a. The **Configured Rules** section at the bottom lists information about each rule within the policy, which, in this case, is just one.

 **Note**

Each rule has a Match action at its beginning to determine whether this rule applies. Therefore, multiple rules can be specified for a single policy, and the Match action for each rule determines its applicability. Rules also have directionality. The single rule here applies only as the request enters the service or appliance.

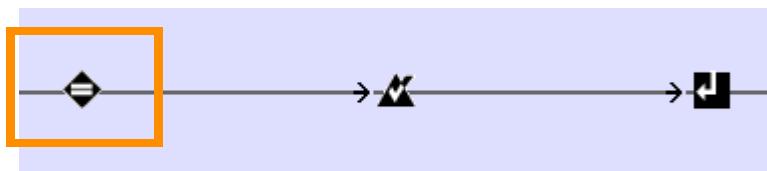
- \_\_\_ b. The rule that is selected in the Configured Rules section is detailed on the Rule Configuration Path (the horizontal line that crosses the center of the dialog).



### Information

A **rule** is composed of a match and its subsequent processing actions. Processing occurs from left to right, as specified on the rule configuration path. In this case, if the request passes the match test, then the message is validated, and then it is moved to the output context (for eventual output).

- \_\_\_ c. Double-click the **Match** action on the rule configuration path to open the rule.



- \_\_\_ 5. On the “Configure a Match Action” page, you see a Matching Rule of **MyBasicFirewall**, which is the name that the wizard generates. Click **Edit** (the [...] button) to open the definition of this rule.



- \_\_\_ 6. On the Configure Matching Rule page, select the **Matching Rule** tab at the top of the window.

The page shows the details of this matching rule.

Matching Rule: MyBasicFirewall [up]

[Apply](#) [Cancel](#) [Undo](#) [Export](#) | [View Log](#) | [View Status](#) ||

Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Match with PCRE	<input type="radio"/> on <input checked="" type="radio"/> off
Boolean Or Combinations	<input type="radio"/> on <input checked="" type="radio"/> off

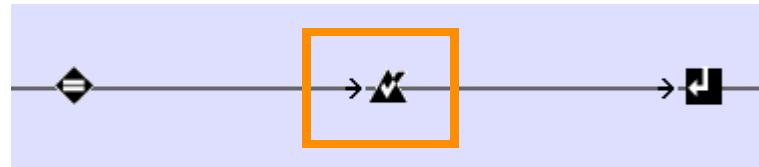


### Note

Notice that the wizard used the same name, **MyBasicFirewall**, for multiple resources in your service: service name, policy name, matching rule name. This works because each of these resources is a separate type of object that is defined within the domain.

- \_\_\_ 7. Click **Cancel** to leave this window.  
\_\_\_ 8. Click **Cancel** again to leave the Configure a Match Action page.

- \_\_\_ 9. Back on the policy editor window, double-click the **Validate** action.



The Configure Validate Action window opens. The **Input** field indicates that the action operates on the document that comes in from the INPUT context (the received document). The validation details look familiar (use schema URL, AddressMsg.xsd, to validate); you entered them in the wizard. Since the validation does not change the document, the **output** field is empty.

 **Configure Validate Action** [Help](#)

**Basic** [Advanced](#)

**Input**

Input	<input type="text" value="INPUT"/>	<input type="button" value="INPUT"/>	*
-------	------------------------------------	--------------------------------------	---

**Options**

 **Validate**

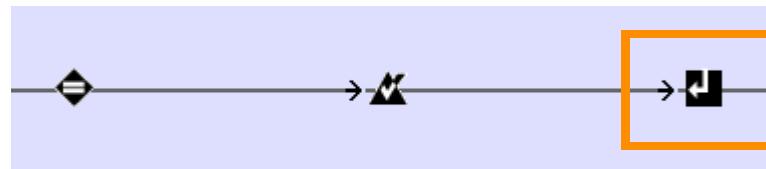
<b>Schema Validation Method</b>	<input type="radio"/> Validate Document via Attribute Rewrite Rule <input type="radio"/> Validate Document via Schema Attribute <input checked="" type="radio"/> Validate Document via Schema URL <input type="radio"/> Validate Document via WSDL URL <input type="radio"/> Validate Document with Encrypted Sections  *
<b>Schema URL</b>	<input type="text" value="local:///"/> <input type="button" value="AddressMsg.xsd"/> <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> <input type="button" value="Var Builder"/>
<b>SOAP Validation</b>	<input type="text" value="Body"/>
<b>Asynchronous</b>	<input type="radio"/> on <input checked="" type="radio"/> off

**Output**

Output	<input type="text" value="OUTPUT"/>	<input type="button" value="OUTPUT"/>
--------	-------------------------------------	---------------------------------------

- \_\_\_ 10. Click **Cancel** to close the window.

- \_\_\_ 11. Back on the policy editor window, double-click the **Results** action.



- \_\_\_ 12. The Configure Results Action window opens. The **Results** action takes the input document (the original document that is passed to the rule), and moves it to the output (output document from the rule processing).

**Configure Results Action** [Help](#)

**Basic** [Advanced](#)

**Input**

Input	<input type="text" value="INPUT"/> INPUT	<input type="button" value="INPUT"/>	*
-------	--	--------------------------------------	---

**Options**

**Results**

<b>Destination</b>	<input type="text" value="cert:///"/> cert:///	<input type="button" value="Upload..."/>	<input type="button" value="Fetch..."/>	<input type="button" value="Edit..."/>	<input type="button" value="View..."/>	<input type="button" value="Var Builder"/>
<b>Asynchronous</b>	<input type="radio"/> on <input checked="" type="radio"/> off					
<b>Number of Retries</b>	<input type="text" value="0"/>					
<b>Retry Interval</b>	<input type="text" value="1000"/> msec					
<b>Method</b>	<input type="button" value="POST"/> *					

**Output**

Output	<input type="text"/> OUTPUT	<input type="button" value="OUTPUT"/>
--------	-----------------------------	---------------------------------------

- \_\_\_ 13. Click **Cancel** to close the window.
- \_\_\_ 14. Click **Cancel** to close the policy editor. If you are prompted about unsaved changes, click **OK**.

- \_\_\_ 15. Back on the firewall editor window, look to the lower left. This section describes the back-end (server) for this service. Since you specified a loopback proxy, the received message is sent back to the client after any processing.

**Back End**

With a loopback proxy back end type, there is no back end server; the device is responding to the client.

With this type, there is no back end server which needs credentials from the device.

With this type, the response type is always "unprocessed".

- \_\_\_ 16. On the lower right, you have the Front End (client) details. Recall these values from the wizard entries: the IP addresses (all configured on the appliance), and on which port the service is listening.
- \_\_\_ 17. So far you have examined the specifications that the wizard generates. You now make one change. The **Request Type** specified in the "Front End" section is **SOAP**. Your incoming message, although it is an XML document, is not wrapped in a SOAP envelope that uses the drop-down list. Change the **Request Type** to **XML**.



- \_\_\_ 18. To commit the change, click **Apply**.

- \_\_\_ 19. Examine the status of the objects that are created to support this service. On the firewall editor window, click **View Status**.



The Object Status window opens. All of the objects have an **op-state** of **up**.

<b>name</b>	<b>config-state</b>	<b>op-state</b>	<b>admin-state</b>	<b>detail</b>	<b>logs</b>
<b>XML Firewall Service</b>					
MyBasicFirewall [XML Firewall Service]	New	up	enabled		
default [XML Manager]	Saved	up	enabled		
default [User Agent]	Saved	up	enabled		
MyBasicFirewall [Processing Policy]	New	up	enabled		
MyBasicFirewall [Matching Rule]	New	up	enabled		
MyBasicFirewall_request [Processing Rule]	New	up	enabled		
MyBasicFirewall_request_validate [Processing Action]	New	up	enabled		
MyBasicFirewall_request_results [Processing Action]	New	up	enabled		

- \_\_\_ 20. Close the Object Status window. You are now ready for testing.

### 3.3. Validation: XML firewall testing

You now use the cURL command-line tool to communicate with the XML firewall that you configured.

- \_\_\_ 1. In the first test case, you use cURL to send a well-formed and valid XML document to the XML firewall. The proxy responds with an identical copy of the request message as the response.
  - \_\_\_ a. Open the `AddressReq.xml` file in your favorite editor and review it before you run this test case.
  - \_\_\_ b. On your Linux desktop, open a command-line terminal by selecting **Computer > Gnome Terminal**.
  - \_\_\_ c. Go to your `<lab_files>/simpleFW` directory. Remember, this directory contains the AddressReq-related files.
  - \_\_\_ d. Type the following command in the command prompt:

```
curl -H "Content-Type: text/xml" --data-binary @AddressReq.xml  
http://<dp_public_ip>:<xmlfw_validation_port>
```

`-H` adds the following Content-Type header to the HTTP headers sent to the service.

`--data-binary` specifies the data to send on the HTTP request. In this case, the at sign (@) indicates that the data is contained in the file named `AddressReq.xml` in the current directory.

The `http://` parameter is the URL of the targeted service.

- \_\_\_ e. You get the `AddressReq.xml` contents echoed back to you, since the XML firewall is not yet doing any transformation.



#### Note

It is a good practice to keep the command prompt window open so you can recall and edit commands, rather than rekeying the same or similar commands. You can use the up and down arrow keys on the keyboard to recall previous commands.

- \_\_\_ 2. In the second test case, edit the `AddressReq.xml` file to force the validation to fail. Resubmit the same command as in the previous test case.
  - \_\_\_ a. Use a text editor to change the `AddressReq.xml` file. You can use one of the following methods to **invalidate** the XML schema:
    - Remove an element
    - Change the spelling of a tag
    - Enter an additional element in the source XML file

**Note**

Do not change the `AddressReq.xml` file so that it causes well-formedness errors. For example, renaming only the start tag and not the end tag does not cause schema validation because it first generates a well-formedness error. The error messages in these cases are different.

- \_\_\_ b. Type the following command in the command prompt:

```
curl -H "Content-Type: text/xml" --data-binary @AddressReq.xml  
http://<dp_public_ip>:<xmlfw_validation_port>
```

- \_\_\_ c. The transmission of this invalid message causes the schema validation action to fail. The service returns a SOAP fault message that indicates Internal Error (from client).
- \_\_\_ d. Remove the changes to `AddressReq.xml` and save.
- \_\_\_ e. Your validating firewall is working. Click **Save Config** to save the XML firewall service properties to the startup configuration.
- \_\_\_ f. Now it is time to build an XML firewall service that transforms the incoming message.

## Create a basic XML firewall to do message transformation

In this section, you create a basic loopback proxy and configure a document processing policy that does simple XSL transformation.

Incoming XML requests are matched against a match rule that determines whether the incoming traffic is subject to a document processing rule. The document processing rule specifies the steps that are applied to incoming documents, and it consists of one or more actions.

You create an XML firewall that does an XSL transformation on incoming XML traffic. The incoming XML request is checked for an exact match with a given name. If the name matches, the corresponding address is returned. If an exact match is not found, an error message is returned as the XML response.

This section implements the scenario that is described in three steps:

- Transformation: XML firewall **creation**
- Transformation: XML firewall **configuration**
- Transformation: XML firewall **testing**

## 3.4. Transformation: XML firewall creation

- \_\_\_ 1. Create a basic XML firewall service to do message transformation.

An XML request/response scenario is simulated in this section. An XML firewall transforms messages on an incoming request message that `AddressReq.xml` represents.

The XML firewall applies the `AddressTransform.xsl` XSL style sheet to the request message.

The XSL style sheet checks for an exact match of the name element and expected values. An error message is returned to the user if the expected values are not found; otherwise, the requested address is returned.

- \_\_\_ a. Open these files in your favorite editor and examine them.
- \_\_\_ b. Create a service. In the Control Panel, click **XML Firewall**.
- \_\_\_ c. On the Configure XML Firewall catalog page, you see the firewall that you tested. To create a firewall service in the previous section, you used the wizard. Unfortunately, there is no wizard that targets transformation. So, for this section, you use the firewall service editor. Click **Add Advanced**.
- \_\_\_ d. You see the editor page similar to the one that you examined in the previous section. For this new firewall service, enter or select the following values:
  - **Firewall Name:** enter `MyTransformFirewall`
  - **Firewall Type:** select **Loopback** (can cause the page to reload)
  - **Local IP address:** enter `<dp_public_ip>`
  - **Device Port:** enter `<xmfw_transform_port>`
  - **Request Type:** select **XML**



### Note

The Firewall Policy is `(none)`. The previous wizard generated that for you. In the next step, you create your own policy.

### 3.5. Transformation: XML firewall configuration

In this step, you create the firewall policy (document processing policy) to do XSL transformation on an incoming message.

- \_\_ 1. Start the creation of a new firewall policy.
  - \_\_ a. Click **New** (the [+] button) to create a policy.



- \_\_ b. The policy editor opens. Enter the **Policy Name**: Policy\_AddressTransformer
- \_\_ c. Click **Apply Policy** to save the policy.
- \_\_ d. Click **New Rule**. A new rule is automatically generated.

- \_\_\_ e. For a new rule, the policy editor automatically adds a **Match** action. To configure it, double-click the Match action icon.

### Configure XML Firewall Style Policy

**Policy:**  
Policy Name: Policy\_AddressTransformer \*

**Rule:**  
Rule Name: Policy\_AddressTransformer\_rule\_0 Rule Direction: Both Directions

Create rule: Click New, drag action icons onto line. Edit rule: Click on rule, double-click on action

Filter Sign Verify Validate Encrypt Decrypt Transform Route AAA Results Advanced

ORIGIN SERVER

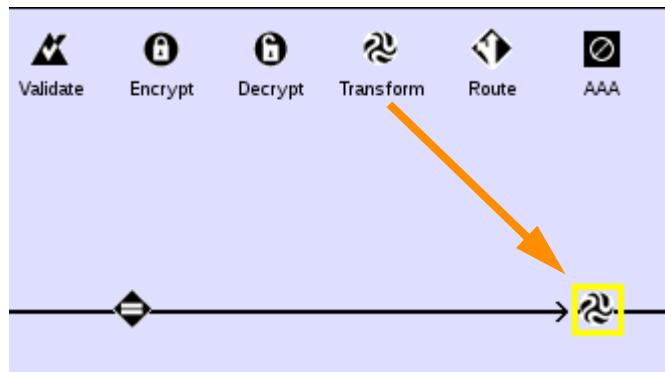
- \_\_\_ f. In the Configure a Match Action page, select the **MyBasicFirewall** matching rule. The wizard in the previous section created this matching rule for you.
- \_\_\_ g. Click **Done** to commit and close the window.



#### Note

As a practice, you might want to prebuild a matching rule that accepts all URLs, so it can be reused in multiple policies. You might prebuild any other application-specific matching rules as well.

- \_\_\_ 2. Add a **Transform** action to the processing steps of the **AddressTransformerPolicy** policy.
- \_\_\_ a. Drag the **Transform** icon onto the rule configuration path, to the right of the **Match** action, and release.

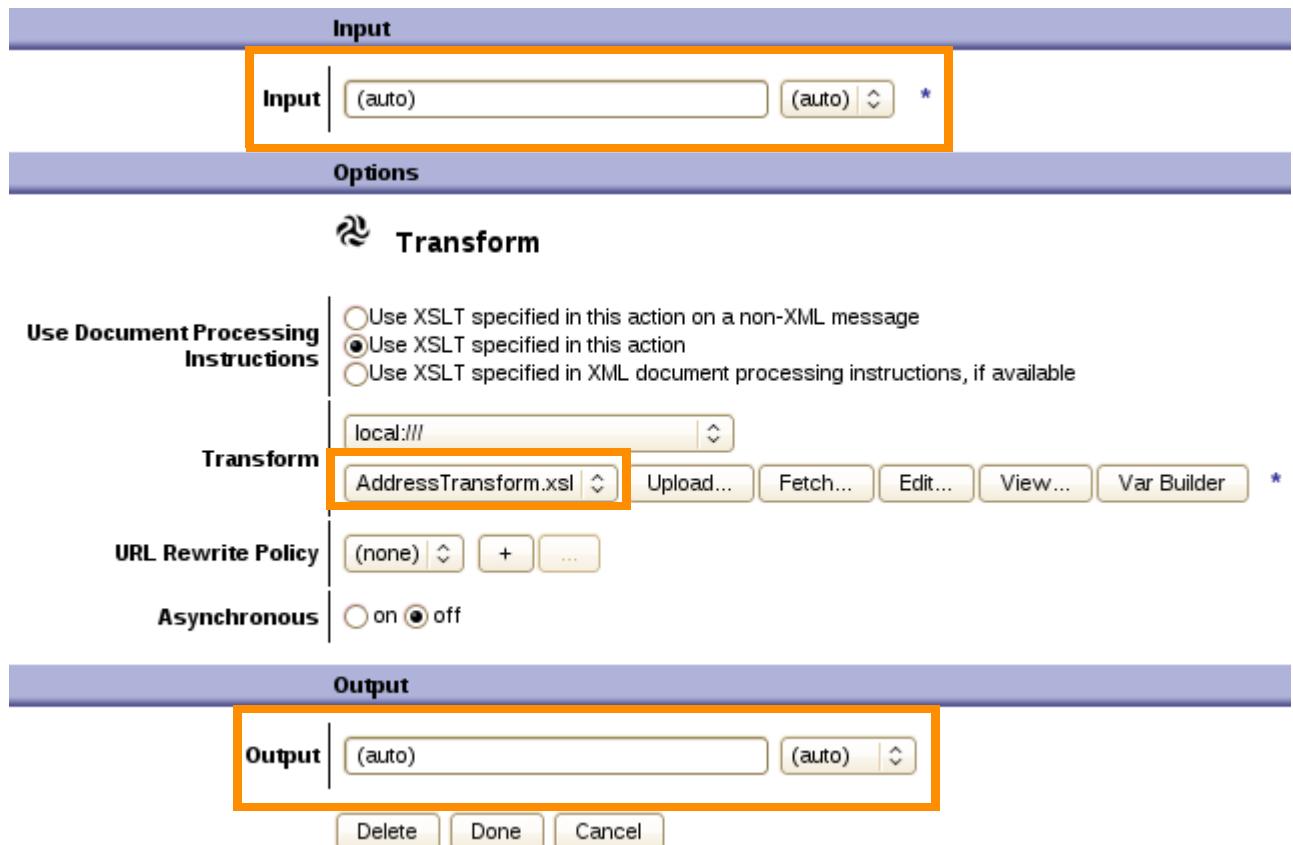


- \_\_\_ b. Double-click the **Transform** action to configure the action.
- \_\_\_ c. In the Configure Transform Action window, select **Use XSLT specified in this action**.

The XSL file that you need is not on the appliance. You must upload it in a way that is similar to how you uploaded the XSD file in the previous section.

- \_\_\_ d. Verify that the **Processing Control File** is set to the `local:` directory.
- \_\_\_ e. Click **Upload** to upload an XSL style sheet to the DataPower appliance.
- \_\_\_ f. Use the **Browse** button to search and select the `AddressTransformer.xsl` style sheet from `<lab_files>/simpleFW`.
- \_\_\_ g. Click **Upload**. You get a successful upload message.
- \_\_\_ h. Click **Continue**.

- \_\_\_ i. Verify that the Configure Transform Action window is displayed with the newly uploaded XSL style sheet.



- \_\_\_ j. Verify that the **Input** field is set to **auto**. This means that the XML document comes from the INPUT context, the received XML document.
- \_\_\_ k. Verify that the **output** field is also set to **auto**. This means that the transformed document is placed in a new context, which the policy editor creates. A later action picks it up from there. In this rule, there is no following action, so the policy editor puts the transformed XML document in the output context.
- \_\_\_ l. Click **Done** to save these configuration changes, and to close the window.
- \_\_\_ 3. Configure the rule to act on XML traffic that comes from the client (a “request rule”), and save the new rule.
- \_\_\_ a. The rule configuration path window reopens. Click **Rule Direction** and select **Client to Server**.



- \_\_\_ b. Click **Apply Policy** to save the policy. The new request rule is created under the configured rules section of the policy editor.
- \_\_\_ c. At the top of the policy editor, click **Close Window**.
- \_\_\_ d. On the Configure an XML firewall page, verify that the **Firewall Policy** field is populated with the **Policy\_AddressTransformer** value.



- \_\_\_ e. Click **Apply** to commit the configuration of the newly created XML firewall service.
- \_\_\_ f. Click **View Status** to verify the operational state of the XML firewall and its related objects. The **op-state** is **up** for all objects.
- \_\_\_ g. Close the Object Status window.
- \_\_\_ h. Back on the Configure XML firewall page, click **Cancel**. You are returned to the Control Panel. The service definition is applied.
- \_\_\_ i. Click **Save Config** to save the XML firewall service properties to the startup configuration.

### 3.6. Transformation: XML firewall testing

You use the cURL command-line tool to communicate with the XML firewall that you configured.

- \_\_\_ 1. In the first test case, you used cURL to send a well-formed and valid XML document to the XML firewall. The firewall service responded with the transformed document.
  - \_\_\_ a. Open the `AddressReq.xml` file in your favorite editor and review it before you run this test case.
  - \_\_\_ b. Open a terminal window.
  - \_\_\_ c. Go to your `<lab_files>/simpleFW` directory.
  - \_\_\_ d. Enter the following command in the command prompt:

```
curl -H "Content-Type: text/xml" --data-binary @AddressReq.xml
http://<dp_public_ip>:<xmlfw_transform_port>
```
  - \_\_\_ e. The XSL style sheet looks for input of “Mr. John Doe”. If it finds the correct values in the XML, it transforms the XML to a related HTML document. Verify that you get the following response (which is not formatted the same way in the command prompt window):

```
<html xmlns:dpedu="http://dpedu.ibm.com"
      xmlns:xalan="http://xml.apache.org/xslt">
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Address Information</title>
</head>
<body>
<table border="0" width="80%">
<tbody>
<tr>
<td width="20%">Name</td><td width="80%"></td>
</tr>
<tr>
<td></td><td>Title: Mr.</td>
</tr>
<tr>
<td></td><td>First name: John</td>
</tr>
<tr>
<td></td><td>Last name: Doe</td>
</tr>
<tr>
<td>Address</td><td></td>
</tr>
<tr>
<td></td><td>Street: 94 On Demand Street</td>
</tr>
<tr>
<td></td><td>City: New York</td>
</tr>
<tr>
<td></td><td>State: NY</td>
</tr>
<tr>
<td></td><td>Zip: 10036</td>
</tr>
</tbody>
</table>
</body>
</html>
```

- \_\_\_ 2. In the second test case, edit the `AddressReq.xml` file by changing the `<address:firstName>` element to something other than “John”. Resubmit the same command as in the previous test case.

- \_\_\_ a. Enter the following command in the command prompt:

```
curl -H "Content-Type: text/xml" --data-binary @AddressReq.xml  
http://<dp_public_ip>:<xmlfw_transform_port>
```

- \_\_\_ b. In this test, the style sheet does not find the required values, so it builds an ERROR response. Verify that you get the following error:

```
html xmlns:dpedu="http://dpedu.ibm.com"  
      xmlns:xalan="http://xml.apache.org/xslt ">  
<head>  
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type">  
<title>Address Information</title>  
</head><body>  
<table width="80%" border="0"><tbody>  
<h5>ERROR</h5>  
<p></p>Error finding address info for: Mr. Jim Doe  
</tbody></body></html>
```

You are now able to configure an XML firewall to do a message transformation on incoming XML documents.

- \_\_\_ c. Remove your changes to `AddressReq.xml`, save the file, and exit the editor.

## Create an XML firewall to integrate validation and transformation

The objective of this section is to provide an introduction to more advanced ESB-like message flows.

Complex message flows are constructed from lower-level, simpler flows from a set of mediation primitives that are logically grouped together to implement some business function.

The first section of this exercise covered the schema validation action, and the second section covered the message transformation action.

The sections covered the configuration that is required to implement each of those actions (primitives) in a simple message flow.

In this section, you use the simpler actions and primitives to implement a more complex message flow.

### Implementation approaches

Two approaches can be used to implement this schema validation and message transformation integrated scenario.

- **Approach #1:** This approach involves building the scenario from the ground up. The steps involve building a new XML firewall and dragging the **validation** and the **Transform** actions onto the rule configuration path, in order. This approach involves running **section 1** and **section 2** sequentially. However, instead of each of these actions having independent document processing policies, the integrated scenario involves the configuration of these two actions on the **same** document processing policy. For any message against which this policy is run, first the incoming message is validated against the XML schema, and thereafter the message is transformed into an appropriate response or error message.
- **Approach #2:** This approach involves building on some of the previous work that you did in **section 1** of this exercise. In section 1, you recall that the configuration of the validation action resulted in the service automatically appending a **Results** action to move the input message to the output unchanged. You can augment this document processing policy by replacing the simple **Results** action with a **Transform** action that uses `AddressTransform.xsl`.

This section uses the second approach to implement the integrated scenario.

The basic steps are:

- Integration: edit the **MyBasicFirewall** document processing policy
- Integration: XML firewall **testing**

### 3.7. Integration: Edit the MyBasicFirewall policy

Edit the **MyBasicFirewall** document processing policy and replace the **Results** action with a **Transform** action that uses the `AddressTransform.xsl` style sheet.

- \_\_\_ 1. To edit the existing policy, open the service first. On the Control Panel, click **XML Firewall**.
- \_\_\_ 2. On the Configure XML Firewall page, select your first firewall service, **MyBasicFirewall**.
- \_\_\_ 3. The Configure XML Firewall page for the MyBasicFirewall service is displayed.
- \_\_\_ 4. Click **Edit** (the [...] button) to modify the MyBasicFirewall Firewall Policy.
- \_\_\_ 5. The MyBasicFirewall policy editor opens. Drag on the **Results** action to the **Delete** (trash can) icon.
- \_\_\_ 6. Drag the **Transform** icon to the right of the **Validate** icon on the configuration path.
- \_\_\_ 7. Double-click the **Transform** icon to open the “Configure Transform Action” window.
- \_\_\_ 8. Click **Use XSLT specified in this action**.
- \_\_\_ 9. Since you already uploaded the style sheet, you can select it from the drop-down list. In the “Processing Control File” section, select the `local:` directory. Use the **(none)** list to select `AddressTransform.xsl`.
- \_\_\_ 10. Verify that the **Input** field is set to **auto**. In this case, the action picks up the document from the INPUT context.
- \_\_\_ 11. Verify that the **output** field is also set to **auto**. The transformed document is moved to the output context.
- \_\_\_ 12. Click **Done**.
- \_\_\_ 13. The rule configuration path window reopens. Click **Apply Policy** and then after the window refreshes, click the **Close Window** link.
- \_\_\_ 14. You are now back on the Configure XML firewall page. Click **Apply** to save the changes.
- \_\_\_ 15. Click **Save Config** to save the XML firewall service properties to the startup configuration.

### 3.8. Integration: XML firewall testing

\_\_\_ 1. In the first test case, you send the `AddressReq.xml` with the expected name. In this case, since the style sheet finds the exact match, it responds with the appropriate HTML message.

- \_\_\_ a. Open the `AddressReq.xml` file in your favorite editor and review it before you run this test case.
- \_\_\_ b. On your Linux desktop, open a command-line terminal by selecting **Computer > Gnome Terminal**.
- \_\_\_ c. Go to your `<lab_files>/simpleFW` directory.
- \_\_\_ d. Type the following command in the command prompt (be sure to enter the port correctly):

```
curl -H "Content-Type: text/xml" --data-binary @AddressReq.xml  
http://<dp_public_ip>:<xmlfw_validation_port>
```

- \_\_\_ e. Verify that you get an HTML response similar to what you received when testing the MyTransformFirewall service previously. It is HTML that contains:

```
<td></td><td>Title: Mr.</td>  
</tr>  
<tr>  
<td></td><td>First name: John</td>  
</tr>  
<tr>  
<td></td><td>Last name: Doe</td>
```

\_\_\_ 2. In the second test case, edit the `AddressReq.xml` file to force the validation to fail. Resubmit the same command as in the previous test case.

- \_\_\_ a. Click the **localuser Home** icon on the desktop to open the Linux desktop File Manager.
- \_\_\_ b. Click the relevant folders, starting with **dplabs** and going down to the `<lab_files>/simpleFW` subdirectory.
- \_\_\_ c. Right-click the `AddressReq.xml` file and select the **gedit** editor.
- \_\_\_ d. Change the `AddressReq.xml` file. You can use one of the following methods to invalidate the XML schema:
  - Remove an element
  - Change the spelling of a tag
  - Enter an additional element in the source XML file
- \_\_\_ e. Save the changes.

- 
- \_\_\_ 3. Type the following command in the command prompt:

```
curl -H "Content-Type: text/xml" --data-binary @AddressReq.xml  
http://<dp_public_ip>:<xmlfw_validation_port>
```

- \_\_\_ 4. Verify that you get a returned document with a <faultstring> of Malformed content (from client). Later, you learn some troubleshooting skills to further debug this problem.
- \_\_\_ 5. Remove the changes that you made to AddressReq.xml.
- \_\_\_ 6. You might also want to try one more test where you change the <name> to something other than “John”. Run the cURL command. The response is an HTML-tagged document that indicates an ERROR.

You are now able to configure an XML firewall to do a message validation and transformation on incoming XML documents.

- \_\_\_ a. Remove all of your changes to AddressReq.xml.

The best way to confirm the actual path of the message through the message flow is to use the multi-step probe facility. This technique is shown later.

## End of exercise

## Exercise review and wrap-up

In this exercise, you created an XML firewall that does message validation and transformation. In the first two examples, you created a separate service and policy for each operation. In the third section, two actions are integrated into a single policy.

# Exercise 4. Creating an advanced multi-protocol gateway

## What this exercise is about

This exercise shows you how to create a multi-protocol gateway (MPGW) service from a WSDL file. You learn how to configure an MPGW document processing policy with actions. Content-based routing is configured by creating an MPGW that contains a document processing policy with a Route action. You learn the steps that are required to create, configure, and test MPGWs.

## What you should be able to do

At the end of the lab, you should be able to:

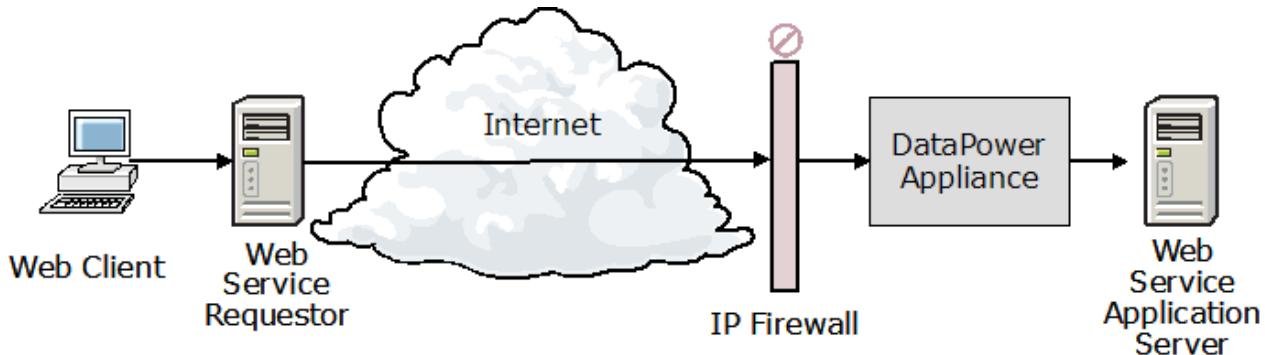
- Create an MPGW from a WSDL definition
- Configure a document processing policy with more actions
- Configure content-based routing by using a Route action
- Test the MPGW policy by using the command-line tool cURL
- Perform basic debugging by using the system log and multistep probe

## Introduction

The DataPower SOA appliance provides the capabilities to set up multiple MPGWs, or virtual firewalls, to protect back-end applications. Each of these firewalls operates on a port with a policy that consists of a set of rules. Each rule contains actions that complete one or more functions.

An MPGW has similar functions to a proxy server. Both services examine an incoming XML request from a client. Both services complete some processing on the request. Then, both services either reject the request or possibly transform the message. Finally, the message can be forwarded to a back-end server that hosts the application. The MPGW service can also be configured to accomplish content based routing, where an appropriate back-end server is

selected based on the contents of the message. A typical deployment scenario is pictured here.



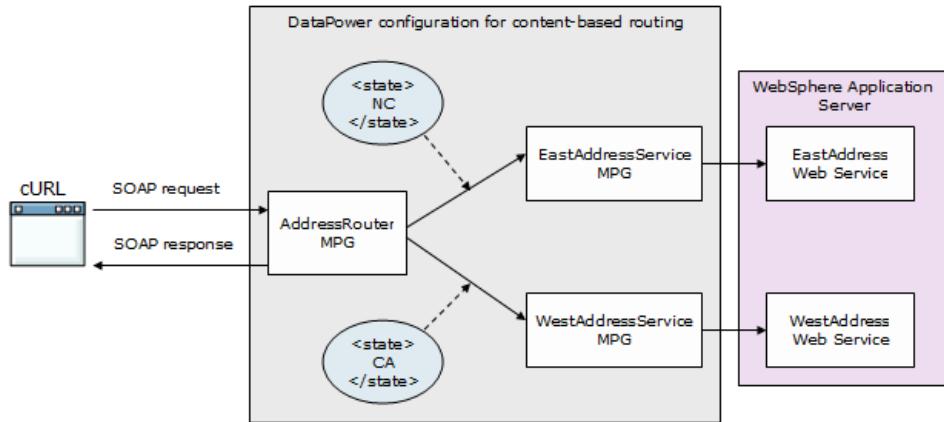
## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- The **Address Search** web services that run on WebSphere Application Server
- Access to the `<lab_files>` directory

## Exercise instructions

You first configure two MPGWs to statically route messages to a preconfigured web service. The third firewall fronts these two firewalls and dynamically routes messages to either of these statically bound firewalls, depending on the contents of the <state> field in the incoming SOAP request. The overall architecture is pictured here.



This exercise implements the scenario that is described in five steps:

- Step 1: Create, configure, and test EastAddressService
- Step 2: Create, configure, and test WestAddressService
- Step 3: Create and configure an AddressRouter MPGW
- Step 4: Configure the East and West Address MPGWs for routing
- Step 5: Perform end-to-end content based routing (CBR) scenario testing

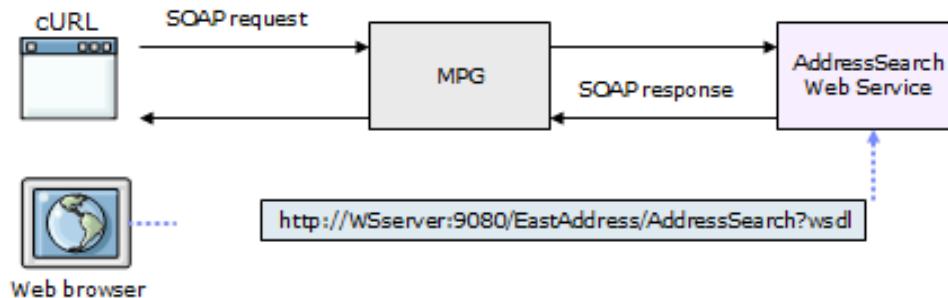
## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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
  - <dp\_internal\_ip>: IP address of the DataPower appliance development and administrative functions
  - <dp\_public\_ip>: IP address of the public services on the appliance
  - <dp\_WebGUI\_port>: instructor-assigned address for the WebGUI
  - <backend\_server\_ip>: IP address for the services that run in WebSphere Application Server (AddressSearch web services, LDAP, registry)

- *<was\_server\_port>*: port number for the services that are running in WebSphere Application Server (AddressSearch web services, LDAP, registry)
- *<mpgw\_east\_port>*: instructor-assigned port number for the MPGW that handles the East Address web service
- *<mpgw\_west\_port>*: instructor-assigned port number for the MPGW for the West Address web service
- *<mpgw\_content\_based\_routing\_port>*: instructor-assigned port number for the MPGW that provides the route function
- *<localhost\_address>*: IP address that points to the local system itself (usually 127.0.0.1)

## 4.1. Create a basic MPGW to validate SOAP messages

In this exercise, you create an MPGW that uses a WSDL file of the web service. The MPGW is tested with cURL by sending already created SOAP messages to the web service by an MPGW, which is configured to forward to the web service.



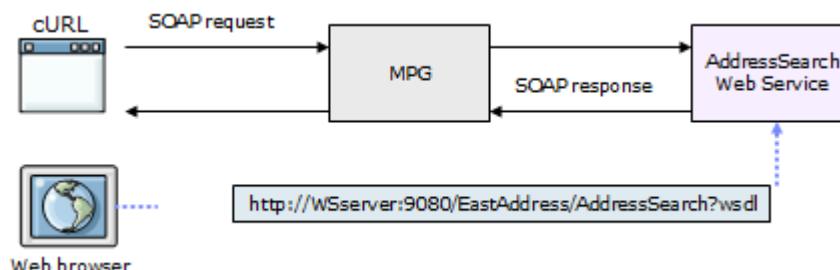
This lab implements the scenario that is described in four essential steps:

- Section 1: WSDL review
- Section 2: MPGW creation
- Section 3: FSH (front-side handler) creation
- Section 4: MPGW testing
- Section 5: Working with the debugging tools

### **Section 1: WSDL review**

- 1. The MPGW that is created is used to validate SOAP messages to and from a preconfigured web service. Hence, it is critical that you understand the WSDL definitions. This step gives you an opportunity to study the specific WSDL file before you implement the steps to actually create and configure the MPGW.
  - a. Using any text editor such as gedit, open and review the `EastAddressSearch.wsdl` file that is located in the `<lab_files>/AdvMPGW` directory.

The most important artifacts to understand when communicating with a web service are the definitions that are found in the WSDL file.



## Section 2: Create a multi-protocol gateway for EastAddressSearch

Configure a new multi-protocol gateway on the IBM WebSphere DataPower SOA Appliance to forward requests to the `EastAddressSearch` web service.

- \_\_\_ 1. In the IBM WebSphere DataPower SOA Appliance WebGUI, click **Control Panel > Multi-Protocol Gateway**.



- \_\_\_ 2. From the Configure Multi-Protocol Gateway screen, click **Add**.
- \_\_\_ 3. Enter `EastAddressSearch` as the name of the new gateway. You can optionally enter a description for the gateway in the summary field, such as `East Address Search MPGW`.
- \_\_\_ 4. In the **XML Manager** field, leave the **default** XML manager selected.
- \_\_\_ 5. In the **Multi-Protocol Gateway Policy** field, click **+** (new) to create a multi-protocol gateway policy.

**Configure Multi-Protocol Gateway**

General Advanced Stylesheet Params Headers Monitors WS-Addressing WS-ReliableMes

Apply Cancel

**General Configuration**

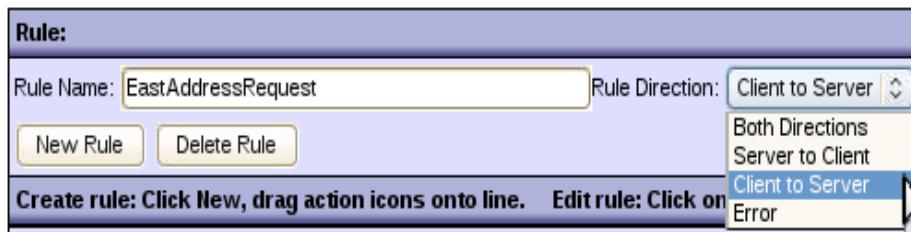
<b>Multi-Protocol Gateway Name</b>	<input type="text" value="EastAddressSearch"/> *	<b>XML Manager</b>	<input type="text" value="default"/> + ... *
<b>Summary</b>	<input type="text" value="East Address Search MPG "/>	<b>Multi-Protocol Gateway Policy</b>	<input type="text" value="(none)"/> + ... *
<b>Type</b>	<input type="radio"/> dynamic-backend <input checked="" type="radio"/> static-backend	<b>URL Rewrite Policy</b>	<input type="text" value="(none)"/> + ...

- \_\_\_ 6. Enter `EastAddressSearch` as the processing policy name.

- \_\_\_ 7. Configure a message processing rule to forward any client request message to the back-end service.
- \_\_\_ a. In the processing rule editor for EastAddressSearch, click **New Rule** and rename the rule to: EastAddressRequest



- \_\_\_ b. Set the processing rule direction to **Client to Server**.



- \_\_\_ c. Double-click the **Match** action icon to configure it.
- \_\_\_ d. Click + (new) to add a **matching rule**.
- \_\_\_ e. Create a matching rule with the following settings:
- **Name:** MatchAll
  - **Match with PCRE** (Perl-compatible regular expression): **off**
  - **Boolean Or Combinations:** **off**

The screenshot shows the 'Matching Rule' configuration dialog. It has 'Apply' and 'Cancel' buttons at the top. Below that is a 'Name' field containing 'MatchAll'. Under 'Administrative State', there are two radio buttons: 'enabled' (selected) and 'disabled'. The 'Comments' field is empty. Under 'Match with PCRE', there are two radio buttons: 'on' (unchecked) and 'off' (selected). Under 'Boolean Or Combinations', there are two radio buttons: 'on' (unchecked) and 'off' (selected).

- \_\_\_ f. Click the **Matching Rule** tab, and click **Add** to add a matching rule.
  - \_\_\_ g. Set the **Matching Type** to **URL** and the **URL Match** property to: \*
  - \_\_\_ h. Click **Apply** to commit the new matching rule properties.
  - \_\_\_ i. Click **Apply** to save the matching rule configuration.
  - \_\_\_ j. Click **Done** to use the new `MatchAll` matching rule.
- \_\_\_ 8. Add a **Transform** action that remaps the namespace of the incoming request to the expected namespace.



### Note

The East AddressSearchService web service (back-end service) expects the application namespace value of `http://east.address.training.ibm.com` for all incoming messages. However, some messages might not have this namespace value.

In a later section of this exercise, you set up an MPGW that does content based routing (CBR). This MPGW routes the message to the EastAddressSearch firewall that was created. Messages sent to the content based routing MPGW use a generic namespace value (for example, `http://address.training.ibm.com`), which is remapped to the correct value (for example, `http://east.address.training.ibm.com`). Otherwise, schema validation in the EastAddressSearch MPGW fails.

- \_\_\_ a. In the rule configuration area, drag a **Transform** action after the **Match** action.



- \_\_\_ b. Double-click the **Transform** action to open the **Transform** action configuration window.
- \_\_\_ c. Verify that **Use XSLT specified in this action** is selected.
- \_\_\_ d. Click **Upload** to upload a style sheet that is called `<lab_files>\AdvMPGW\Address-EastRenameNamespace.xsl` that copies the input document but changes the application namespace to `http://east.address.training.ibm.com` for all incoming messages.
- \_\_\_ e. Click **Continue**.

- \_\_\_ f. After uploading the style sheet, make sure that the Transform configuration window contains the values, as shown:

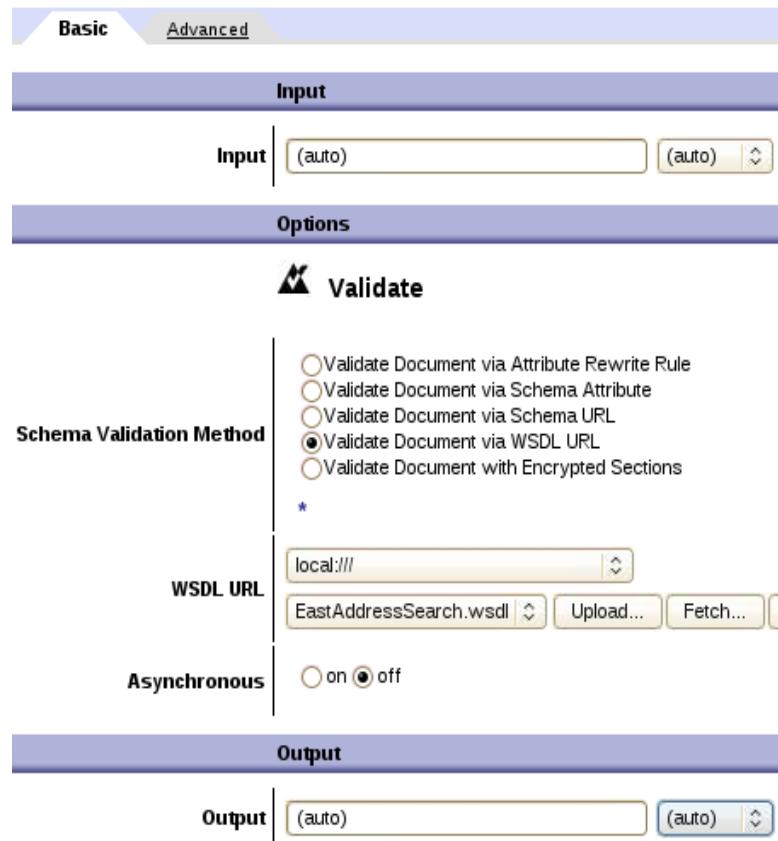


- \_\_\_ 9. Click **Done**.
- \_\_\_ 10. Add a **Validate** action that validates the incoming WSDL request.
- \_\_\_ a. In the rule configuration area, drag a **Validate** action after the **Transform** action.



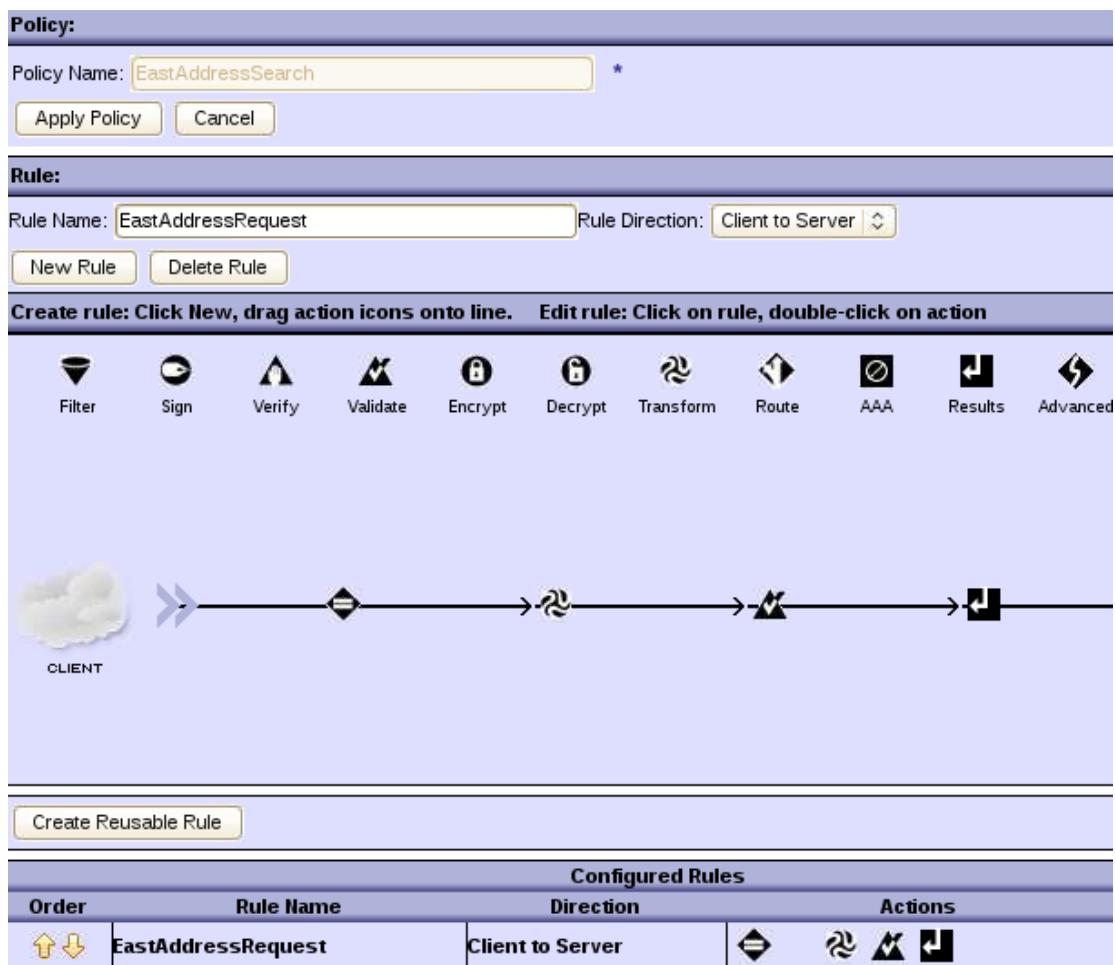
- \_\_\_ b. Double-click the **Validate** action to open the **Validate** action configuration window.
- \_\_\_ c. Verify that **Validate Document via a WSDL URL** is selected.
- \_\_\_ d. Click **Upload** to upload a .wsdl file that is called <lab\_files>\AdvMPGW\EastAddressSearch.wsdl that copies the input document.
- \_\_\_ e. Click **Continue**.

- \_\_\_ f. After uploading the wsdl, make sure that the Validate configuration window contains the values, as shown:

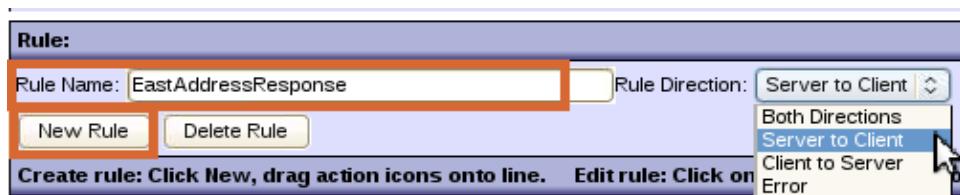


- \_\_\_ 11. Click **Done**.
- \_\_\_ 12. Add a **Results** action to the `EastAddressRequest` document processing rule to forward the results to the back-end service.
- In the processing rule editor for `EastAddressRequest`, drag a **Results** action after the **Validate** action.
  - Double-click the **Results** action to configure it.
  - Confirm that both the **Input** and **Output** parameters are set to **(auto)**. It is also acceptable for the output parameter to be **OUTPUT**.
  - Click **Done** to save the **Results** action settings.
  - Click **Apply Policy** to save the changes.

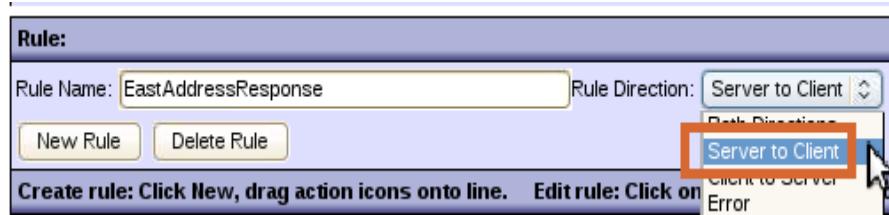
- \_\_\_ f. Confirm that the new document processing rule is in the list of configured rules.



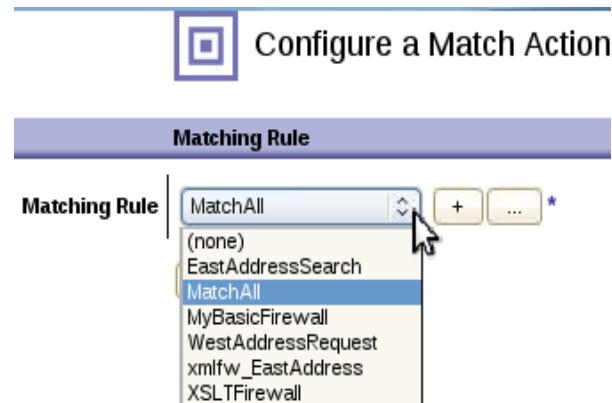
- \_\_\_ 13. Create a second document processing rule named EastAddressResponse. The response rule returns the message from the back side server through the DataPower appliance back to the client.
- \_\_\_ a. In the processing rule editor for AddressSearchMPGWPOLICY, click **New Rule** and rename the rule to: EastAddressResponse



- \_\_\_ b. Set the processing rule direction to **Server to Client**.



- \_\_\_ c. Double-click the **Match** action icon to configure it.  
 \_\_\_ d. Click the list box arrows and select the **MatchAll** matching rule.

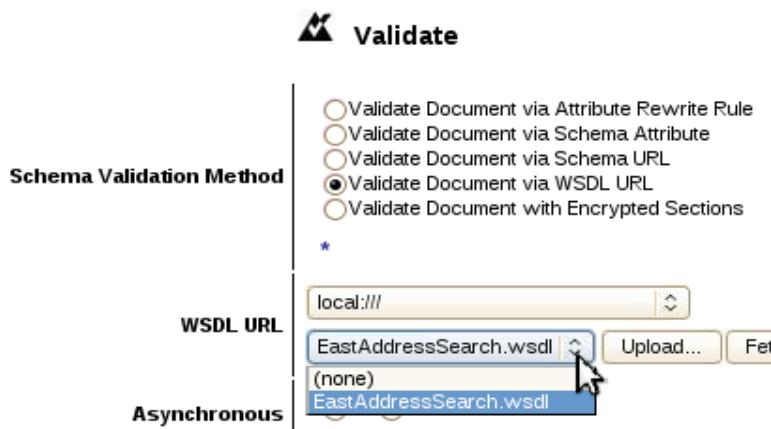


- \_\_\_ e. Click **Done** to use the MatchAll matching rule.  
 14. Add a **Validate** action that validates the incoming WSDL request.  
 \_\_\_ a. In the rule configuration area, drag a **Validate** action after the **Match** action.



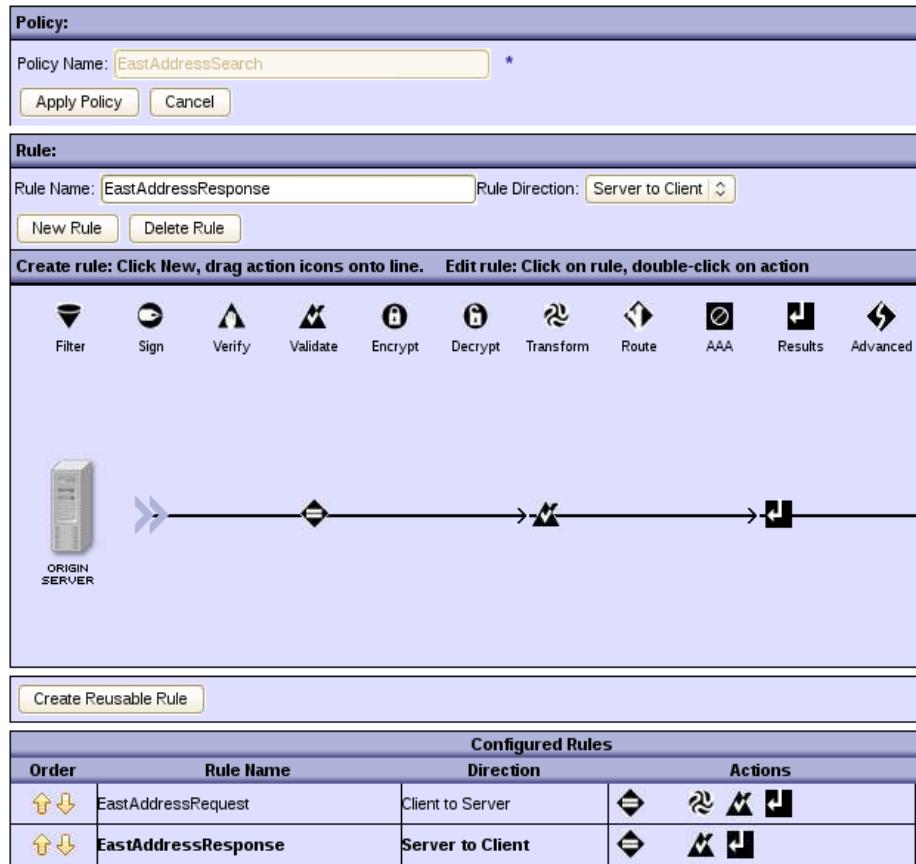
- \_\_\_ b. Double-click the **Validate** action to open the **Validate** action configuration window.  
 \_\_\_ c. Verify that **Validate Document via a WSDL URL** is selected.

- \_\_\_ d. Select the `EastAddressSearch.wsdl` in the local directory on the appliance by clicking the list box arrows that are located in the WSDL URL section.



- \_\_\_ 15. Click **Done**.
- \_\_\_ 16. Add a **Results** action to the `EastAddressResponse` document processing rule to forward the results to the back side service.
- \_\_\_ a. In the processing rule editor for `EastAddressResponse`, drag a **Results** action after the **Validate** action.
- \_\_\_ b. Double-click the **Results** action to configure it.
- \_\_\_ c. Confirm that both the **Input** and **Output** parameters are set to **(auto)**. It is also acceptable for the output parameter to be **OUTPUT**.
- \_\_\_ d. Click **Done** to save the **Results** action settings.
- \_\_\_ e. Click **Apply Policy** to save the changes.

- \_\_\_ f. Confirm that the new document processing rule is in the list of configured rules.



- \_\_\_ 17. Click **Apply Policy** and then **Close Window** to close the `EastAddressSearch` document policy editor.



### Note

The multi-protocol gateway policy has two document processing rules that are defined:

- **EastAddressRequest** accepts request messages from the client with a URL path of `/EastAddressSearch` and forwards the messages to the back-end server.
- **EastAddressResponse** accepts any response message from the back-end server and forwards the request to the client.

The remaining steps in this section assign the address of the actual `EastAddressSearch` web service and various quality of service settings.

- \_\_\_ 18. Set a static back-end connection to the `EastAddressSearch` web service.

- \_\_\_ a. On the Configure Multi-Protocol Gateway page, select **static-backend** as the back-end connection type.

**General Configuration**

<b>Multi-Protocol Gateway Name</b>	<b>XML Manager</b>
<input type="text" value="EastAddressSearch"/> *	<input type="button" value="default"/> <input type="button" value="+"/> <input type="button" value="..."/> *
<b>Summary</b>	<b>Multi-Protocol Gateway Policy</b>
<input type="text" value="East Address Search MPG"/>	<input type="button" value="EastAddressSearch"/> <input type="button" value="+"/> <input type="button" value="..."/> *
<b>Type</b>	<b>URL Rewrite Policy</b>
<input type="radio"/> dynamic-backend <input checked="" type="radio"/> static-backend	<input type="button" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>

- \_\_\_ b. Set the back-end URL to:

`http://WSserver:9080/EastAddress/services/AddressSearch`

**Important**

Remember to replace `<backend_server_ip>` with the value provided to you by the instructor. The value of the back-end server is either a literal IP address or a static host alias, such as `WSserver`.

**Back side settings**

<b>Backend URL</b>
<input type="text" value="http://9.10.15.25:9080/EastAddr"/> *
<input type="button" value="MQHelper"/> <input type="button" value="TibcoEMSHelper"/>
<input type="button" value="WebSphereJMSHelper"/> <input type="button" value="IMSConnectHelper"/>

- \_\_\_ 19. Make sure that the **Response Type** field is set to **SOAP**.

**Information**

The **Request Type** and **Response Type** settings determine the validation steps that are applied by the gateway to incoming and outgoing messages.

- **SOAP** validates the message against the SOAP schema in use. In effect, the gateway checks whether the message contains a valid SOAP envelope for web service messages.
- **XML** verifies that the message contains an XML document.
- **Pass-Thru** literally transmits the message through the gateway without any processing or modification.

- **Non-XML** represents flat file text or any binary file format. The gateway passes the message itself without modification. However, processing rules can authenticate, authorize, or route the message to another destination.

20. Make sure that the **Back attachment processing format** is set to **Dynamic**.



### Information

Instead of converting large amounts of binary data into text, many web service engines allow binary data to be stored as attachments to the SOAP message. There are two common binary data encoding schemes that are used in the industry:

- **MIME (Multipurpose Internet Mail Extensions)** defines an encoding format for sending binary information over Internet email messages. Some web services use this scheme to efficiently transmit binary files as an attachment on a text-based SOAP message. Most web services engines, including the IBM WebSphere web services runtime, support this standard.
- **DIME (Direct Internet Message Encapsulation)** was a separate Internet standard that Microsoft proposed as a simpler method to encapsulate binary information in a web service message. Although some web services have supported DIME, the newer SOAP Message Transmission Optimization Mechanism (MTOM) specification now supersedes DIME.

Unfortunately, most vendors support only one scheme or the other. DataPower SOA appliance services, such as the multi-protocol gateway, solve this problem by automatically converting attachments between these two types.

The **front attachment processing format** setting determines how the service interprets attachments that are sent from the client. On the other side, the **back attachment processing format** determines how the service encodes attachments in outgoing messages from the service to the back-end resource.

For example, to convert message attachments from a DIME format to MIME, set the **front attachment processing format** to **DIME** and the **back attachment processing format** to **MIME**.

These settings affect messages that contain attachments.

21. Leave the **Back Side Timeout** value set to **120** seconds (2 minutes).

22. Leave the **Stream Output to Back** set to **Buffer Messages**.



### Information

The other setting, **stream-until-infraction**, continuously sends parts of the request message to the back-end resource unless the gateway encounters some information that causes the message to fail validation.

- \_\_\_ 23. Leave the **HTTP Version to Server** set to **HTTP 1.1**.
- \_\_\_ 24. Change the **Propagate URI** setting to **off**.



### Information

If the **Propagate URI** setting is **on**, the gateway overwrites the back-end URL value with the original URL path from the client.

For example:

- The client sends a request to the multi-protocol gateway with a URL value of `http://<dp_public_ip>:<mpgw_east_port>/EastAddressSearch`.
- The back-end URL value is set to `http://<backend_server_ip>:9080/EastAddress/services/AddressSearch`.

With **Propagate URI** set to **on**, the gateway forwards the request message with a URL value of `http://<backend_server_ip>:9080/EastAddressSearch`.

\_\_\_ 25. Leave the **Compression** set to **off**.

Your back-end resource settings look like the following picture.

The screenshot shows a configuration interface with the following sections:

- Response Type**: Options include JSON, Non-XML, Pass-Thru, SOAP (selected), and XML.
- Back attachment processing format**: Options include Dynamic (selected), MIME, DIME, and Detect.
- Back Side Timeout**: A text input field containing "120" with a mandatory asterisk (\*) next to it.
- Stream Output to Back**: Options include Buffer Messages (selected) and Stream Messages.
- HTTP Version to Server**: Options include HTTP 1.0 and HTTP 1.1 (selected).
- Propagate URI**: Options include on (selected) and off.
- Compression**: Options include on (selected) and off.

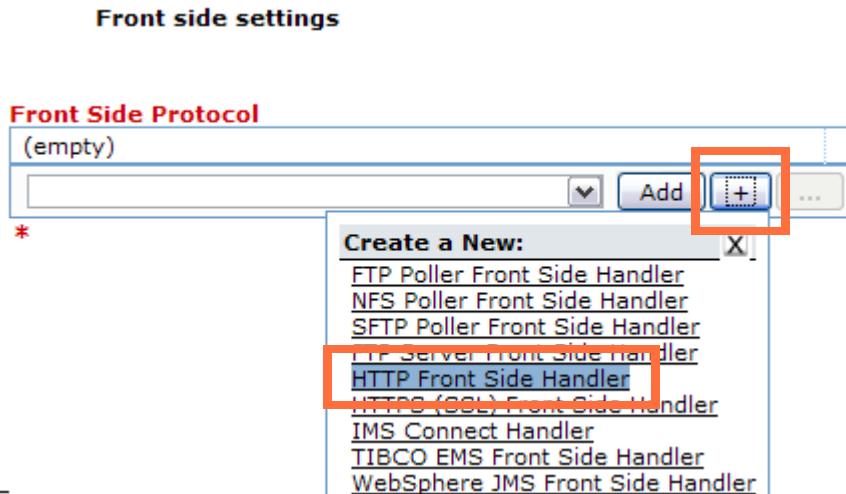
### **Section 3: Configure an HTTP front side protocol handler**

The multi-protocol gateway is configured to forward requests to the East Address Search web service. However, the gateway does not support any incoming requests at the moment.

Create a front side protocol handler to receive client requests over an HTTP connection.

- \_\_\_ 1. Create an HTTP front side protocol handler named `EastAddressSearch`.
  - \_\_\_ a. In the Front Side Protocol section, click **+** (new).

- \_\_\_ b. Select **HTTP Front Side Handler** from the list.



- \_\_\_ c. Configure the new HTTP front side handler with the following values. Leave all other settings to the default values.
- **Name:** EastAddressSearch
  - **Local IP address:** <dp\_public\_ip>
  - **Port Number:** <mpgw\_east\_port>, as assigned by the instructor

HTTP Front Side Handler

<b>Name</b>	<input type="text" value="EastAddressSearch"/>
<hr/>	
<b>Administrative State</b>	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
<hr/>	
<b>Comments</b>	<input type="text"/>
<hr/>	
<b>Local IP Address</b>	<input type="text" value="dp_public_ip"/>
<hr/>	
<b>Port Number</b>	<input type="text" value="657"/>
<hr/>	
<b>HTTP Version to Client</b>	<input type="text" value="HTTP 1.1"/> <input type="button" value="▼"/>

**Note**

The local IP address value of <dp\_public\_ip> allows the front side protocol handler to receive messages on all of the Ethernet interfaces on the IBM WebSphere DataPower SOA Appliance.

- \_\_\_ d. Click **Apply** to save the changes that are made to the HTTP front side handler.
- \_\_\_ 2. Apply the multi-protocol gateway.
  - \_\_\_ a. Click **Apply** to save the changes that are made to the multi-protocol gateway.

 Configure Multi-Protocol Gateway

( General Advanced Stylesheet Params

Multi-Protocol Gateway status: [up]

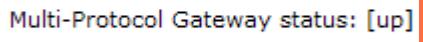
**General Configuration**

**Multi-Protocol Gateway Name**  
EastAddressSearch \*

**Summary**  
East Address Search MPG

- \_\_\_ b. Verify that the Multi-Protocol Gateway status is **up**.

 Multi-Protocol Gateway status: [up]

**General Configuration**

- \_\_\_ c. Click **Save Config**.
- \_\_\_ 3. Examine the multi-protocol gateway settings for all front side protocol handlers.
  - \_\_\_ a. Locate the **Request Type** setting below the Front Side Protocol list. Verify that the expected request message type is **SOAP**, matching the setting for the Response Type.
  - \_\_\_ b. Leave the **Front attachment processing format** to **Dynamic**.
  - \_\_\_ c. The **Front Side Timeout** value determines the number of seconds any front side handler idles before abandoning a transaction. Leave this value at **120** seconds, or 2 minutes.

- \_\_\_ d. The **Stream Output to Front** setting determines whether the gateway can start sending portions of the response message back to the client. Leave this setting at **Buffer Messages**.

<b>Request Type</b>	
<input type="radio"/> JSON	
<input type="radio"/> Non-XML	
<input type="radio"/> Pass-Thru	
<input checked="" type="radio"/> SOAP	
<input type="radio"/> XML	
<b>Front attachment processing format</b>	
<input checked="" type="radio"/> Dynamic	
<input type="radio"/> MIME	
<input type="radio"/> DIME	
<input type="radio"/> Detect	
<b>Front Side Timeout</b>	
120	*
<b>Stream Output to Front</b>	
<input checked="" type="radio"/> Buffer Messages	
<input type="radio"/> Stream Messages	

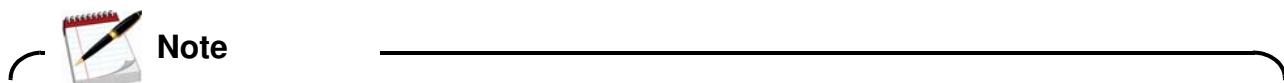
- \_\_\_ e. The Multi-Protocol Gateway `EastAddressSearch` is now ready for testing.

#### Section 4: *EastAddressSearch MPGW testing*

Test the MPGW configuration by opening a command prompt window and running cURL. The following steps ensure that the back-end web service is 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. Open a command prompt and go to the `<lab_files>/AdvMPGW` directory.
- \_\_\_ 2. Run the following command:  

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary
@findByLocation.xml
http://<dp_public_ip>:<mpgw_east_port>/EastAddressSearch
```



The `findByLocation.xml` file is the SOAP message that submits a request to the web service to search for addresses by providing either a state or city name. This file is used to test the functional use of the MPGW.

- \_\_\_ 3. Verify in the command prompt that an XML list of addresses is returned.

- \_\_\_ 4. Open the `findByLocation.xml` file in a text editor. Notice that the application namespace value is `http://address.training.ibm.com`. The **Transform** action changes this namespace value to `http://east.address.training.ibm.com` so that validation succeeds.

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:q0="http://address.training.ibm.com" >
  <soapenv:Body>
    <q0:findByLocation>
      <city></city>
      <state>NC</state>
    </q0:findByLocation>
  </soapenv:Body>
</soapenv:Envelope>
```

- \_\_\_ 5. Change the value of `<state>` to: CA

For example, `<state>CA</state>` would be in the `findByLocation.xml` prebuilt SOAP message. Run the cURL command again. You get a fault string of:

`com.ibm.training.address.AddressNotFoundException: Cannot find any address entries that are in CA.`

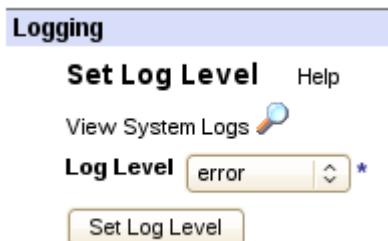
Change the value of `<state>` to NC. You send this message to the East Address MPGW again in the next section.

Modify the `findByLocation.xml` file to force a schema validation failure. You can change an element name. You now get a fault string of Internal Error. Check the system logs for error messages. Be sure to undo this change before proceeding.

## Section 5: Working with the debugging tools

Unfortunately, testing does not always go perfectly, so you need some debugging skills and tools. In this section, you first examine what the system log contains, how to change its level of detail, and how to use the multi-step probe to analyze flow through a rule.

- \_\_\_ 1. Click **Control Panel > troubleshooting**.
- \_\_\_ 2. Verify that the log level is at **error**. If not, set it to that value, and click **Set Log Level**.



- 
- \_\_\_ 3. Click **View System Logs**. A log window opens. Ignore it for the next few steps.
  - \_\_\_ 4. First, see what a correctly running service log looks like. Make sure that the `findByLocation.xml` file is back to its original valid state.
  - \_\_\_ 5. Go back to your command prompt from the previous section, and resend the cURL command:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary
@findByLocation.xml
http://<dp_public_ip>:<mpgw_east_port>/EastAddress/services/AddressSe
arch
```

A list of addresses in NC is returned.

- \_\_\_ 6. Switch back to your log window, and click **Refresh Log**.

The most current entry is at the top. The message flows successfully. Depending on what the original log level and activity is, you might have few or no entries that are related to running the service for this message.

- \_\_\_ 7. On the troubleshooting panel, change the log level to **debug**. Click **Set Log Level**.
- \_\_\_ 8. Click **Confirm**.



#### Note

The recommendation is to run a log level of **debug** only when necessary.

- \_\_\_ 9. Run the cURL command again. You still see addresses returned.
  - \_\_\_ 10. On the log window, click **Refresh Log**.
  - \_\_\_ 11. Search the entries until you see the new transaction entry (the message that first hit the policy). Reading up from there, you see:
    - The **Match** action is satisfied
    - The **Transformation** of the namespace
    - The **Validation** of the message
    - The call to the back side server

After that, you see the processing of the response.

12:27:03	mpgw	debug	2264719		0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Header-Retention Policy: TE = OFF. TE Header = (NULL), URL: http://WSserver:9080/EastAddress/services/AddressSearch
12:27:03	network	debug	2264719		0x80e003ca	xmlmgr (default): Attempting TCP connect to WSserver
12:27:03	multipstep	info	2264719	request	0x80c00002	mpgw (EastAddressSearch): rule (EastAddressRequest): #3 results: 'generated from dpvar_1' completed OK.
12:27:03	memory-report	debug	2264719	request	0x80e0068d	mpgw (EastAddressSearch): Processing [Rule (EastAddressRequest), Action ('EastAddressRequest_results_0', results()), Input(dpvar_1), Output(NULL)] finished: memory used 294440
12:27:03	multipstep	info	2264719	request	0x80c00002	mpgw (EastAddressSearch): rule (EastAddressRequest): #2 validate: 'dpvar_1 wsdl local:///EastAddressSearch.wsdl' completed OK.
12:27:03	memory-report	debug	2264719	request	0x80e0068d	mpgw (EastAddressSearch): Processing [Rule (EastAddressRequest), Action ('EastAddressRequest_validate_0', validate()), Input(dpvar_1), Output(NULL)] finished: memory used 273016
12:27:03	ws-proxy	debug	2264719		0x80a002ac	xmlmgr (default): wsdl Compilation Request: Found in cache (local:///EastAddressSearch.wsdl)
12:27:03	ws-proxy	debug	2264719		0x80a002aa	xmlmgr (default): wsdl Compilation Request: Checking cache for URL local:///EastAddressSearch.wsdl
12:27:03	multipstep	info	2264719	request	0x80c00002	mpgw (EastAddressSearch): rule (EastAddressRequest): #1 xform: 'Transforming INPUT with local:///Address-EastRenameNamespace.xsl results stored in dpvar_1' completed OK.
12:27:03	memory-report	debug	2264719	request	0x80e0068d	mpgw (EastAddressSearch): Processing [Rule (EastAddressRequest), Action ('EastAddressRequest_xform_0', xform(local:///Address-EastRenameNamespace.xsl))), Input(INPUT), Output(dpvar_1)] finished: memory used 218072
12:27:03	multipstep	debug	2264719	request	0x80c0004e	mpgw (EastAddressSearch): Stylesheet URL to compile is 'local:///Address-EastRenameNamespace.xsl'
12:27:03	xmlparse	debug	2264719	request	0x80e003ab	mpgw (EastAddressSearch): Finished parsing: http://172.16.78.44:6957/EastAddressSearch
12:27:03	xmlparse	debug	2264719	request	0x80e003a6	mpgw (EastAddressSearch): Parsing document: 'http://172.16.78.44:6957/EastAddressSearch'
12:27:03	xslt	debug	2264719		0x80a002ac	xmlmgr (default): xslt Compilation Request: Found in cache (local:///Address-EastRenameNamespace.xsl)
12:27:03	xslt	debug	2264719		0x80a002aa	xmlmgr (default): xslt Compilation Request: Checking cache for URL local:///Address-EastRenameNamespace.xsl

- \_\_\_ 12. Edit the `findByLocation.xml` file. Change the state to `CA` and run the cURL command again.
- \_\_\_ 13. This response is the AddressNotFoundException. Refresh the log window to see the new entries.

- \_\_\_ 14. Scan down until you see the new transaction. The request message processes successfully as before. However, in this case, it is an incorrect state that the web service detected, and it returned a fault.

12:37:31	mpgw	info	2266271	response	[REDACTED]	0x80e000b4	mpgw (EastAddressSearch): rule (EastAddressResponse): selected via match 'MatchAll' from processing policy 'EastAddressSearch'
12:37:31	mpgw	debug	2266271		[REDACTED]	0x81000171	Matching (MatchAll): Match: Received URL [/EastAddress/services/AddressSearch] matches rule '*'
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e0012a	mpgw (EastAddressSearch): Selecting Backside Processing Rule Based on URL: /EastAddress/services/AddressSearch
12:37:31	mpgw	info	2266271		[REDACTED]	0x80e0015b	mpgw (EastAddressSearch): HTTP response code 500 for 'http://WSserver:9080/EastAddress/services/AddressSearch'
12:37:31	mpgw	info	2266271		[REDACTED]	0x80e0012d	mpgw (EastAddressSearch): Using Backside Server: http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	memory-report	debug	2266271		[REDACTED]	0x80e0068e	mpgw (EastAddressSearch): Request Finished: memory used 187568
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e00159	mpgw (EastAddressSearch): Outbound HTTP on new TCP session using HTTP/1.1 to http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Compression Policy: Off, URL: /EastAddress/services/AddressSearch
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Header-Retention Policy:MQMD = OFF. MQMD Header = (NULL), URL: http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Header-Retention Policy: Range = OFF. Range Header = (NULL), URL: http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Header-Retention Policy: Accept-Encoding = OFF. Accept-Encoding Header = (NULL), URL: http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	mpgw	debug	2266271		[REDACTED]	0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Header-Retention Policy: TE = OFF. TE Header = (NULL), URL: http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	network	debug	2266271		[REDACTED]	0x80e003ca	xmImgr (default): Attempting TCP connect to WSserver

- \_\_\_ 15. Edit `findByLocation.xml` and change the state back to: NC  
 Also, make the XML invalid (change an element name) so that the schema validation fails.
- \_\_\_ 16. Run the cURL command again. Because of the invalid XML, you get an Internal Error fault.
- \_\_\_ 17. Refresh the log. Locate your transaction and the error entries.

Because the received message has a `<stateBad>` element, rather than the expected `<state>`, a schema validation error is thrown. This error is overridden as a dynamic execution error. The fault is generated.

- \_\_\_ 18. Edit `findByLocation.xml` and reverse the change that caused the schema validation error. You now have a valid XML file that specifies NC.
- \_\_\_ 19. Now experiment with the probe. Click **Control Panel > MPGW > EastAddressSearch**.
- \_\_\_ 20. Click **Show Probe** at the top of the Configure MPGW page. The Transaction List for the probe is now shown.
- \_\_\_ 21. Click **enable probe**, and **Close**.

- \_\_\_ 22. Send the cURL command again. Since you repaired the XML message, you receive a list of messages.
- \_\_\_ 23. Go back to the Transaction List, and click **Refresh**.
- \_\_\_ 24. Click the plus sign (+) icon next to the magnifying glass to expand the entry for this message. Since it is a successful message, you see both a request and a response entry.

<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	
<input type="checkbox"/>	1238352	request	http://9.10.10.37:6887/EastAddress/services/AddressSearch	http://9.10.10.45:9080/EastAddress/services/AddressSearch	EastAddressSearch_r	
<input type="checkbox"/>	1238352	response	http://9.10.10.37:6887/EastAddress/services/AddressSearch	http://9.10.10.45:9080/EastAddress/services/AddressSearch	EastAddressSearch_r	

- \_\_\_ 25. Examine the processing of the request and response messages in the probe.
  - \_\_\_ a. Click the magnifying glass next to the request message. A probe rule window opens.
  - \_\_\_ b. The actions at the top are the same as the ones defined in the processing rule. Since the rule ran successfully, all of the actions in the rule are now shown.

#### Input Context 'INPUT' of Step 1



- \_\_\_ c. Note the square brackets around the first magnifying glass. The trace entries refer to the items indicated in the square brackets. In this case, the trace is for the message as it was first presented to the rule.



#### Note

The **Content** tab presents the contents of the specified context. The text pane is XML-formatted. If the contents are not in XML, you can click **Show unformatted** to get a standard text window.

- \_\_\_ d. Before moving on, note the namespace for q0:  
http://address.training.ibm.com
- \_\_\_ e. Click the other tabs to examine their contents. Many are empty in this situation.

To move forward in the trace, you can either click **Next** in the upper right of the window, or click the next magnifying glass.

- \_\_\_ f. For the trace that follows the transform, the **Content** tab presents the **dpvar\_1** context. Recall that the output context for the transform was **dpvar\_1**. Also, notice that the result of the transform is evident in this tab: the namespace is now `http://east.address.training.ibm.com`.
- \_\_\_ g. Examine the traces of the rest of the actions. When you are finished, close this window.
- \_\_\_ h. Back in the Transaction List, click the response message.
- \_\_\_ i. In the probe rule window, the contents of the INPUT context are the results of the request message.
- \_\_\_ j. As before, examine the rest of the trace as you want. Close the window when you are finished.
- \_\_\_ 26. Examine the probe after sending an invalid message.
- \_\_\_ a. Again, edit `findByLocation.xml` to make the XML invalid (for example, change an element name), so that the schema validation fails.
- \_\_\_ b. Run the cURL command. You get an Internal Error response.
- \_\_\_ c. In the Transaction List, click **Refresh**.
- \_\_\_ d. A new entry is now shown. It is a single message and is displayed in red to indicate an error.

view	trans#	type	inbound-url	outbound-url	rule
[+]	1238352	request	<code>http://[REDACTED]:6887/EastAddress/services/AddressSearch</code>	<code>http://[REDACTED]:9080/EastAddress/services/AddressSearch</code>	EastAddressSearch
[+]	1241104	request	<code>http://[REDACTED]:6887/EastAddress/services/AddressSearch</code>	<code>http://[REDACTED]:9080/EastAddress/services/AddressSearch</code>	EastAddressSearch

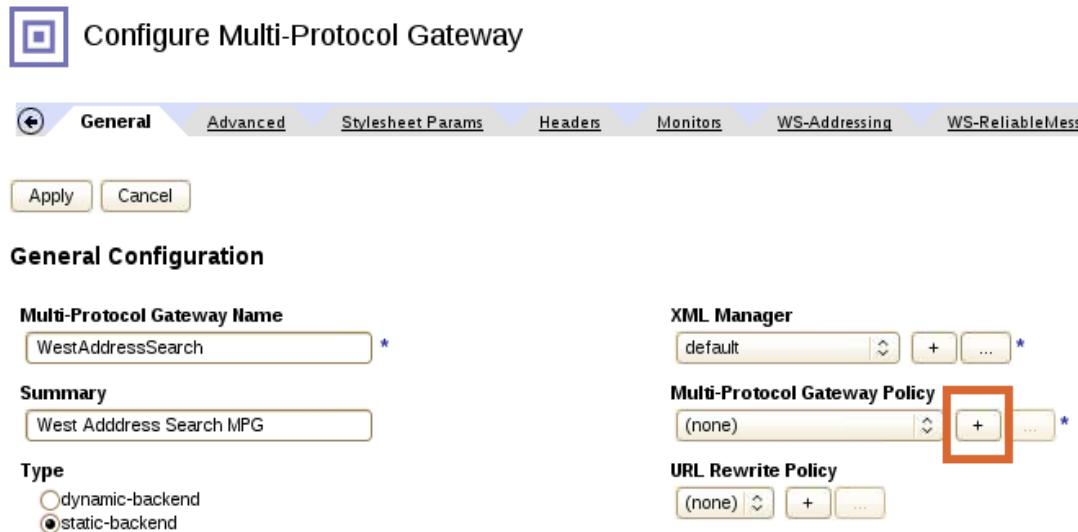
- \_\_\_ e. Click the magnifying glass next to the new request.
- \_\_\_ f. In the probe rule window, notice that the list of actions is shorter. It displays the actions that were processed.
- \_\_\_ g. Click the magnifying glass, which is the result of the validation.
- \_\_\_ h. The text at the top of the window indicates that the transaction was ended. It also lists the error message. You can see the erroneous element in the **Content** tab.
- \_\_\_ i. Scroll the tab bar to the right, and click **Service Variables**. Many of the DataPower service variables that were active during the transaction are shown.
- `var://service/error-subcode` indicates the original error that was detected. In this case, it is the schema validation error.
  - `var://service/error-code` is the general error code. In this case, it is a dynamic execution error.
  - `var://service/error-message` is the detailed error condition.

- `var://service/formatted-error-message` is what is returned to the client.
- \_\_\_ j. When you are finished with the trace, close the window.
- \_\_\_ 27. When you are finished with the probe, click **Disable Probe** in the Transaction List, and click **Close**.
- \_\_\_ 28. Close the Transaction List.
- \_\_\_ 29. Clean up `findByLocation.xml` to make it a valid XML message again.
- \_\_\_ 30. You now have some basic experience in using the system log and the probe to debug. If you have errors later in this exercise, you can use these same techniques to resolve your problems on your own.

## 4.2. Create a multi-protocol gateway for WestAddressSearch

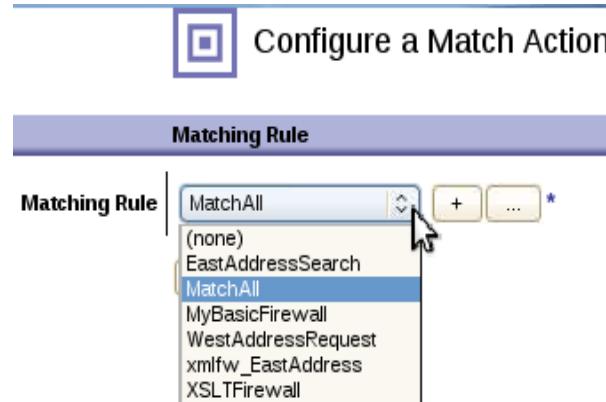
In this step, you create an MPGW based on a WSDL for the West Address Search web service. The steps are the same as the previous section, except that you use the `WestAddressSearch.wsdl` file.

- \_\_\_ 1. In the IBM WebSphere DataPower SOA Appliance WebGUI, click **Control Panel > Multi-Protocol Gateway**.
- \_\_\_ 2. From the Configure Multi-Protocol Gateway screen, click **Add**.
- \_\_\_ 3. Enter `WestAddressSearch` as the name of the new gateway. You can optionally enter a description for the gateway in the summary field, such as `West Address Search MPG`.
- \_\_\_ 4. In the **XML Manager** field, leave the **default** XML manager selected.
- \_\_\_ 5. In the **Multi-Protocol Gateway Policy** field, click **+** (new) to create a multi-protocol gateway policy.

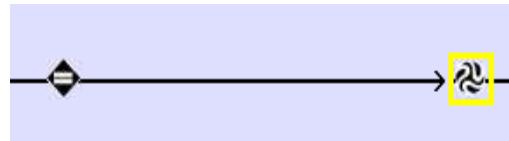


- \_\_\_ 6. Enter `WestAddressSearch` as the processing policy name.
- \_\_\_ 7. Configure a message processing rule to forward any client request message to the back-end service.
  - \_\_\_ a. In the processing rule editor for `WestAddressSearch`, click **New Rule** and rename the rule to: `WestAddressRequest`
  - \_\_\_ b. Set the processing rule direction to **Client to Server**.
  - \_\_\_ c. Double-click the **Match** action icon to configure it.

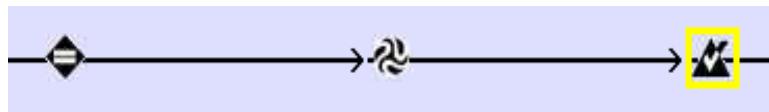
- \_\_\_ d. Click the list box arrows and select the **MatchAll** matching rule.



- \_\_\_ e. Click **Done** to use the MatchAll matching rule.
- \_\_\_ 8. Add a **Transform** action that remaps the namespace of the incoming request to the expected namespace.
- \_\_\_ a. In the rule configuration area, drag a **Transform** action after the **Match** action.



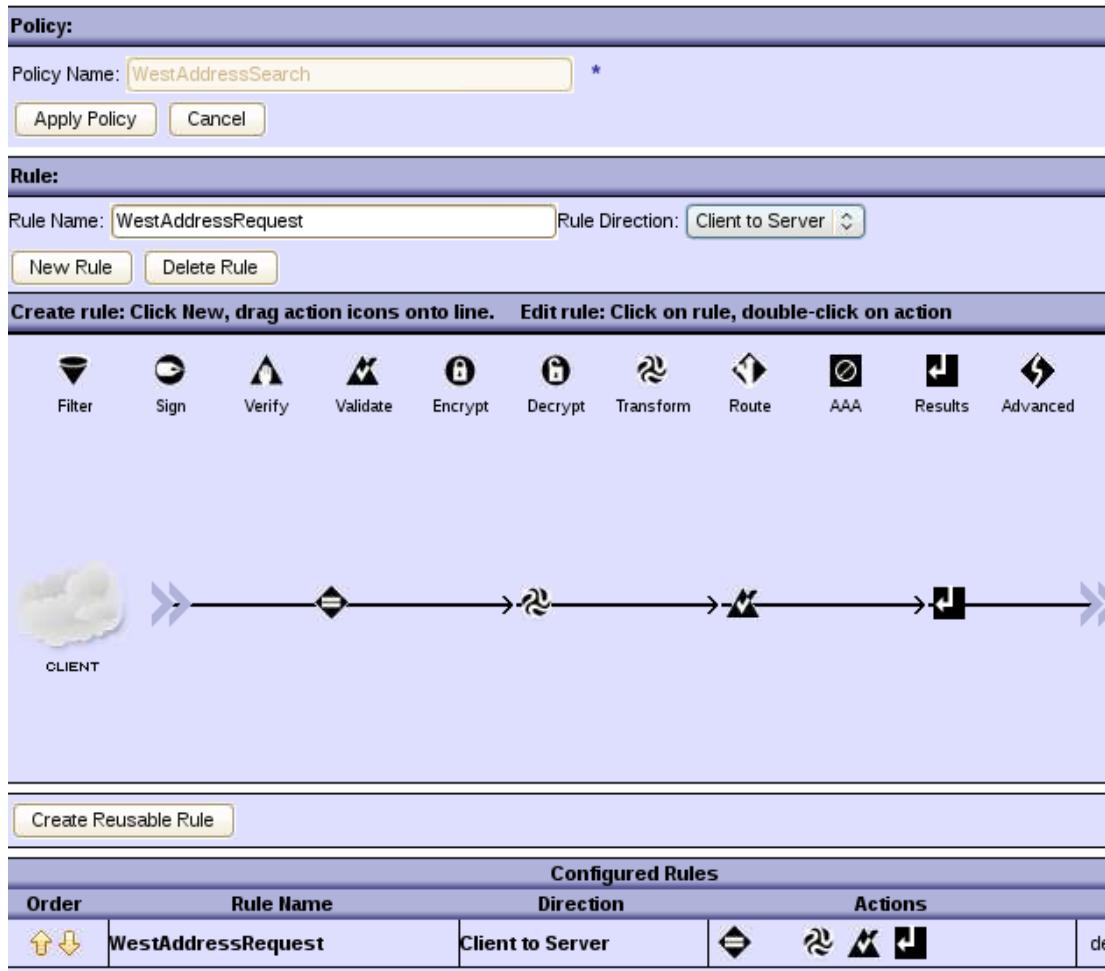
- \_\_\_ b. Double-click the **Transform** action to open the **Transform** action configuration window.
- \_\_\_ c. Verify that **Use XSLT specified in this action** is selected.
- \_\_\_ d. Click **Upload** to upload a style sheet that is called  
`<lab_files>/AdvMPGW/Address-WestRenameNamespace.xsl` that copies the input document but changes the application namespace to  
`http://West.address.training.ibm.com` for all incoming messages.
- \_\_\_ e. Click **Continue**.
- \_\_\_ f. After uploading the style sheet, make sure that the Transform configuration window contains the values.
- \_\_\_ 9. Click **Done**.
- \_\_\_ 10. Add a **Validate** action that validates the incoming WSDL request.
- \_\_\_ a. In the rule configuration area, drag a **Validate** action after the **Transform** action.



- \_\_\_ b. Double-click the **Validate** action to open the **Validate** action configuration window.

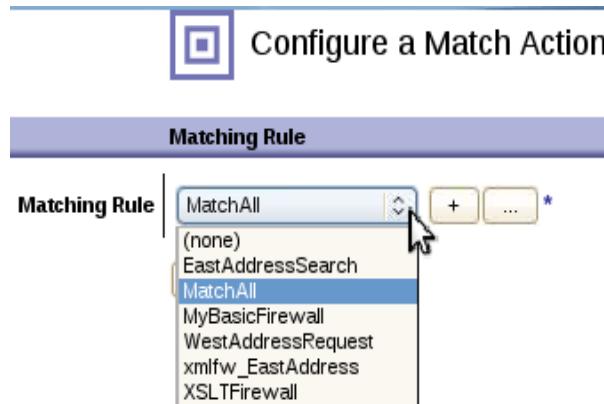
- \_\_\_ c. Verify that **Validate Document via a WSDL URL** is selected.
  - \_\_\_ d. Click **Upload** to upload a wsdl that is called  
`<lab_files>\AdvMPGW\WestAddressSearch.wsdl` that copies the input document.
  - \_\_\_ e. Click **Continue**.
  - \_\_\_ f. After uploading the wsdl, make sure that the Validate configuration window contains the values.
- \_\_\_ 11. Click **Done**.
- \_\_\_ 12. Add a **Results** action to the `WestAddressRequest` document processing rule to forward the results to the back-end service.
- \_\_\_ a. In the processing rule editor for `WestAddressRequest`, drag a **Results** action after the **Validate** action.
  - \_\_\_ b. Double-click the **Results** action to configure it.
  - \_\_\_ c. Confirm that both the **Input** and **Output** parameters are set to **(auto)**. It is also acceptable for the output parameter to be **OUTPUT**.
  - \_\_\_ d. Click **Done** to save the **Results** action settings.
  - \_\_\_ e. Click **Apply Policy** to save the changes.

- \_\_\_ f. Confirm that the new document processing rule is now shown in the list of configured rules.



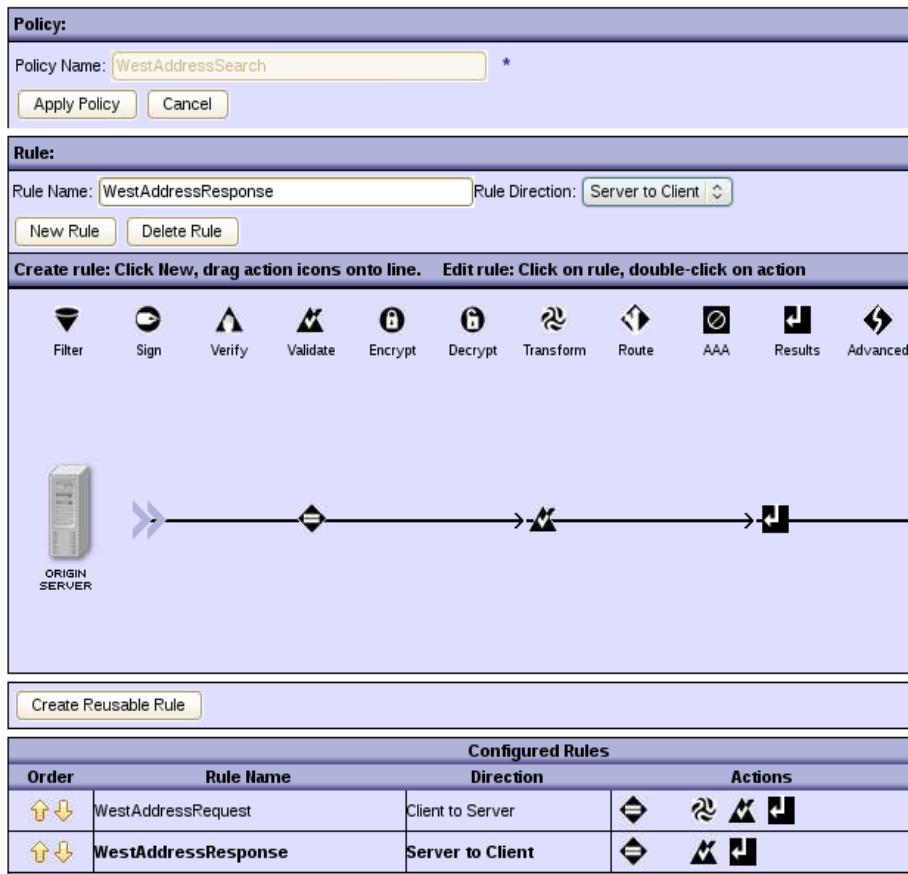
- \_\_\_ 13. Create a second document processing rule named `WestAddressResponse`. The response rule returns the message from the back side server through the DataPower appliance back to the client.
- In the processing rule editor for AddressSearchMPGWPOLICY, click **New Rule** and rename the rule to: `WestAddressResponse`
  - Set the processing rule direction to **Server to Client**.
  - Double-click the **Match** action icon to configure it.

- \_\_\_ d. Click the list box arrows and select the **MatchAll** matching rule.



- \_\_\_ e. Click **Done** to use the **MatchAll** matching rule.
- \_\_\_ 14. Add a **Validate** action that validates the incoming WSDL request.
- \_\_\_ a. In the rule configuration area, drag a **Validate** action after the **Match** action.
- 
- \_\_\_ b. Double-click the **Validate** action to open the **Validate** action configuration window.
- \_\_\_ c. Verify that **Validate Document via a WSDL URL** is selected.
- \_\_\_ d. Select the `WestAddressSearch.wsdl` in the local directory on the appliance by clicking the list box arrows that are located in the WSDL URL section.
- \_\_\_ 15. Click **Done**.
- \_\_\_ 16. Add a **Results** action to the `WestAddressResponse` document processing rule to forward the results to the back side service.
- \_\_\_ a. In the processing rule editor for `WestAddressResponse`, drag a **Results** action after the **Validate** action.
- \_\_\_ b. Double-click the **Results** action to configure it.
- \_\_\_ c. Confirm that both the **Input** and **Output** parameters are set to **(auto)**. It is also acceptable for the output parameter to be **OUTPUT**.
- \_\_\_ d. Click **Done** to save the **Results** action settings.
- \_\_\_ e. Click **Apply Policy** to save the changes.

- \_\_\_ f. Confirm that you see the new document processing rule in the list of configured rules.



- \_\_\_ 17. Click **Apply Policy** and then **Close Window** to close the WestAddressSearch document policy editor.
- \_\_\_ 18. Set a static back-end connection to the WestAddressSearch web service.
- \_\_\_ a. On the Configure Multi-Protocol Gateway page, select **static-backend** as the back-end connection type.
- \_\_\_ b. Set the back-end URL to:  
`http://WSserver:9080/WestAddress/services/AddressSearch`
- \_\_\_ 19. Change the **Propagate URI** setting to **off**.

## 4.3. Configure an HTTP front side protocol handler

The multi-protocol gateway is configured to forward requests to the West Address Search web service. However, the gateway does not support any incoming requests at the moment.

Create a front side protocol handler to receive client requests over an HTTP connection.

- \_\_\_ 1. Create an HTTP front side protocol handler named `WestAddressSearch`.
  - \_\_\_ a. In the Front Side Protocol section, click **+** (new).
  - \_\_\_ b. Select **HTTP Front Side Handler** from the list.
  - \_\_\_ c. Configure the new HTTP front side handler with the following values. Leave all other settings to the default values.
    - **Name:** `WestAddressSearch`
    - **Local IP address:** `<dp_public_ip>`
    - **Port Number:** `<mpgw_west_port>`, as assigned by the instructor
  - \_\_\_ d. Click **Apply** to save the changes that are made to the HTTP front side handler.
- \_\_\_ 2. Apply the multi-protocol gateway.
  - \_\_\_ a. Click **Apply** to save the changes that are made to the multi-protocol gateway.
  - \_\_\_ b. Verify that the multi-protocol gateway status is **up**.
  - \_\_\_ c. Click **Save Config**.
  - \_\_\_ d. The multi-protocol gateway `WestAddressSearch` is now ready for testing.

### **Section 1: WestAddressSearch MPGW testing**

Test the MPGW configuration by opening a command prompt window and running cURL. The following steps ensure that the back-end web service is 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. Open a command prompt and go to the `<lab_files>/AdvMPGW` directory.
- \_\_\_ 2. Edit the `findByLocaiton.xml` file, changing the state value from NC to CA. Since NC is not on the West Coast, the WestAddressSearch web service is not able to locate the state information.
  - \_\_\_ a. Run the following command to open the gedit editor.

```
gedit findByLocation.xml &
```

  - \_\_\_ b. Change the state value of NC to: CA
  - \_\_\_ c. Save the file.

- \_\_\_ 3. Run the following command:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_west_port>/WestAddressSearch
```

- \_\_\_ 4. Verify in the command prompt that an XML list of west addresses is returned.

## 4.4. Create an MPGW for content based routing

In this part, an MPGW service is configured to do content based routing. Content based routing involves determining how to route a message given the value of fields inside the message.

- 1. Create the front-end MPGW called AddressRouter, which receives all client SOAP requests and routes them based on the contents of the <state> field to either the EastAddressSearch or the WestAddressSearch MPGW.
  - a. In the Control Panel, click the **MPGW** icon.
  - b. On the Configure MPGW page, you see the list of existing MPGW services. To create one click **Add**.
  - c. Under General Configuration, enter:
    - **MPGW Name:** AddressRouter
    - **Summary:** Content based routing that uses an MPGW
    - **Type:** dynamic-backend



 A screenshot of a software interface showing the "General" tab selected in a navigation bar. Below the tabs are two buttons: "Apply" and "Cancel". The main area is titled "General Configuration" and contains several configuration fields:
 

- Multi-Protocol Gateway Name:** AddressRouter (with a required asterisk)
- Summary:** Content based routing using an MPGW
- Type:** dynamic-backend (radio button selected)
- XML Manager:** default (with a dropdown arrow, a plus sign, and an ellipsis button)
- Multi-Protocol Gateway Policy:** (none) (with a dropdown arrow, a plus sign, and an ellipsis button)
- URL Rewrite Policy:** (none) (with a dropdown arrow, a plus sign, and an ellipsis button)

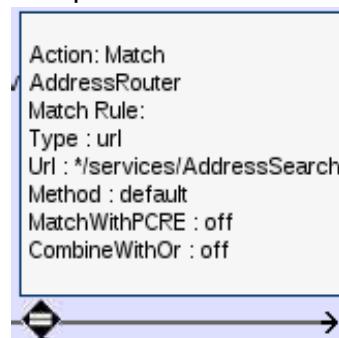


The MPGW Type **dynamic-backend** implies that the back-end service called by the MPGW is determined during the document processing policy. A **Route** action is configured in the document processing policy that determines the back-end service.

2. Create an HTTP front side protocol handler named `AddressRouter` as the device port from which this service accepts client requests.
- In the Front Side Protocol section, click **+** (new).
  - Select **HTTP Front Side Handler** from the list
  - Configure the new HTTP front side handler with the following values. Leave all other settings to the default values.
    - **Name:** AddressSearch
    - **Local IP address:** `<dp_public_ip>`
    - **Port Number:** `<mpgw_content_based_routing_port>`, as assigned by the instructor
3. Click **Apply** to save the changes that are made to the HTTP front side handler.
4. Create the AddressRouter MPGW policy.
- Click **New** (the **[+]** button) in the **Multi-protocol Gateway Policy** field to create an MPGW policy.
  - For **Policy Name**, enter: `AddressRouter`
  - Click **Apply Policy**.
  - Click **New Rule** to create a rule.
  - Create a request rule by selecting the **Rule Direction** list and selecting **Client to Server**.
  - In the rule configuration area, a **Match** action is added. Double-click the **Match** action to open the configuration window.
  - Create a matching rule that is called **AddressRouter** that matches all URLs with the string: `*/services/AddressSearch`
    - Click **New** (the **[+]** button) in the **Matching Rule** field
    - For the **Name** field, enter: `AddressRouter`
    - Click the **Matching Rule** tab
    - Click **Add**
    - **Matching Type:** URL
    - **URL Match:** `*/services/AddressSearch`

Matching Type	<input type="text" value="URL"/> *
URL Match	<input type="text" value="*/services/AddressSearch"/> *
 <input type="button" value="Apply"/> <input type="button" value="Cancel"/>	

- Click **Apply**, **Apply** again, and then **Done**.
- When the matching rule is created and completed the configuration, the **Match** action looks like this picture:



- \_\_\_ 5. Create the **Route** action in the AddressRouter MPGW policy.
- \_\_\_ a. Add a **Route** action to the request rule by dragging a **Route** action onto the rule configuration area.



- \_\_\_ b. Double-click the **Route action** to configure content based routing.
- \_\_\_ c. The Route action configuration panel is now shown. Select the **Use XPath to Select Destination** option.
- \_\_\_ d. Click **New** (the [+] button) to create an **XPath Routing Map**.
- \_\_\_ e. Enter the name for the XPath Routing Map. In this case, call it: AddressRouter
- \_\_\_ f. Select the **Rules** tab; then click **Add**.
- \_\_\_ g. Click the **XPath tool** to request a helper panel, which helps create the XPath expression.

XPath Expression  Remote Host  Remote Port  SSL	<input type="text"/> <input type="text"/> <input type="text"/> <input checked="" type="radio"/> on <input type="radio"/> off	<input type="button" value="XPath Tool"/>
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>		*

- \_\_\_ h. Upload the <lab\_files>/AdvMPGW/findByLocation.xml file. Click **Upload**, **Browse**, select the file, click **Upload**, and **Continue**.

- \_\_\_ i. In **Namespace Handling**, select the **local** option.

If the sample XML document does not show at the bottom of the panel, click **Refresh**.

- \_\_\_ j. Click the value inside the `<state>` element. That causes an XPath expression to show in the XPath text area.
- \_\_\_ k. Edit the contents to create a suitable expression to match your content based routing requirements. Depending on any previous editing of this file, the `<state>` contents might be different. For this test case, the `<state>` field is being populated with `NC`. You might easily enhance the function of the routing here by using an OR or AND XPath expression to add more states to this list.

This sample contains `<state>NC</state>`, which routes it to the EastAddressSearch MPGW.

**XPath \***

```
/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'NC']
```

**Refresh**

**Done**

**Cancel**

**Content of sample XML file.**

**Click on an element, attribute name, or attribute value to select an XPath expression.**

```

 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:q0="http://address.training.ibm.">
  <soapenv:Body>
    <q0:findByLocation>
      <city>NYC</city>
      <state>NC</state>
    </q0:findByLocation>
  </soapenv:Body>
</soapenv:Envelope>

```

- \_\_\_ l. The XPath Routing Map is now populated with the XPath expression configured in the previous step. Click **Done**.

- \_\_\_ m. Next, enter the **Remote Host Address** and **Remote Port** as <localhost\_address> and <mpgw\_east\_port> to specify the routing destination address. Click **Apply**.

XPath Expression	<input type="text" value="/*[local-name()='Envelope']/*[local-name():"/> "/>	XPath Tool
Remote Host	<input type="text" value="dp_public_ip"/>	*
Remote Port	<input type="text" value="6357"/>	*
SSL	<input checked="" type="radio"/> on <input type="radio"/> off	



### Information

You might wonder why the Remote Host Address points to the appliance rather than the web service. By pointing back to the appliance and the EastAddressSearch port, the request that is forwarded to the existing firewall service is processed according to its rules and policy. Access to the web services from the location-specific, pre-existing firewall service, and from the general routed firewall service, is allowed without duplicating any processing.

- \_\_\_ n. The configured XPath expression that forms the basis of the content based routing is now displayed on the XPath Routing Map as shown. Click **Add** to add a destination.

\*

	Remote Host	Remote Port	SSL	
local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'NC']	dp_public_ip	6357	off	Add

- \_\_\_ o. Use the previous steps as a guide to configure an additional routing destination for <state>CA</state>. In this sample of <state>CA</state>, it routes to the WestAddressSearch MPGW. Specify the **Remote Host Address** and **Remote Port** as <dp\_public\_ip> and <mpgw\_west\_port> to indicate the routing destination address.

## Rules

XPath Expression	Remote Host	Remote Port	SSL
/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'NC']	dp_public_ip	6557	off
/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'CA']	dp_public_ip	6558	off

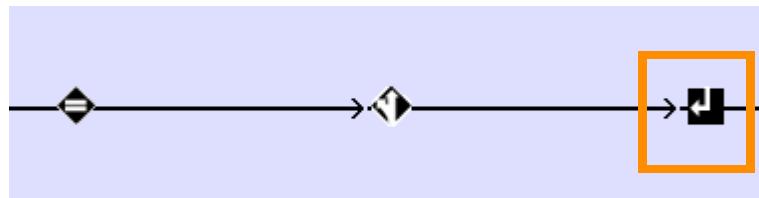
- \_\_\_ p. Click **Apply** on the Configure XPath Routing Map page.  
 \_\_\_ q. In the output context, select **(auto)** from the pick list to the right, and then type **NULL** into the output field.

The screenshot shows the 'Route (Using Stylesheet or XPath Expression)' configuration panel. It consists of several sections:

- Input:** Shows 'Input' and a dropdown menu set to '(auto)'.
- Options:**
  - Selection Method:** Radio button selected for 'Use XPath to Select Destination'.
  - XPath Routing Map:** Set to 'AddressRouter'.
  - Asynchronous:** Radio button selected for 'on'.
- Output:** Shows 'Output' and a dropdown menu set to 'NULL'.

- \_\_\_ r. Click **Done** on the “Configure Route Action” panel to complete the configuration of the content based routing function.

- \_\_\_ s. Back on the policy editor page, drag a **Results** action to the processing rule, after the **Route** action, to move the request from the INPUT context to the output context.

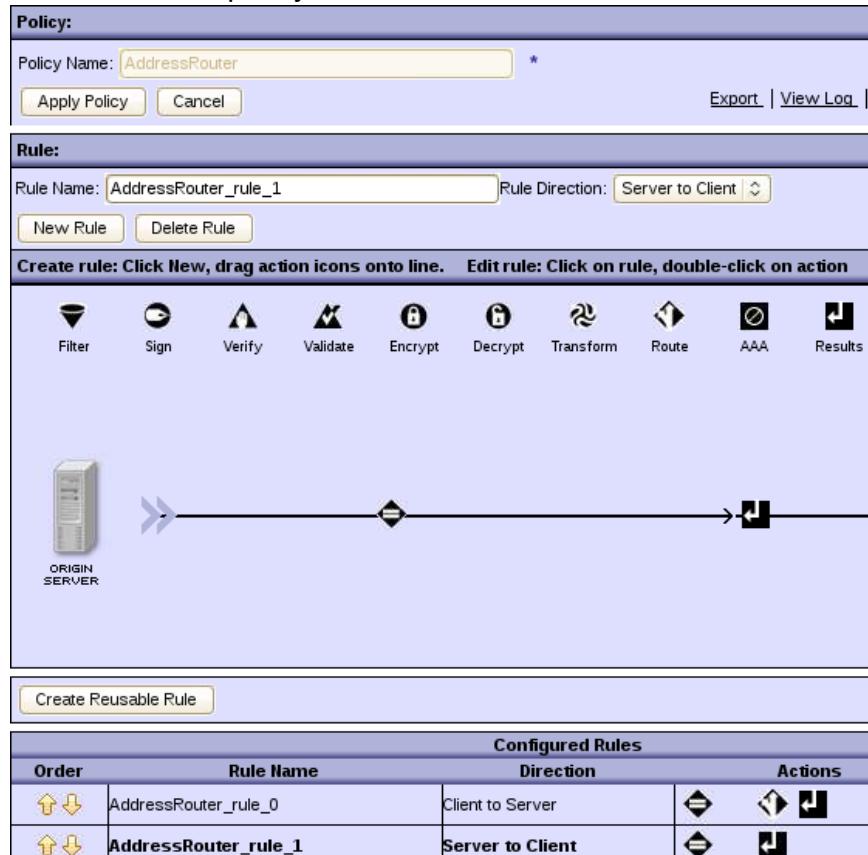


- \_\_\_ t. Double-click the **Results** action to review its settings. Click **Done**.
- \_\_\_ u. Click **Apply Policy**. Ensure that the **Route** action and **Results** action are shown on the request rule as shown in the Configured Rules section.

Configured Rules					
Order	Rule Name	Direction	Actions		
	AddressRouter_rule_0	Client to Server			

- \_\_\_ 6. Create a response rule to return the results of the **Route** action to the client.
- \_\_\_ a. In the rule configuration area, click **New Rule** to create a rule.
- \_\_\_ b. Create a response rule by selecting the **Server to Client** in the **Rule Direction** list.
- \_\_\_ c. Double-click the **Match** action icon to configure it.
- \_\_\_ d. Click the list box arrows and select the **MatchAll** matching rule.
- \_\_\_ e. Click **Done** to use the **MatchAll** matching rule.
- \_\_\_ f. Add a **Results** action after the **Match** action in the rule configuration area.

- \_\_\_ g. Click **Apply Policy** to commit the response rule, and then **Close Window** to save and close the policy editor window.



- \_\_\_ h. Change the **Propagate URI** setting to **off**.
- \_\_\_ i. Back on the Configure MPGW page, click **Apply** to save the AddressRouter MPGW configuration
- \_\_\_ j. Click **Save Config** to save your configuration to the appliance.

## 4.5. Perform end-to-end content based routing scenario testing

Use cURL to test the topology. Prebuilt SOAP messages are sent to the AddressRouter MPGW, which forwards the messages to either of two static back-end MPGWs that were defined, depending on the content of the <state> field.

- \_\_\_ 1. Open a command prompt window and run cURL. Assuming that cURL is run from the <lab\_files>/AdvMPGW> directory, run the following command:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary
@findByLocation.xml
http://<dp_public_ip>:<mpgw_content_based_routing_port>/services/Addr
essSearch
```

- \_\_\_ a. What do you observe when the value of State is set to NC (<state>NC</state>) in the findByLocation.xml prebuilt SOAP message? You get a valid response because the AddressRouter MPGW sends your request to the East Address web service.
- \_\_\_ b. What do you observe when the value of State is set to CA, <state>CA</state>, in the findByLocation.xml prebuilt SOAP message? Again, you get a valid response.
- \_\_\_ 2. **(Optional)** Modify the XPath expression to allow both <state>NY</state> and <state>NC</state> versions of findByLocation.xml to be routed to the East service. A suggestion is to use an OR, which is represented by using a vertical bar (|) in the XPath expression.
- \_\_\_ 3. **(Optional)** Try the same technique with the West states.
- \_\_\_ 4. Use the log files to do any troubleshooting that is necessary.
- \_\_\_ 5. Consider trying the multi-step probe if a deeper analysis of the symptoms is required. Recall that the probe looks at one service. If you want to follow the traffic through both services, you must enable a probe for each one.

### End of exercise

## **Exercise review and wrap-up**

In this exercise, you created three MPGWs.

Two MPGWs called EastAddressSearch and WestAddressSearch were created from WSDL files. The MPGW wizard generated a document processing policy. A Transform action was added to remap the namespace.

The third MPGW created is the AddressRouter firewall. Its policy contained a Route action that forwarded the message to the respective firewall based on the contents of the <state> field. A Set Variable action was added to remap the URI for the back-end web service.

The MPGWs were all tested with cURL by submitting a prebuilt SOAP message to the MPGWs.

The system log was analyzed for message traffic for both successful and unsuccessful messages. Additionally, the multi-step probe was used to review message traffic through a rule.

# Exercise 5. Adding error handling to a service policy

## What this exercise is about

In this exercise, you add an On Error action and an Error rule to a service policy.

## What you should be able to do

At the end of the lab, you should be able to:

- Configure a service policy with an On Error action
- Configure a service policy with an Error rule

## Introduction

The processing within a policy is expected to occasionally have errors. Policies can use **On Error** actions and error rules to deal with them. This exercise adds an error rule and an **On Error** action to provide documentation on a failure.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- Access to the `<lab_files>` directory

## Exercise instructions

### Preface

- Complete the steps in Exercise 1: Exercise setup before starting this lab.
- This exercise also depends on the previous completion of Exercise 4: Create an advanced multi-protocol gateway.
- 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
  - *<dp\_internal\_ip>*: instructor-assigned address for the DataPower appliance
  - *<mpgw\_east\_port>*: the port for the East Address service

## 5.1. Adding an error rule to the policy

For this part, you add error handling to an existing document processing policy. This error rule takes control when your validation or transformation rule has an error. The error rule contains a **Transform** action that produces an HTML document that indicates the error.

Next, you add an **On Error** action to the existing request rule. **On Error** actions specify the processing rule that is called during any errors in subsequent actions. The request rule to specify different error handling, processing rules at different steps within the request rule, is allowed. This action also applies to response rules.

This part is divided into the following sections:

- Create the error rule and add it to the policy
- Test the error rule
- Add an **On Error** action to the policy
- Test the **On Error** action
- Add an error rule and **On Error** action
- Test the new rule and action

### **Part 1: Create the error rule and add it to the policy**

- \_\_\_ 1. Examine the `custom-error.xsl` file that is used to transform error messages.
  - \_\_\_ a. In File Manager, go to: `<lab_files>/errors`
  - \_\_\_ b. Right-click and open the `custom-error.xsl` style sheet in an editor to view it. This file is used to return a custom error message. It returns an HTML response that includes a traceable transaction number, error code, error subcode, and the error message for correlation during problem discovery.



#### Note

Note the following style sheet code:

```
Transaction ID:  
<xsl:value-of  
select="dp:variable('var://service/transaction-id')"/>
```

This code uses the DataPower extension function `dp:variable` to retrieve the service variable `var://service/transaction-id`, and append it to the HTML text `Transaction ID`:

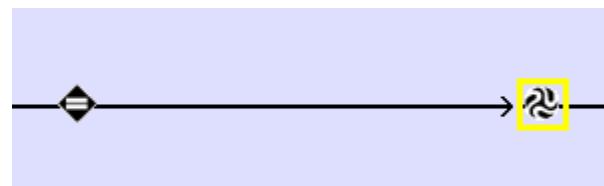
Recall the service variable from a previous probe step.

You find similar code for other service variables.

- \_\_\_ c. Close the editor.
- 2. Edit the EastAddressSearch service to add the error rule to the policy.
  - \_\_\_ a. In the DataPower WebGUI, on the Control Panel, click **MPG**.
  - \_\_\_ b. Click **EastAddressSearch** in the list.
  - \_\_\_ c. Edit the EastAddressSearch multi-protocol gateway policy by clicking the [...] (**Edit**). The current configuration of the policy displays, with its single request rule.
  - \_\_\_ d. Click **New Rule** to create a rule.
  - \_\_\_ e. Enter `EastAddressSearch_ErrorRule` in the **Rule Name** field. Usually, you use the default rule name, but in this case, you want a specific name.
  - \_\_\_ f. Select **Error** in the **Rule Direction** list.
  - \_\_\_ g. Double-click the **Matching Rule** icon.
  - \_\_\_ h. In the Configure a Match Action window, click **New** (the [+]) to create a matching rule.
  - \_\_\_ i. In the **Main** tab of the Configure Matching Rule page, enter a name of: `GenericErrorcode`
  - \_\_\_ j. Select the **Matching Rule** tab. Select **Add** to add a property.
  - \_\_\_ k. In the window to specify the new matching rule property, select a matching type of **Error Code**, and enter an error code of `0x0*` (zeros). This value matches all generic error codes.



- \_\_\_ l. Click **Apply**, **Apply** again, and then **Done**.
- \_\_\_ m. Back on the service policy editor, drag the **Transform** icon onto the rule configuration path after the matching rule.



This action is used to transform the error message that the DataPower device generates into an HTML file that includes some of the service variable values.

- \_\_\_ n. Double-click the **Transform** action.
- \_\_\_ o. Select the **Use XSLT specified in this action** radio box.
- \_\_\_ p. Click **Upload, Browse**, and go to the file:  
<LAB\_FILES>/errors/custom-error.xsl
- \_\_\_ q. Click **Open, Upload, Continue**, and then **Done**.
- \_\_\_ r. In the Configure MPGW Style Policy window, click **Apply Policy**. You now have three rules that are listed in the Configured Rules section at the bottom of the window.

Configured Rules					
	Rule Name	Direction	Actions		
1	EastAddressSearch_request	Client to Server		<a href="#">delete rule</a>	
2	EastAddressSearch_response	Server to Client		<a href="#">delete rule</a>	
3	MyEastAddressSearch_ErrorRule	Error		<a href="#">delete rule</a>	

- \_\_\_ s. Click **Close Window**.
- \_\_\_ t. Click **Apply** to save the changes.

## Part 2: Test the error rule

- \_\_\_ 1. In File Manager, go to: <lab\_files>\errors
- \_\_\_ 2. Right-click and Open `findByLocation.xml`, and verify that `<state>` is set to: NC
- \_\_\_ 3. From a command prompt, in the `<lab_files>\errors` directory, enter the command:
 

```
curl -H "Content-Type: text/xml" -H "SOAPAction: \"\""
--data-binary @findByLocation.xml
http://<dp_public_ip>:<mpgw_mpgw_east_port>/EastAddress/services/AddressSearch
```
- \_\_\_ 4. You receive the `<Address>` elements as expected.
- \_\_\_ 5. In `findByLocation.xml`, change the `<state>` to: CA
- \_\_\_ 6. Resubmit the cURL command.
- \_\_\_ 7. You get an `AddressNotFoundException` message. No error handling occurred because there was an acceptable response from the back-end server.
- \_\_\_ 8. Edit `findByLocation.xml` again and change `<state>` back to NC  
Also, change the message to make it fail schema validation (for example, change `<state>` to `<stateBad>` and be sure to change both the start and end tags.)
- \_\_\_ 9. Start the multi-step probe for EastAddressSearch.
  - \_\_\_ a. Click **Show Probe** at the top of the “Configure Multi-Protocol Gateway” page.

- \_\_\_ b. Click **Enable Probe** at the top of the transaction list page.
- \_\_\_ 10. Resubmit the cURL command.
- \_\_\_ 11. You receive an HTML result that contains the service variables that are generated from `custom-error.xsl`. The error message text gets a result that is an error message that identifies the faulty element.
- \_\_\_ 12. On the Multi-step Probe Transaction List, click **Refresh**.  
Your transaction has a plus sign (+) in front of the magnifying glass icon.
- \_\_\_ 13. Click the plus sign (+) to expand the entry.



view trans#	type	inbound-url	outbound-url	rule	client-
2464303	request	http://172.16.78.44:6957/EastAddress /services/AddressSearch	http://WSServer:9080/EastAddress /services/AddressSearch	EastAddressRequest	172.16
2464303	error	http://172.16.78.44:6957/EastAddress /services/AddressSearch	http://WSServer:9080/EastAddress /services/AddressSearch	EastAddressSearch_ErrorRule	172.16



### Information

The red color of the `Request` message processing on the `EastAddressRequest` rule identifies an error on the request rule. When the processing rule experienced an error, the processing was passed to the error rule, `EastAddressSearch_ErrorRule`. The error rule that is processed successfully is represented as the black color text.

- \_\_\_ 14. Click the magnifying glass for the **request**. Examine the probe. The details are similar to what is seen previously, but notice that the trace ends after the **Validate** action failure.
- \_\_\_ 15. Click the magnifying glass for the **error**. The probe for running the error rule is shown. The Transform action is using `custom-error.xsl`. The output context displays the HTML you received in the command prompt.
- \_\_\_ 16. Close the Probe Details window.

### Part 3: Add an **On Error** action to the policy

In this section, you add an **On Error** action in the request rule. This action sets the error handling for any subsequent actions in the rule.

This particular **On Error** action specifies the already-existing error rule as the error handler. You see some differences in the error message that is generated and the probe details.

- \_\_\_ 1. Go back to the EastAddressSearch policy editor.

- \_\_ 2. Be sure that the request rule is displayed on the rule configuration path. If not, select the request rule in the Configured Rules section.

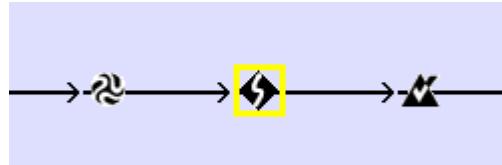
The screenshot shows the configuration interface for a service policy. At the top, there's a header with 'Rule:' and tabs for 'Rule Name' (set to 'EastAddressRequest') and 'Rule Direction' (set to 'Client to Server'). Below this are buttons for 'New Rule' and 'Delete Rule'. A status bar at the bottom says 'Create rule: Click New, drag action icons onto line.' and 'Edit rule: Click on rule, double-click on action'.

The main area displays a flow diagram from 'CLIENT' to 'ORIGIN SERVER'. The path consists of several action icons: Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, Advanced, and a trash can icon. The 'Advanced' icon is highlighted with a red box.

Below the diagram is a 'Create Reusable Rule' button. The 'Configured Rules' section contains a table:

Configured Rules			
Order	Rule Name	Direction	Actions
	EastAddressRequest	Client to Server	
	EastAddressResponse	Server to Client	
	EastAddressSearch_ErrorRule	Error	

- \_\_ 3. Drag the **Advanced** icon to the configuration path before the **Validate** action.

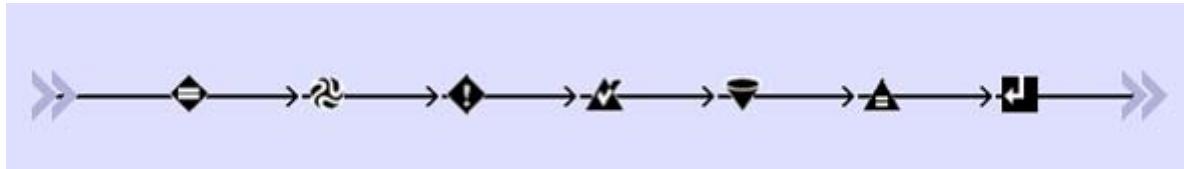


- \_\_ 4. Double-click the **Advanced** icon to open the Configure Action page.  
 \_\_ 5. Scroll down to select **On Error**, and click **Next**.  
 \_\_ 6. Set the **Error Mode** to **Cancel** because you want the rule to terminate after the error handling.  
 \_\_ 7. For the Processing Rule, use the list to select **EastAddressSearch\_ErrorRule**.  
 \_\_ 8. Leave the **Error Input** field empty. The default behavior is to have the failed action context become the input context for the Processing Rule.

- \_\_\_ 9. Leave the **Error output** field empty. The **Transform** action in the error rule sets the OUPUT context.



- \_\_\_ 10. Click **Done**.
- \_\_\_ 11. Add a **Filter** action to block the invocation of the retrieveAll operation in the web service.
- \_\_\_ a. Drag the **Filter** icon onto the rule path, following the **Validate** action.
  - \_\_\_ b. Double-click the **Filter** action, click **Upload**, and **Browse** to go to the filtering style sheet `EastAddressSearch.xsl`.
  - \_\_\_ c. Select the file, click **Open**, **Upload**, **Continue**, and then click **Done**.
  - \_\_\_ d. Confirm that the rule looks like the image:



- \_\_\_ 12. Click **Apply Policy**, and then **Close Window** on the policy editor.

- \_\_\_ 13. Click **Apply** on the Configure MPGW page.

- \_\_\_ 14. Click **Save Config**.

#### **Part 4: Test the On Error action**

- \_\_\_ 1. Run the `curl` command again on the `findByLocation.xml` file that still contains the bad element.

```
curl -H "Content-Type: text/xml" --data-binary @findByLocation.xml
http://<dp_public_ip>:<mpgw_east_port>/EastAddress/services/Address
Search
```

You receive an HTML result that contains the service variables that are generated from `custom-error.xsl`, as before.

- \_\_\_ 2. On the Multi-step Probe Transaction List, click **Refresh**.
- \_\_\_ 3. Expand your transaction as before. You see **request** and **call** probe entries.
- \_\_\_ 4. Examine the request probe.
  - \_\_\_ a. Click the magnifying glass for the **request**.

The transaction ends at the validation as before, but you now see an extra magnifying glass. Essentially, the **On Error** action is getting control.

- \_\_\_ b. Select it and examine the content tab. It is the same as the failing **Validate** action input context. You allowed this default behavior in the **On Error** action.
- \_\_\_ c. Examine the service variables. They are the result of the error processing; that is, they have the HTML response that is built by the error rule.
- \_\_\_ 5. Examine the **call** probe. Click the magnifying glass for the **call**. This probe is for running the error rule.

The **Transform** action is still using `custom-error.xsl`.

The output context displays the HTML you received in the command prompt, which contains the error message on the validation error.

WebSphere DataPower XI52						IBM
Refresh	Flush	Disable Probe	Export Capture	View Log	Send Message	Close
view	trans#	type	inbound-url	outbound-url	rule	client-ip
2519039	request		http://172.16.78.44:6957/EastAddress/services/AddressSearch	http://WSserver:9080/EastAddress/services/AddressSearch	EastAddressRequest	172.16.8.1
2519039	call		http://172.16.78.44:6957/EastAddress/services/AddressSearch	http://WSserver:9080/EastAddress/services/AddressSearch	EastAddressSearch_ErrorRule	172.16.8.1



### Information

When an error occurred on the `EastAddressRequest` rule, the processing implemented the configuration of the On Error action. Notice that the processing type of call is shown on the multi-step probe for the processing action. Before, when using the Error Rule, the processing type of `error` was called.

- \_\_\_ 6. Close the Probe Details window.
- \_\_\_ 7. On the Multi-step Probe Transaction List, click **Disable Probe** to stop the probe recording for this service.
- \_\_\_ 8. Edit the `findByLocation.xml` file to remove your changes.

## Part 5: Add another Error rule and On Error action

- \_\_\_ 1. In the policy editor, click **New Rule**.
- \_\_\_ 2. Name the new error rule: EastAddressSearch\_filter\_ErrorRule
- \_\_\_ 3. Create it the same way as the previous error rule, but in the **Transform** action, upload filter-custom-error.xsl instead.
- \_\_\_ 4. Click **Apply Policy** to commit the new error rule.

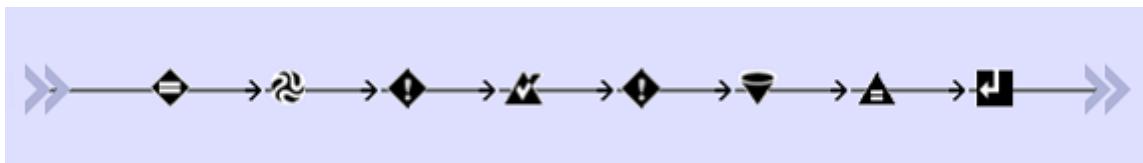
Configured Rules				
Order	Rule Name	Direction	Actions	
↑↓	EastAddressRequest	Client to Server	⊖ ↖ ↙ ↗ ↘ ↛ ↜ ↞	
↑↓	EastAddressResponse	Server to Client	⊖ ↖ ↙ ↗ ↘ ↛ ↜ ↞	
↑↓	EastAddressSearch_ErrorRule	Error	⊖ ↖ ↙ ↗ ↘ ↛ ↜ ↞	
↑↓	EastAddressSearch_filter_ErrorRule	Error	⊖ ↖ ↙ ↗ ↘ ↛ ↜ ↞	

- \_\_\_ 5. Select the request rule `EastAddressRequest` to move it to the rule configuration path.
- \_\_\_ 6. Using the **Advanced** icon, add an **On Error** action just before the **Filter** action.
- \_\_\_ 7. In the Configure On Error Action page, set the **Error Mode** to **Cancel** and leave the **Error Input** and **Error output** fields at **auto**. For the Processing Rule, select the newly created `EastAddressSearch_filter_ErrorRule`.

 **On Error**

<b>Error Mode</b>	<input type="button" value="Cancel"/>
<b>Processing Rule</b>	<input type="button" value="EastAddressSearch_filter_ErrorRule"/> <input type="button" value="+"/> <input type="button" value="..."/> <input type="button" value="Var Builder"/>
<b>Error Input</b>	<input type="text"/> (none)
<b>Error Output</b>	<input type="text"/> (none)
<b>Asynchronous</b>	<input type="radio"/> on <input checked="" type="radio"/> off
<input type="button" value="Delete"/> <input type="button" value="Done"/> <input type="button" value="Cancel"/>	

- \_\_\_ a. Confirm that the rule looks like the image:



- \_\_\_ 8. Click **Done**, and then **Apply Policy**.
- \_\_\_ 9. Close the policy editor, and then click **Apply**.

## Part 6: Send a message to test the new error-handling

1. This time, send a different message that causes the **Filter** action to reject your message:

```
curl -H "Content-Type: text/xml" --data-binary @retrieveAll.xml  
http://<dp_public_ip>:<mpgw_east_port>/EastAddress/services/  
AddressSearch
```

Recall that you did not include the **retrieveAll** operation when you specified the WSDL while creating the MPGW. The HTTP response indicates that an unsupported operation was attempted, and the message was rejected.



### Information

The second **On Error** action overrides the previous one. Therefore, this new **On Error** action directed the failing message to the second error rule, which produced a different error message for the client.

2. Click **Save Config**.

### End of exercise

## Exercise review and wrap-up

In this exercise, you examined the two ways to manage errors that occur while a policy is running: error rules, and **On Error** actions.

You also experienced the effect of adding the **On Error** action to call a different error rule within the same policy.

# Exercise 6. Creating cryptographic objects

## What this exercise is about

This exercise shows you how to create cryptographic keys by using the DataPower crypto tools. Keys can be created on the appliance or uploaded externally. You create a crypto identification credential by storing certificate-key pairs that are used in securing SSL connections. You also create a validation credential object for validating certificates. These objects are used as part of a crypto profile.

## What you should be able to do

At the end of the exercise, you should be able to:

- Generate cryptographic keys by using the WebSphere DataPower cryptographic tools
- Upload key files to the WebSphere DataPower SOA Appliance
- Create a cryptographic identification credential by using a cryptographic key object
- Validate certificates by using a validation credential object

## Introduction

The DataPower appliance supports generating certificate-key pairs on the appliance or uploading key files that contain certificate-key pairs to the appliance. You can use the `keytool` command that is shipped with a JDK to create key files and upload them to the appliance.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **Eclipse**, for the Java `keytool.exe` contained within the Eclipse subdirectory
- Access to the `<lab_files>` directory

## Exercise instructions

### Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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
  - *<dp\_internal\_ip>*: instructor-assigned DataPower appliance address
  - *<dp\_WebGUI\_port>*: administrative port for the WebGUI

## 6.1. Generate a certificate-key pair on the DataPower appliance

In this section, you create a certificate-key pair on the DataPower appliance. The certificate-key pair can be used during SSL connections. The generated certificate that contains your public key can be presented to clients. A client can create a message signed with your public key that only you can decrypt by using the private key that you generated.

- 1. Use a web browser to log on to the DataPower WebGUI:

`https://<dp_internal_ip>:<dp_WebGUI_port>`

- 2. Generate a certificate-key pair on the DataPower appliance.
  - a. In the DataPower WebGUI vertical navigation bar, select **ADMINISTRATION > Miscellaneous > Crypto tools**.



### Note

The Generate key web page is not available from the **Keys and Certs Management** icon on the DataPower WebGUI.

- \_\_\_ b. The Generate key page allows you to use the information that is entered on this page to generate a certificate-key pair. The fields from **Country Name** down to **Common Name** are part of the distinguished name. Enter the following information for the distinguished name:

- **Country Name (C)**: US
- **State or Province (ST)**: CA
- **Locality (L)**: Los Angeles
- **Organization (O)**: IBM
- **Organizational Unit (OU)**: Software group
- **Common Name (CN)**: Student

### Generate Key

**LDAP (reverse) Order of RDNs**

<input type="radio"/> on	<input checked="" type="radio"/> off
Country Name (C)	
US	
<input type="radio"/> on	<input checked="" type="radio"/> off
State or Province (ST)	
CA	
<input type="radio"/> on	<input checked="" type="radio"/> off
Locality (L)	
Los Angeles	
<input type="radio"/> on	<input checked="" type="radio"/> off
Organization (O)	
IBM	
<input type="radio"/> on	<input checked="" type="radio"/> off
Organizational Unit (OU)	
Software Group	
<input type="radio"/> on	<input checked="" type="radio"/> off
Organizational Unit 2 (OU)	
<input type="radio"/> on	<input checked="" type="radio"/> off
Organizational Unit 3 (OU)	
<input type="radio"/> on	<input checked="" type="radio"/> off
Organizational Unit 4 (OU)	
<b>Common Name (CN)</b>	Student *

- \_\_\_ c. The remaining fields are for certificate-key pair information. Enter the following information and leave the remaining fields at their default values:

- **Export Private Key**: on
- **Object Name**: StudentKeyObj

<b>Export Private Key</b>	<input checked="" type="radio"/> on	<input type="radio"/> off
<b>Generate Self-Signed Certificate</b>	<input checked="" type="radio"/> on	<input type="radio"/> off
<b>Export Self-Signed Certificate</b>	<input checked="" type="radio"/> on	<input type="radio"/> off
<b>Generate Key and Certificate Objects</b>	<input checked="" type="radio"/> on	<input type="radio"/> off
<b>Object Name</b>	StudentKeyObj	
<b>Using Existing Key Object</b>		
<b>Generate Key</b>		

**Note**

If you do not select **on** for export private key or export self-signed certificate, then you cannot download the keys. They are placed in the `cert:///` directory.

- \_\_\_ d. Click **Generate Key**.
- \_\_\_ e. Click **Confirm** to proceed with generating the private key and self-signed certificate, and then click **Close**.

**Information**

This action generates a private key and self-signed certificate and places them in the DataPower appliance `temporary:///` directory. Two objects, each with the name `StudentKeyObj`, are created for both the private key and self-signed certificate.

In addition to generating the private key and self-signed certificate, a **certificate signing request (CSR)** is also generated. A CSR is a request message sent to a certificate authority (CA) to create a digital certificate. A CSR consists of identifying information (common name, for example) and your public key. The request is signed with your private key, but the actual private key is not included in the request. The CA issues a signed digital certificate that replaces your self-signed certificate.

- \_\_\_ 3. Verify the generation of the private key and certificate objects called `StudentKeyObj`.
  - \_\_\_ a. Click the **Control Panel** link at the top of the WebGUI.
  - \_\_\_ b. Click the **Keys & Certs Management** icon.
  - \_\_\_ c. Click the **Keys** link.
  - \_\_\_ d. Verify that you see the `StudentKeyObj` referencing the generated private key file.

Name	Status	Op-State	Logs	File Name
<code>StudentKeyObj</code>	<code>new</code>	<code>up</code>		<code>cert:///StudentKeyObj-privkey.pem</code>

- \_\_\_ e. Click the web browser back, or **Control Panel > Keys and Certs Management** again.
- \_\_\_ f. In the Keys and Certificates Management page, in the **Basics** section, click the **Certificates** link.

- \_\_\_ g. Verify that you see the `StudentKeyObj` referencing the generated self-signed certificate file.

Name	Status	Op-State	Logs	File Name
StudentKeyObj	new	up		cert:///StudentKeyObj-sscert.pem

- \_\_\_ h. Click the `StudentKeyObj` certificate.  
 \_\_\_ i. Set `Ignore Expiration Dates` to **on**.

Crypto Certificate: StudentKeyObj [up]

Administrative State

enabled  disabled

File Name

cert:///   
StudentKeyObj-sscert.pem

Password

Password Alias

on  off

Ignore Expiration Dates

on  off

- \_\_\_ j. Click **Apply**.

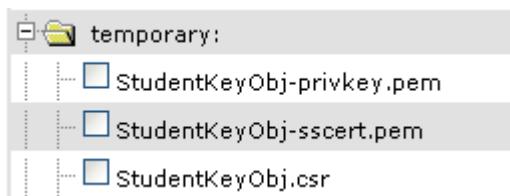


#### Note

The two objects that are generated by the DataPower appliance with the same name are separate objects. They have different file names in the `cert:` directory.

- \_\_\_ 4. View the private key and self-signed certificate that were exported to the temporary directory.  
 \_\_\_ a. Click **Control Panel > File Management**.

- \_\_\_ b. Expand the `temporary:` directory. You see the exported private key, the self-signed certificate, and the CSR:



The `StudentKeyObj-privkey.pem` is the private key file. The `StudentKeyObj-sscert.pem` is the self-signed certificate file.

## 6.2. Creating cryptographic objects

- \_\_\_ 1. Create an identification credential object.

 **Information**

An identification credential object is used to reference a certificate-key pair during an SSL connection. You create an identification credential that references the certificate-key objects that are created in the previous steps. The identification credential object is used to identify yourself during an SSL connection, and to participate in the SSL handshake.

- \_\_\_ a. Click the **Control Panel** link at the top of the WebGUI.
- \_\_\_ b. Click the **Keys and Certs Management** icon.
- \_\_\_ c. Under **SSL**, click the **Identification Credentials** link.
- \_\_\_ d. On the Configure Crypto Identification Credentials page, click **Add**.
- \_\_\_ e. On the next page, enter the following information:
- **Name**: StudentIdCred
  - **Crypto Key**: StudentKeyObj
  - **Certificate**: StudentKeyObj

Crypto Identification Credentials

Name	<input type="text" value="StudentIdCred"/> *
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Crypto Key	<input type="text" value="StudentKeyObj"/> + ... *
Certificate	<input type="text" value="StudentKeyObj"/> + ... *
Intermediate CA Certificate	(empty) <input type="button" value="Add"/> + ...

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

This action creates an identification credential object. If a third-party CA signed the certificate, you can specify CA certificates in the **Intermediate CA Certificate** field.



### Information

A **validation credential object** is used to validate the authenticity of certificates and digital signatures that are presented to a service.

- \_\_\_ 2. Click the **Control Panel** link at the top of the WebGUI.
- \_\_\_ 3. Click the **Keys and Certs Management** icon.
- \_\_\_ 4. Under **SSL**, click the **Validation Credentials** link.
- \_\_\_ 5. On the “Configure Crypto Validation Credentials” page, click **Add**.
  - \_\_\_ a. Enter the following information (leave the remaining fields at their default values):
    - **Name:** StudentKeyValCred
    - **Certificates:** StudentKeyObj (select it from the list)
- \_\_\_ 6. Click **Add** to add the StudentKeyObj certificate.

Crypto Validation Credentials

<b>Name</b>	<input type="text" value="StudentKeyValCred"/> *
<hr/>	
<b>Administrative State</b>	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
<hr/>	
<b>Certificates</b>	<div style="border: 1px solid #ccc; padding: 5px; width: 200px; height: 50px; margin-bottom: 5px;"> <input type="text" value="StudentKeyObj"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> </div> <div style="border: 1px solid #ccc; padding: 5px; width: 200px; height: 50px; position: relative;"> <input style="width: 100%; height: 100%; position: absolute; top: 0; left: 0;" type="button" value="Add"/> </div>
<hr/>	
<b>Certificate Validation Mode</b>	<input type="text" value="Match exact certificate or immediate issuer"/>
<hr/>	
<b>Use CRL</b>	<input checked="" type="radio"/> on <input type="radio"/> off

- \_\_\_ 7. Click **Apply** to save the **crypto validation credential**. This validation credential validates the StudentKeyObj certificate, if presented.
- \_\_\_ 8. Create a server **Crypto Profile** to use in an SSL communication.



## Information

A **crypto profile** is used to identify certificate-key pairs in an SSL connection. It can also validate presented certificates by using a validation credential object.

- \_\_ a. Click the **Control Panel > Keys and Certs Management** icon.
- \_\_ b. Under **SSL**, click the **Crypto Profile** link.
- \_\_ c. On the Configure Crypto Profile page, click **Add** to create a crypto profile.
- \_\_ d. Enter the following information (leave the remaining fields at their default values):
  - **Name:** StudentServerCP
  - **Identification Credentials:** StudentIdCred (select it from the list)
  - **Disable SSL version 2:** Clear this option

Crypto Profile

<b>Name</b> <input type="text" value="StudentServerCP"/>	<b>Administrative State</b> <input checked="" type="radio"/> enabled <input type="radio"/> disabled
<b>Identification Credentials</b> <input type="text" value="StudentIdCred"/>	
<b>Validation Credentials</b> <input type="text" value="(none)"/>	
<b>Ciphers</b> <input type="text" value="HIGH:MEDIUM:!aNULL!:eNULL:@STRENGT"/>	
<b>Options</b> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> Enable default settings</li> <li><input type="checkbox"/> Disable SSL version 2</li> <li><input type="checkbox"/> Disable SSL version 3</li> <li><input type="checkbox"/> Disable TLS version 1</li> <li><input type="checkbox"/> Permit insecure SSL renegotiation to a legacy SSL client</li> </ul>	
<b>Send Client CA List</b> <input type="radio"/> on <input checked="" type="radio"/> off	

- \_\_ e. Click **Apply**.

This crypto profile specifies an identification credential object with a certificate-key pair. If the DataPower appliance requests a client certificate during

SSL authentication, then specify a validation credential object that validates the client certificate.

- \_\_\_ 9. Create a client crypto profile to use in an SSL communication.
- \_\_\_ a. On the Configure Crypto Profile page, click **Add** to create another crypto profile.
- \_\_\_ b. Enter the following information (leave the remaining fields at their default values):
- **Name:** StudentClientCP
  - **Validation Credentials:** StudentKeyValCred
  - **Disabled SSL version 2:** Clear this option

Crypto Profile

<b>Name</b>	<input type="text" value="StudentClientCP"/> *
<hr/>	
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Identification Credentials	<input type="button" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>
Validation Credentials	<input type="button" value="StudentKeyValCred"/> <input type="button" value="+"/> <input type="button" value="..."/>
Ciphers	HIGH:MEDIUM:aNULL:eNULL:@STRENGTH
Options	<input checked="" type="checkbox"/> Enable default settings <input type="checkbox"/> Disable SSL version 2 <input type="checkbox"/> Disable SSL version 3

- \_\_\_ c. Click **Apply**.

The SSL client uses this crypto profile to validate a presented certificate. It uses the `StudentKeyValCred` validation credential object.

## 6.3. Import a certificate-key pair onto the DataPower appliance

In this section, you use the Java `keytool` command to generate a certificate-key pair and upload the key files to the DataPower appliance.

- 1. Use the Java `keytool` command to generate a keystore with a certificate-key pair. A shell script (equivalent to a Windows batch file) is created to specify the parameters that are required to generate the certificate-key pair by using the Java `keytool` command.

- a. Open a terminal window.
- b. Switch to the crypto directory for the lab files.  
`cd <lab_files>/crypto`
- c. Run the `keytool.sh` shell script.  
`./keytool.sh`

The Java `keytool` command generates a keystore named `dpedu.p12` with the password `dpadmin` and stored in the **pkcs12** format. The RSA key algorithm is used to generate a certificate-key pair. The digital certificate contains the distinguished name `cn=dpedu, o=ibm, c=ca`.

You get a message that the program is generating keys and a certificate.

This file is generated in the same directory as the `keytool` command.



### Note

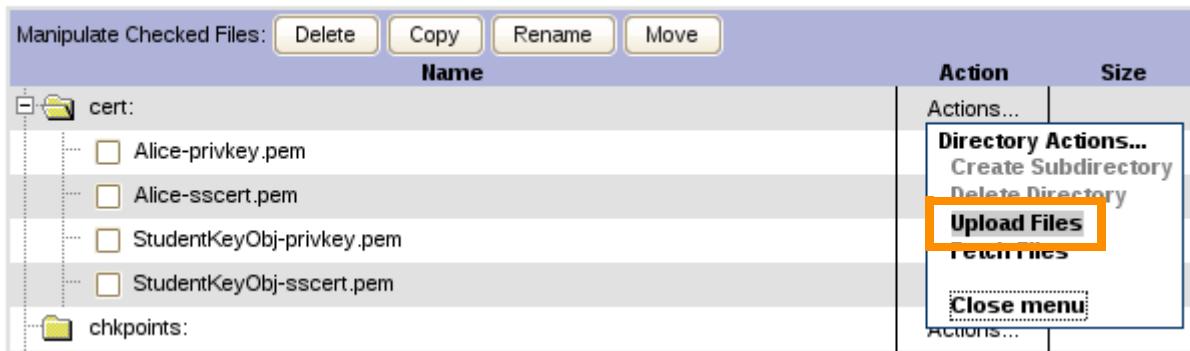
If you prefer not to use the provided shell script, and generate the key manually, proceed as follows:

Type the following command to generate a keystore. You can also copy this command from: `<lab_files>/crypto/keytool.txt`  
(If the pasted keytool command does not work, you might be required to type it directly.)

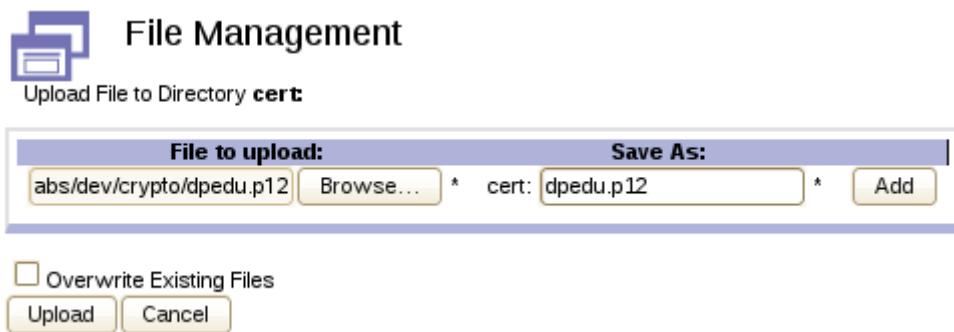
```
keytool -genkey -v -alias dpedu -keypass dpadmin -keystore dpedu.p12  
-storepass dpadmin -storetype "pkcs12" -dname "cn=dpedu, o=ibm, c=ca"  
-keyalg "RSA"
```

- 2. Upload the generated keystore to the DataPower appliance.
  - a. Click the **Control Panel** link at the top of the WebGUI.
  - b. Click the **File Management** icon.

- \_\_\_ c. Select the `cert:` directory and select the **Actions** link. In the subsequent list, select **Upload Files**.



- \_\_\_ d. In the "File Management" window click **Browse** to go to your generated key **dpedu.p12** (in `<lab_files>/crypto`).



- \_\_\_ e. Click **Upload**, and then click **Continue**.
- \_\_\_ 3. Generate a key object from the uploaded keystore.
- \_\_\_ a. Click the **Control Panel** link at the top of the WebGUI.
- \_\_\_ b. Click the **Keys and Certs Management** icon.
- \_\_\_ c. Under **Basics**, click the **Keys** link.
- \_\_\_ d. On the Configure Crypto Key page, click **Add** to create a crypto key object.
- \_\_\_ e. On the next page, enter the name: **DPEduKey**
- \_\_\_ f. In the file name field, select **dpedu.p12** (your uploaded key) in the `cert:` directory.



### Note

You can also upload key files on this page by clicking **Upload**.

- \_\_\_ g. Enter the key password `dpedu` twice. This password is the same one used when generating the key.

## Crypto Key

---

Name	DPEduKey
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
File Name	cert:/// <input type="button" value="Up"/>
	dpedu.p12 <input type="button" value="Up"/>
Password	***** *****
Password Alias	<input type="radio"/> on <input checked="" type="radio"/> off

- \_\_\_ h. Click **Apply**. Make sure that the operation state of your crypto key is **up**.

**Note**

If you get an error after creating the **dpedu.p12** cryptographic key object, the most common cause is the mismatch of time on the workstation and the DataPower appliance.

Check the timestamp on the workstation where the key is generated, and the time on the DataPower appliance (**STATUS > Main > Date and Time**). You must be in the default domain to complete the time lookup. If the time values are different, then an error occurs when you save the cryptographic key configuration. This result is why **on** is chosen in the **Ignore Expiration Dates** field to ignore timestamps.

- 
- \_\_\_ 4. Generate a certificate object from the uploaded keystore.
- \_\_\_ a. Click the **Control Panel** link at the top of the WebGUI.
- \_\_\_ b. Click the **Keys & Certs Management** icon.
- \_\_\_ c. Under **Basics**, click the **Certificates** link.

- \_\_\_ d. On the Configure Crypto Certificate page, scroll down and click **Add** to create a certificate object.
- \_\_\_ e. On the next page, enter the name: DPEduCert
- \_\_\_ f. In the file name field, select **dpedu.p12** (your uploaded keystore) in the cert: directory.
- \_\_\_ g. Enter the password **dpadmin** twice.
- \_\_\_ h. Set Ignore Expiration Dates to **on**.
- \_\_\_ i. Click **Apply**. You see the status of your DPEduCert as **up**.

### Crypto Certificate

**Apply**   **Cancel**

---

<b>Name</b>	<input type="text" value="DPEduCert"/> *
<hr/>	
<b>Administrative State</b>	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
<b>File Name</b>	<input type="text" value="cert:///"/> <input type="text" value="dpedu.p12"/> <input type="button" value="Details..."/>
<b>Password</b>	<input type="password"/> <input type="password"/>
<b>Password Alias</b>	<input type="radio"/> on <input checked="" type="radio"/> off
<b>Ignore Expiration Dates</b>	<input checked="" type="checkbox"/> on <input type="checkbox"/> off

- \_\_\_ j. Click **Save Config** to copy these settings to the startup configuration.

Optionally, you can use these cryptographic key and certificate objects to create an identification credential object. You can then use this identification credential object to create a crypto profile object. This crypto profile can then be used during SSL communication.

## End of exercise

## Exercise review and wrap-up

In this exercise, you generated key certificate objects on the DataPower appliance and also uploaded them from a key file. The key and certificate objects are used to create an identification credential object, referenced by the crypto profile. A crypto profile is used during SSL communication.

# Exercise 7. Using SSL to secure connections

## What this exercise is about

This exercise shows you how to configure SSL connections for DataPower services when using the WebGUI.

## What you should be able to do

At the end of the exercise, you should be able to:

- Create an SSL proxy profile that accepts an SSL connection request from a client
- Create an SSL proxy profile that initiates an SSL connection from a DataPower service

## Introduction

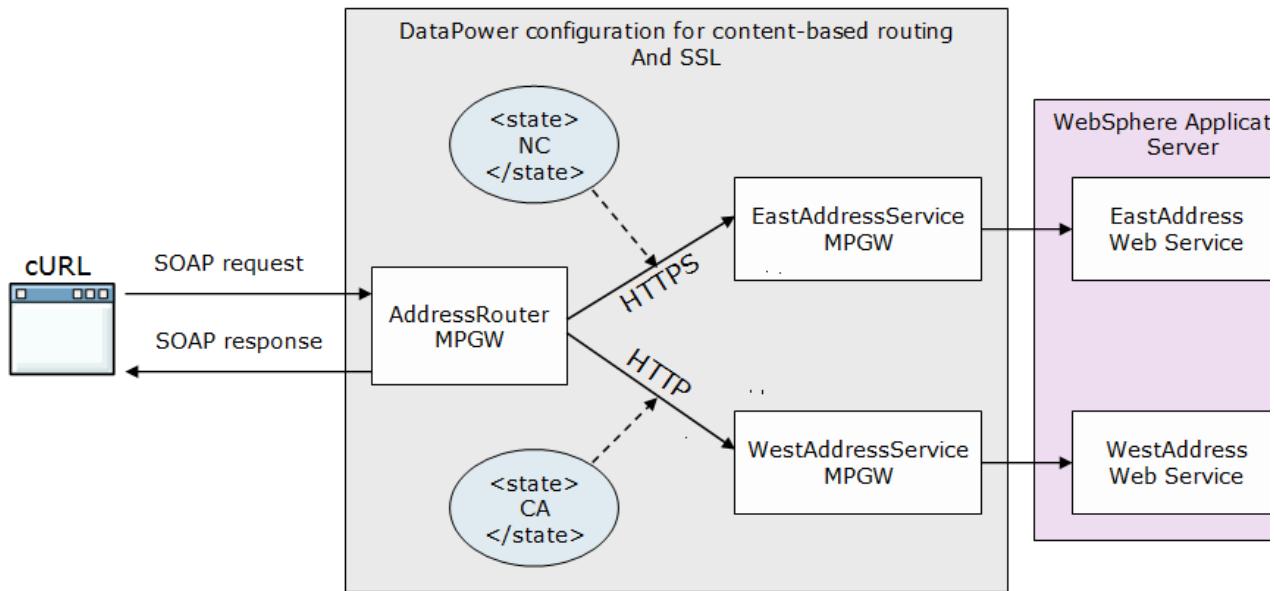
The DataPower appliance uses SSL to support secure communication. To configure an SSL communication between a client and server, you need a certificate-key pair to use during communication. In an earlier exercise, you created several multi-protocol gateway (MPGW) services that route AddressSearch requests to the correct back-end web service. In the previous exercise, you created a certificate-key pair and set up cryptographic objects that can be used during SSL communication. In this exercise, you configure an SSL connection between the routing MPGW and the MPGW that services the East addresses.

You perform the following activities:

- Create an HTTPS front side handler, with an associated crypto profile, for the EastAddressSearch MPGW
- Verify the correct behavior of this HTTPS handler
- Reconfigure the AddressRouter to use SSL to contact EastAddressSearch
- Test the SSL connection from AddressRouter to EastAddressSearch

By the end of the exercise, you changed the connection between AddressRouter MPGW to the EastAddressSearch MPGW to HTTPS.

The connection between AddressRouter and WestAddressSearch remains as HTTP.



## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- Access to the `<lab_files>` directory

## Exercise instructions

### Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 4: Creating an advanced multi-protocol gateway” and “Exercise 6: Creating cryptographic objects”.
- 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*>: directory path of the student lab files
- <*dp\_internal\_ip*>: IP address of the DataPower appliance development and administrative functions
- <*dp\_public\_ip*>: IP address of the public services on the appliance
- <*dp\_WebGUI\_port*>: port number for the appliance WebGUI
- <*backend\_server\_ip*>: IP address for the services that are running in WebSphere Application Server (AddressSearch web services)
- <*was\_server\_port*>: port number for the services that are running in WebSphere Application Server (AddressSearch web services)
- <*mpgw\_east\_port*>: port number for the service that handles East Address web services
- <*mpgw\_east\_ssl\_port*>: port number for the service that handles East Address web services over HTTPS
- <*mpgw\_west\_port*>: port number for the service that handles West Address web services
- <*mpgw\_content\_based\_routing\_port*>: port number for the service that provides the routing function

## 7.1. Verify web service behavior

Before configuring the SSL support, verify that the web service is operating correctly. Also, attempt to use HTTPS to access the same web service.

- \_\_\_ 1. Open a Terminal window.
- \_\_\_ 2. Switch to the `<lab_files>/SSL` directory.
- \_\_\_ 3. Send the cURL command directly to the EastAddressSearch service to retrieve East addresses from NC:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_east_port>/services/AddressSearch
```

A list of addresses in NC is returned.

- \_\_\_ 4. Try to use **HTTPS** to access the EastAddressSearch service. Notice that the protocol changed to `https` and a `-k` argument was added to the command string:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
https://<dp_public_ip>:<mpgw_east_port>/services/AddressSearch -k
```



### Information

The `-k` argument in cURL specifies that cURL does not verify the certificate from the SSL server, which is the DataPower service in this case.

The cURL command itself returns an error, since it cannot find an SSL handler on the receiving end. This problem is expected at this part of the exercise.

- \_\_\_ 5. Now verify correct behavior when going through the AddressRouter service:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_content_based_routing_port>/services/Addr  
essSearch
```

The same list of NC addresses is returned.

## 7.2. Add an HTTPS handler to the EastAddressSearch service

In this section, you add an HTTPS handler to the EastAddressSearch service so that it can handle requests from the AddressRouter service for SSL communications. Since the AddressRouter is initiating the SSL request, the EastAddressSearch service is configured as the SSL server.

- \_\_\_ 1. Log on to the DataPower WebGUI with a web browser:

`https://<dp_internal_ip>:<dp_WebGUI_port>`

- \_\_\_ 2. Open the EastAddressSearch multi-protocol gateway configuration page.
- \_\_\_ 3. In the Front Side Protocol section of the page, use HTTPS to add a handler.
- \_\_\_ a. Click **+** to open the selection list of handler types.
  - \_\_\_ b. Click **HTTPS (SSL) Front Side Handler**.
  - \_\_\_ c. The handler configuration dialog opens. Enter a Name of **EastAddressSearch\_SSL**.
  - \_\_\_ d. For the Local IP Address, use `<dp_public_ip>`, or its alias.
  - \_\_\_ e. Use a Port Number of `<mpgw_east_ssl_port>`.
  - \_\_\_ f. Near the bottom of the page, create an SSL Proxy profile. Click **+**.
  - \_\_\_ g. The Configure SSL Proxy Profile page that opens allows you to specify the SSL server information. Enter a Name of: **EastAddressSSLProfile**
  - \_\_\_ h. For the Reverse (Server) Crypto Profile, select **StudentServerCP**, which you created in the previous exercise.

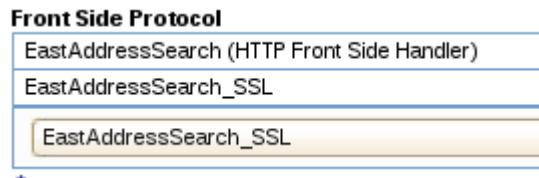
- \_\_\_ i. Leave the other settings at their defaults, which does not require client authentication. Click **Apply**.

SSL Proxy Profile

The screenshot shows the 'SSL Proxy Profile' configuration dialog. At the top are 'Apply' and 'Cancel' buttons. Below is a table with configuration options:

Name	EastAddressSSLProfile
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
SSL Direction	Reverse *
Reverse (Server) Crypto Profile	StudentServerCP <input type="button"/> + <input type="button"/> ... *
Server-side Session Caching	<input checked="" type="radio"/> on <input type="radio"/> off
Server-side Session Cache Timeout	300
Server-side Session Cache Size	20
Client Authentication Is Optional	<input type="radio"/> on <input checked="" type="radio"/> off
Always Request Client Authentication	<input type="radio"/> on <input checked="" type="radio"/> off

- \_\_\_ j. Back on the HTTPS handler configuration page, the SSL Proxy field now contains the new proxy profile object. Click **Apply**.
- \_\_\_ 4. The new HTTPS handler now shows as one of the gateway front side handlers.



- \_\_\_ 5. Since this service is not using SSL to communicate with the back-end web service, the SSL Client Crypto Profile setting remains empty.
- \_\_\_ 6. Click **Apply** for the service.

## 7.3. Test the EastAddressSearch HTTPS access

- \_\_\_ 1. As before, try to access the EastAddressSearch service by using **HTTPS**. Be sure to use **https** and a **-k** argument in the command string:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary
@findByLocation.xml
https://<dp_public_ip>:<mpgw_east_ssl_port>/services/AddressSearch -k
```

The cURL command returns the NC addresses across the SSL connection.

- \_\_\_ 2. Rerun the cURL command, but add a **-v** to the command string. This option puts the command response in verbose mode. You can see the SSL handshake commands flow between the cURL SSL client and the front side handler SSL server.

```
localuser@susehost:~/dplabs/dev/SSL> curl -H "SOAPAction: \"\"\" -H "Content-Type
: text/xml" --data-binary @findByLocation.xml https://172.16.78.44:7971/services
/AddressSearch -k -v
* About to connect() to 172.16.78.44 port 7971 (#0)
* Trying 172.16.78.44... connected
* Connected to 172.16.78.44 (172.16.78.44) port 7971 (#0)
* successfully set certificate verify locations:
*   CAfile: none
   CApath: /etc/ssl/certs/
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using AES256-SHA
* Server certificate:
*       subject: /C=US/ST=CA/L=Los Angeles/O=IBM/OU=Software Group/CN=student
*       start date: 2012-09-17 14:59:08 GMT
```

- \_\_\_ 3. Send the request to the AddressRouter service:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary
@findByLocation.xml
http://<dp_public_ip>:<mpgw_content_based_routing_port>/services/Addr
essSearch
```

The same list of NC addresses is returned. The AddressRouter service is not using SSL. Why did this work?

The Route action in AddressRouter is still using the IP address and port for the HTTP handler, which is active. In the next section, you configure the SSL client in the AddressRouter.

## 7.4. Create the SSL client support in the AddressRouter service

To use the HTTPS front side handler of EastAddressSearch, you configure the AddressRouter to point to the correct port, and to use SSL.

- \_\_\_ 1. Open the AddressRouter multi-protocol gateway configuration page.
- \_\_\_ 2. Change the Route action in the AddressRouter service policy to point to the HTTPS handler.
  - \_\_\_ a. Edit the service policy.
  - \_\_\_ b. In the policy editor, double-click the **Route** action to reconfigure it.
  - \_\_\_ c. Edit the **AddressRouter** XPath Routing Map.
  - \_\_\_ d. Click the **Rules** tab.
  - \_\_\_ e. In the list of XPath expressions, click the **edit** (pencil) icon to edit the expression that points to the EastAddressSearch port `<mpgw_east_port>`:

	Remote Host	Remote Port	SSL	
<code>Body'/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'NC']</code>	dp_public_ip	6977	off	
<code>Body'/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'CA']</code>	dp_public_ip	6978	off	

- \_\_\_ f. Change the Remote Port to: `<mpgw_east_ssl_port>`
- \_\_\_ g. Set SSL to **on**. This setting tells the service to use its crypto profile to configure the SSL client information.

- \_\_\_ h. The XPath expression is not changed. Click **Apply**.

### Edit Rules

XPath Expression	<code>/*[local-name()='Envelope']/*[local-name()='Content']/*[local-name()='Text']</code>
Remote Host	dp_public_ip
Remote Port	7971
SSL	<input checked="" type="radio"/> on <input type="radio"/> off *
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>	

- \_\_\_ i. **Apply** the XPath routing map.
- \_\_\_ j. Click **Done** for the Route action.
- \_\_\_ k. Click **Apply Policy** in the policy editor, and close the window.
- \_\_\_ 3. The SSL client information still must be configured. In the Back side settings section of the page, select **StudentClientCP** for the SSL Client Crypto Profile. This object is the crypto profile object that you created in the previous exercise.
- \_\_\_ 4. Click **Apply** for the service.
- \_\_\_ 5. Examine the SSL proxy profiles for the two services.
- \_\_\_ a. Select **Objects > Crypto Configuration > SSL Proxy Profile**.

### Configure SSL Proxy Profile

 [Refresh List](#)

Name	Status	Op-State	Logs	SSL Direction	Forward (Client) Crypto Profile	Reverse (Server) Cry
AddressRouter	new	up		forward	StudentClientCP	
EastAddressSSLProfile	new	up		reverse		StudentServerCP

An SSL proxy profile object is the container for the SSL crypto profile objects within a single service.

In the EastAddressSearch service, you explicitly created and named the SSL proxy profile EastAddressSSLProfile. For this service, you specified the SSL crypto profile as an SSL server.

For the AddressRouter, you explicitly selected the SSL client crypto profile. The service automatically created the SSL proxy profile, and gave it the name of the service.

## 7.5. Test the SSL connection from the AddressRouter to the EastAddressSearch

Now that the SSL connection between AddressRouter and EastAddressSearch is configured on both ends of the connection, send the request.

- \_\_\_ 1. Send the cURL command to AddressRouter:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary
@findByLocation.xml
http://<dp_public_ip>:<mpgw_content_based_routing_port>/services/AddressSearch
```

You get a list of NC addresses.

- \_\_\_ 2. How do you know that AddressRouter is using the SSL connection? The obvious answer is that you reconfigured it that way. But you can also see it in the log. Open the system log, and have it show **all** of the entries.
- \_\_\_ 3. Look in the log entries for the AddressRouter sending an HTTP request outbound. In the example below, you can see the request. Notice that the protocol is **https**, and the port is **7nn1**, which is **<mpgw\_east\_ssl\_port>** for this test. This shows that AddressRouter is using HTTPS to contact EastAddressSearch.

debug	599841	172.16.80.115	0x80e00159	mpgw (AddressRouter): Outbound HTTP with reused TCP session using HTTP/1.1 to https://dp_public_ip:7971/services/AddressSearch
-------	--------	---------------	------------	---

- \_\_\_ 4. Look higher up in the log, to more current entries. You see the HTTPS front side handler **EastAddressSearch\_SSL** for EastAddressSearch responding to a request. In the log, this type of handler is noted as “source-https”, which is the argument for this handler type when doing command-line configuration instead of the WebGUI.

info	143725	172.16.78.44	0x80e0013a	source-https (EastAddressSearch_SSL): Received HTTP/1.1 POST for /services/AddressSearch from 172.16.78.44
------	--------	--------------	------------	---

Also, notice that this number is a different transaction number (143725 in this example) from the one for AddressRouter (599841). Although EastAddressSearch is being invoked from another service on the same appliance, it is still treated as a separate transaction.

- \_\_\_ 5. Look for more current EastAddressSearch entries. You find several that show the full URL that was used to call EastAddressSearch. Again, you can see the HTTPS protocol and the SSL port in the request.

debug	143725	request	172.16.78.44	0x80e003ab	mpgw (EastAddressSearch): Finished parsing: https://172.16.78.44:7971/services/AddressSearch
debug	143725	request	172.16.78.44	0x80e003a6	mpgw (EastAddressSearch): Parsing document: 'https://172.16.78.44:7971/services/AddressSearch'

- \_\_\_ 6. Save your configuration.

\_\_\_ 7. Test access to EastAddressSearch by using HTTP:

```
curl -H "SOAPAction: \"\"\" -H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_east_port>/services/AddressSearch
```

A list of addresses in NC is still returned. The HTTP handler in EastAddressSearch is still enabled. This MPGW can be accessed by using HTTP and by using HTTPS.

**End of exercise**

## Exercise review and wrap-up

In this exercise, you configured both server-side and client-side SSL on two separate multi-protocol gateways.



# Exercise 8. Protecting against XML threats

## What this exercise is about

XML and web services are subject to various types of attacks that are broadly referred to as XML structural attacks, XML content-based attacks, and denial-of-service attacks. This exercise demonstrates the major XML threat protection features of the DataPower appliance.

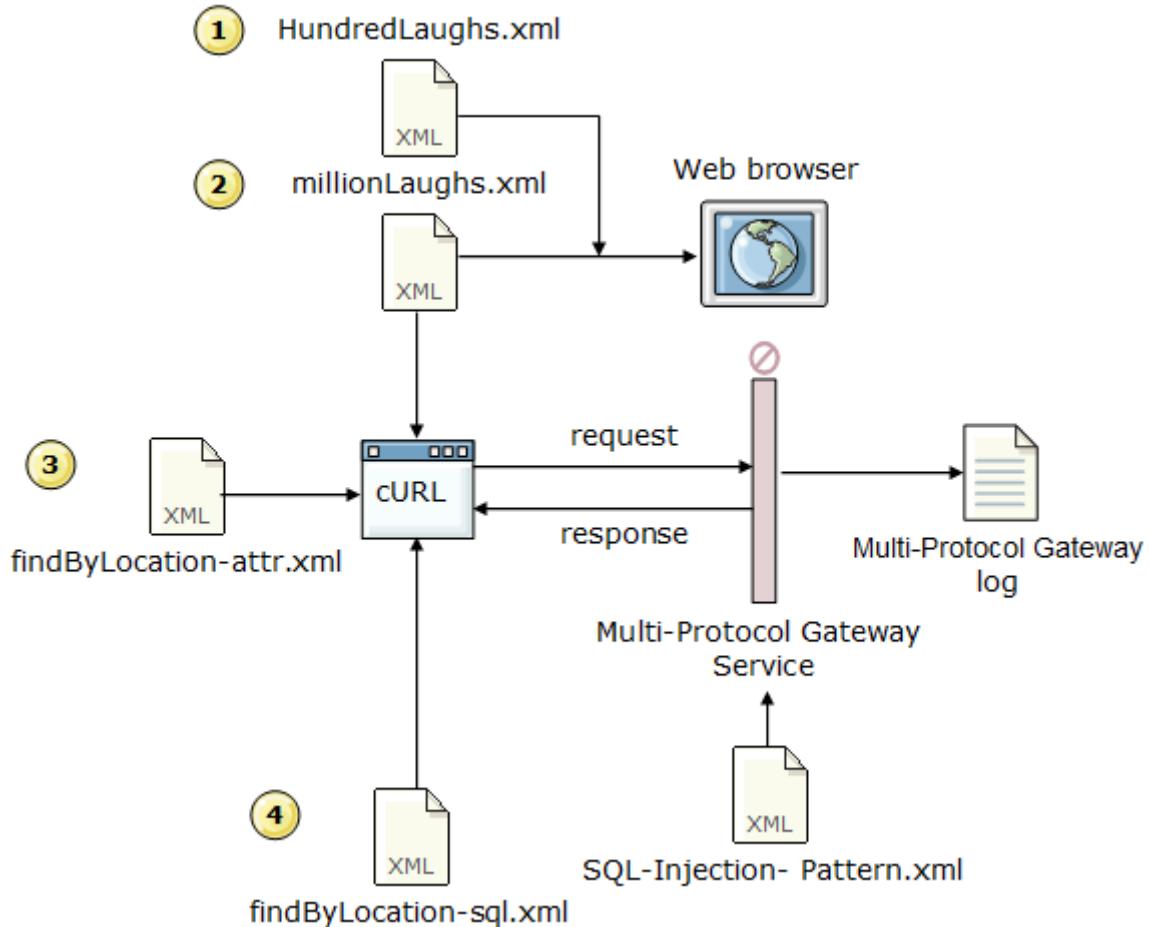
## What you should be able to do

At the end of the exercise, you should be able to:

- Run a recursive entity attack simulation
- Perform a recursive entity threat protection test
- Enable excessive attribute count threat protection
- Enable SQL injection attack prevention

## Introduction

One of the important capabilities of the DataPower SOA appliance is its ability to protect against XML threats before the traffic hits the application server. Since the appliance processes the messages at wire speed, the enterprise can protect itself from these threats at minimal cost. The multi-protocol gateway service inherently provides XML threat protection. This exercise demonstrates a few of these attacks, and shows you how to use and configure some of the XML threat capabilities.



This exercise is organized into four sections:

1. The first section of the exercise demonstrates the resource consumption characteristics of a typical recursive entity attack by using only the web browser.
2. The second section of the exercise demonstrates the recursive entity attack protection feature of the DataPower SOA appliance.
3. The third section of this exercise demonstrates how the default settings of XML parser can be overridden.
4. The fourth section of this exercise demonstrates the SQL injection attack prevention features of the appliance.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance

- Completion of (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- The **Address Search** web services that run on WebSphere Application Server
- Access to the `<lab_files>` directory

## Exercise instructions

### Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 4: Creating an advanced multi-protocol gateway”.
- 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
  - *<dp\_internal\_ip>*: XML threat instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<dp\_public\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<mpgw\_threat\_protect\_port>*: instructor-assigned port number for the multi-protocol gateway you are creating in this exercise
  - *<mpgw\_east\_port>*: instructor-assigned port number for the East multi-protocol gateway you created in the previous exercise, and are reusing in this exercise

## 8.1. Recursive entity attack simulation

This part demonstrates the effects of a recursive entity attack. A recursive entity attack is run when a seemingly small XML document expands into a large document, which can cause the parser to use system resources. This results in a denial-of-service attack, as the system cannot process any other client requests. As shown in the diagram below, two test cases are run.

- **Test case 1:** demonstrates the phenomenon of recursive entity expansion
- **Test case 2:** gives you an opportunity to actually run a recursive entity expansion attack and cause a denial-of-service attack

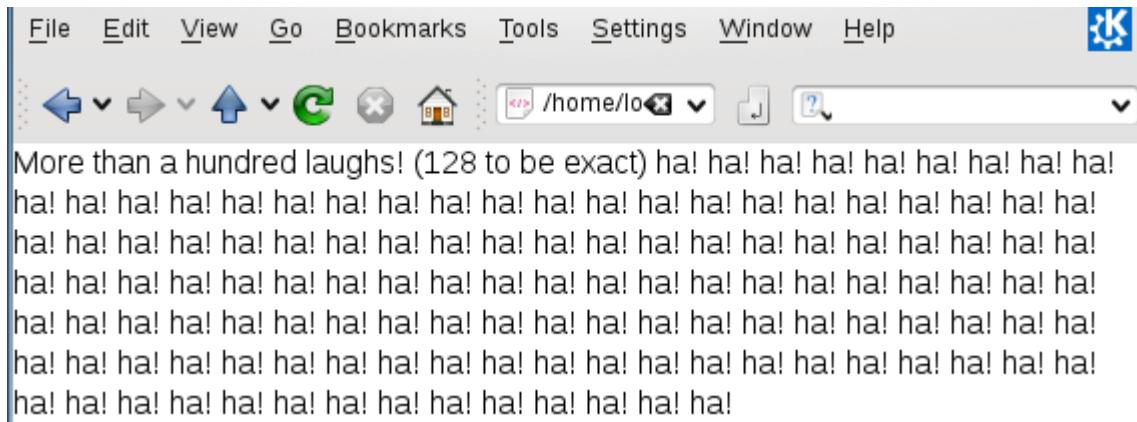
- 1. Run test case 1 to understand the phenomenon of recursive entity expansion.
  - a. Using any text editor, go to the <lab\_files>/XMLthreat directory and open the hundredLaughs.xml file. Observe the relatively small XML document and the entity expansion that is employed.
  - b. Open Konqueror and locate hundredLaughs.xml from the <lab\_files>/XMLthreat directory by using the **File > Open File > hundredLaughs.xml** menu option.



### Information

The Firefox browser detects the large recursive entity expansion in test case 2, so you use a different tool. You simulate the browser behavior by using the Konqueror utility to complete the entity expansion. Konqueror can be opened from the bottom desktop panel by using the icon that is a blue globe on a toothed gear. If you hover over the icon, "Konqueror" displays.

Konqueror completes the recursive entity expansion, and the resulting file is displayed.



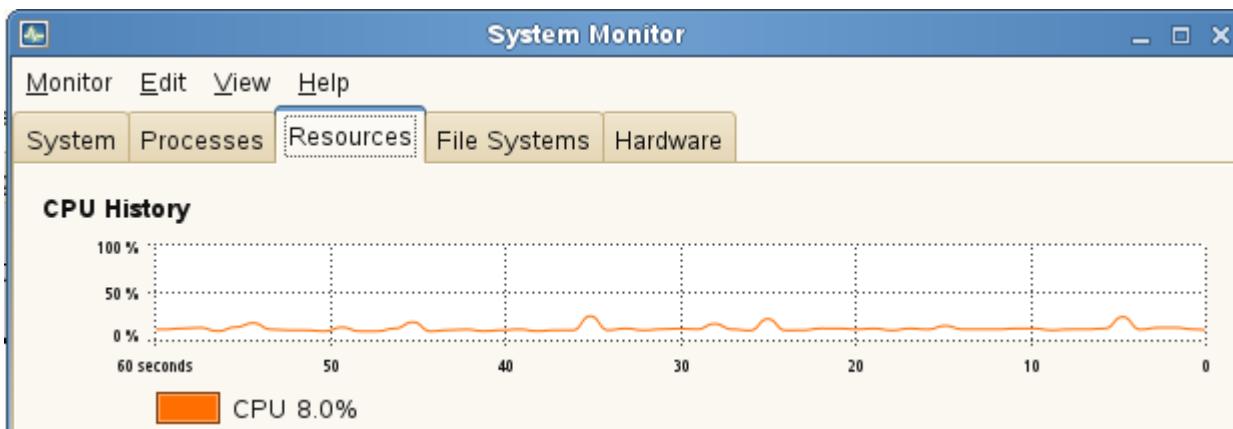
- \_\_\_ c. **(Optional)** You might experiment further with the `hundredLaughs.xml` file by increasing the number of recursive entity expansions that occur.



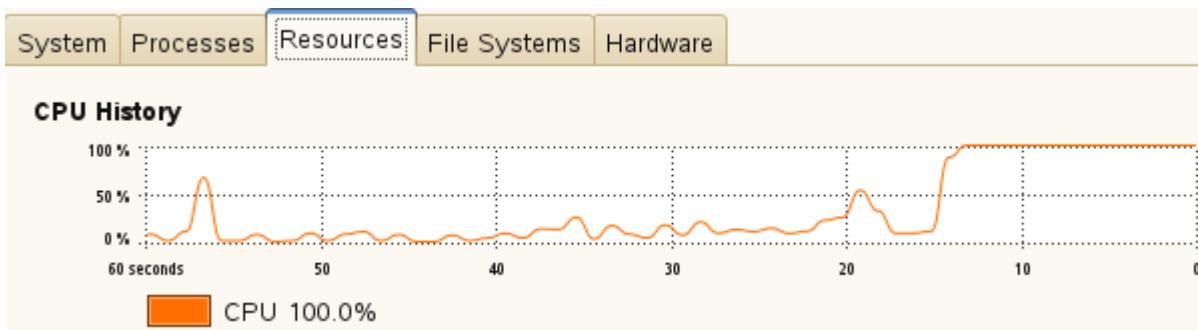
### Information

You can simultaneously observe and track the overall system resource consumption (memory and processor) by using the system monitor.

- \_\_\_ d. Right-click the CPU graph on the lower toolbar on the desktop and select **Open System Monitor**. Observe the processor utilization from the **Resources** tab.



- \_\_\_ 2. Run test case 2 to open a larger recursive entity expansion attack and see its effect on your system.
- \_\_\_ a. Using Konqueror, locate and open the `billionLaughs.xml` file from the `<lab_files>/XMLthreat` directory.
- \_\_\_ b. View the processor utilization from the **Resource** tab of the System Monitor. Notice the spike in processor utilization as the utility starts to load the `billionLaughs.xml` file. The recursive entity expansion is initiated, causing the parser to use the processor resources.



- \_\_\_ c. While the utility is attempting to open the `billionlaughs.xml` file, switch to the **Processes** tab in System Monitor. Verify that the process that runs Konqueror is the process that uses all available processor cycles.

**Note**

You might set the monitor to view active processes before you can see the Konqueror process.

The screenshot shows the System Monitor application window. The title bar says "System Monitor". The top menu bar has tabs: System, Processes (which is selected and highlighted in blue), Resources, File Systems, and Hardware. Below the tabs, it displays load averages: "Load averages for the last 1, 5, 15 minutes: 1.29, 0.57, 0.26". The main area is a table titled "Processes" with the following columns: Process Name, Status, % CPU, Nice, ID, and Waiting Channel. The table contains three rows:

Process Name	Status	% CPU	Nice	ID	Waiting Channel
konqueror	Running	96	0	1806	0
gnome-system-monitor	Running	2	0	22889	0
firefox	Sleeping	0	0	25109	0

- \_\_\_ d. Select the problematic `konqueror` process and click **End process**.  
\_\_\_ e. Close the System Monitor window.

## 8.2. Recursive entity threat protection test

In this section, you create a basic multi-protocol gateway that accepts and processes an XML request and echoes the request back to the client. In a more realistic scenario, the multi-protocol gateway forwards the request to a back-end server, as shown in the following graphic. However, for the purposes of demonstrating the XML threat protection features of the DataPower SOA appliance, you test this recursive entity threat protection against a multi-protocol gateway that is configured to operate in loopback mode.

Create a basic pass-through multi-protocol gateway that operates in loopback mode.

- 1. In the IBM WebSphere DataPower SOA Appliance WebGUI, click **Control Panel > Multi-Protocol Gateway**.



- 2. From the Configure Multi-Protocol Gateway screen, click **Add**.
- 3. Enter `XMLthreat` as the name of the new gateway. You can optionally enter a description for the gateway in the summary field, such as `XML threat test MPGW`.
- 4. In the **XML Manager** field, leave the **default** XML manager selected.
- 5. In the **Multi-Protocol Gateway Policy** field, click **+** (new) to create a multi-protocol gateway policy.

The screenshot shows the 'Configure Multi-Protocol Gateway' dialog box with the 'General' tab selected. The tabs include General, Advanced, Stylesheet Params, Headers, Monitors, WS-Addressing, and WS-ReliableMessage.

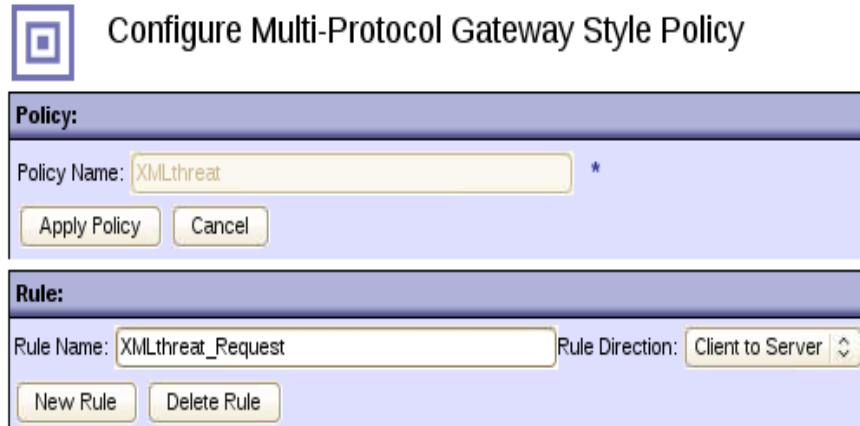
**General Configuration**

<b>Multi-Protocol Gateway Name</b> <input type="text" value="XMLthreat"/> *	<b>XML Manager</b> <input type="text" value="default"/> <b>+</b> <b>...</b> *
<b>Summary</b> <input type="text" value="XML threat protection testing"/>	<b>Multi-Protocol Gateway Policy</b> <input type="text" value="(none)"/> <b>+</b> <b>...</b> *
<b>Type</b> <input checked="" type="radio"/> dynamic-backend <input type="radio"/> static-backend	<b>URL Rewrite Policy</b> <input type="text" value="(none)"/> <b>+</b> <b>...</b>

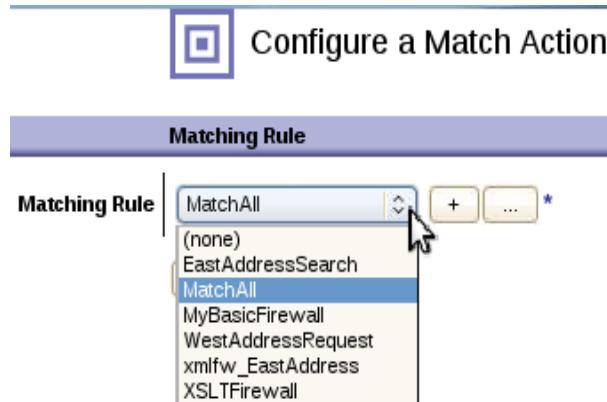
**Apply** **Cancel**

- 6. Enter `XMLthreat` as the processing policy name.

- \_\_\_ 7. Configure a message processing rule to forward any client request messages to the back-end service.
- \_\_\_ a. In the processing rule editor for XMLthreat, click **New Rule** and rename the rule to: XMLthreat\_Request



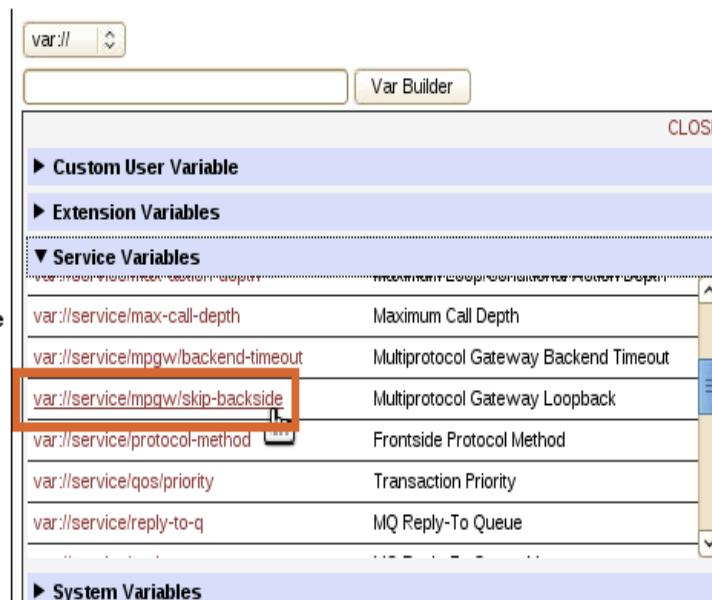
- \_\_\_ b. Set the processing rule direction to **Client to Server**.
- \_\_\_ c. Double-click the **Match** action icon to configure it.
- \_\_\_ d. Click the list box arrows and select the **MatchAll** matching rule.



- \_\_\_ e. Click **Done** to use the MatchAll matching rule.
- \_\_\_ 8. Drag the **Advanced** icon to the processing rule, placing it after the Match action.
- \_\_\_ a. Double-click the **Advanced** icon to configure it.
- \_\_\_ b. Select **Set Variable**.
- \_\_\_ c. Click **Next**.

- \_\_\_ d. Select the `var://service/mpgw/skip-backside` **Service Variable** for the Variable Name.

### Set Variable



The screenshot shows the 'Set Variable' dialog with the 'Variable Name' field set to 'var://'. The 'Service Variables' section is expanded, listing various service variables. The variable 'var://service/mpgw/skip-backside' is highlighted with a red box. The 'Variable Assignment' field is empty.

Variable Name	Description
<code>var://service/max-call-depth</code>	Maximum Call Depth
<code>var://service/mpgw/backend-timeout</code>	Multiprotocol Gateway Backend Timeout
<code>var://service/mpgw/skip-backside</code>	Multiprotocol Gateway Loopback
<code>var://service/protocol-method</code>	Frontside Protocol Method
<code>var://service/qos/priority</code>	Transaction Priority
<code>var://service/reply-to-q</code>	MQ Reply-To Queue

- \_\_\_ e. Enter the number **1** for the Variable Assignment.

### Set Variable



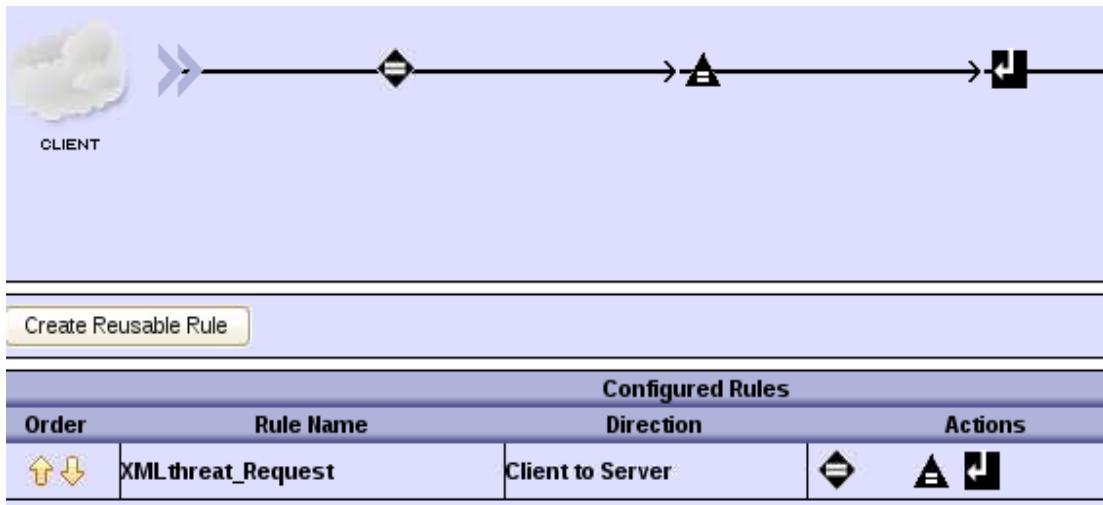
The screenshot shows the 'Set Variable' dialog. The 'Variable Name' field is set to 'var://'. The 'Variable Assignment' field contains the value '1', which is highlighted with a red box. The 'Asynchronous' field has the 'on' radio button selected. At the bottom are 'Delete', 'Done', and 'Cancel' buttons.



#### Note

The Set Variable action is configured to skip the backside processing. Although the multi-protocol gateway does not offer a loopback option when selecting the back-end type, the gateway can be configured to do a loopback. The back end is configured for dynamic selection and the processing that the processing policy controls. Since a service variable is configured to skip backside processing, the inbound message is returned to the client.

- \_\_\_ 9. Click **Done**.
- \_\_\_ 10. Add a **Results** action to the `XMLthreat_Request` document processing rule.
  - \_\_\_ a. Click **Done** to save the **Results** action settings.
  - \_\_\_ b. Click **Apply Policy** to save the changes.
  - \_\_\_ c. Confirm that the new document processing rule is shown in the list of configured rules.



- \_\_\_ 11. Click **Apply Policy** and then **Close Window** to close the XML threat document policy editor.
- \_\_\_ 12. Set a dynamic back-end connection to the XML threat multi-protocol gateway.
  - \_\_\_ a. On the Configure Multi-Protocol Gateway page, select **dynamic-backend** as the back-end connection type.

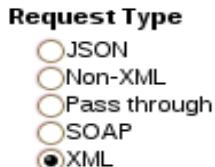
#### General Configuration

**Multi-Protocol Gateway Name**  
 \*

**Summary**

**Type**  
 dynamic-backend  
 static-backend

\_\_\_ 13. Set the **Request Type** field to **XML**.



### Information

When you create a multi-protocol gateway, the request type is automatically set to be of type **SOAP**. However, since you are sending plain XML, modify this setting to **XML** before the multi-protocol gateway is ready to accept traffic for this test.

Create a front side protocol handler to receive client requests over an HTTP connection.

\_\_\_ 14. Create an HTTP front side protocol handler named `XMLthreat`.

- \_\_\_ a. In the Front Side Protocol section, click **+** (new).
- \_\_\_ b. Select **HTTP Front Side Handler** from the list.

#### Front side settings

The screenshot shows the 'Front Side Protocol' configuration screen. A red box highlights the 'Add' button (represented by a plus sign) in the toolbar above the list. A second red box highlights the 'HTTP Front Side Handler' entry in the 'Create a New:' dropdown menu.

- \_\_\_ c. Configure the new HTTP front side handler with the following values. Leave all other settings to the default values.
- **Name:** XMLthreat
  - **Local IP address:** <dp\_public\_ip>
  - **Port Number:** <mpgw\_threat\_protect\_port>, as assigned by the instructor

HTTP Front Side Handler

<b>Name</b>	XMLthreat
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	
Local IP Address	dp_public_ip
Port Number	6952
HTTP Version to Client	HTTP 1.1

- \_\_\_ d. Click **Apply** to save the changes that are made to the HTTP front side handler.
- \_\_\_ 15. Apply the Multi-Protocol Gateway.
- \_\_\_ a. Click **Apply** to save the changes that are made to the multi-protocol gateway.

**Configure Multi-Protocol Gateway**

**General**   **Advanced**   **Stylesheet Params**

**Apply**   **Cancel**   **Delete**

Multi-Protocol Gateway status: [up]

**General Configuration**

**Multi-Protocol Gateway Name**  
XMLthreat \*

**Summary**  
XML threat protection testing

**Type**

- \_\_\_ b. Verify that the Multi-Protocol Gateway status is **up**.
- \_\_\_ c. Click **Save Config**.

- \_\_\_ d. The multi-protocol gateway XMLthreat is now ready for testing.

## 8.3. Test the multi-protocol gateway

- \_\_\_ 1. Run the recursive entity tests against the XMLthreat multi-protocol gateway.
- \_\_\_ a. In a terminal window, go to the <lab\_files>/XMLthreat directory, and run the following command:  

```
curl --data-binary @hundredLaughs.xml
http://<dp_public_ip>:<mpgw_threat_protect_port>/
```

You get the expanded hundredLaughs XML response.
- \_\_\_ b. Rerun the test, but use billionLaughs.xml instead. Observe that the response is a fault string of Malformed content (from the client).
- \_\_\_ c. On the System Log page, click Refresh Log. Observe the messages document size limit of 4194304 bytes exceeded and XML parser limits exceeded. The limit of 4194304 bytes for an XML document is specified in the default XML manager.

07:05:21	mpgw	error	5158177	error	172.16.80.11	0x00030003	mpgw (XMLthreat): XML parser limits exceeded
07:05:21	multipstep	error	5158177	request	172.16.80.11	0x80c00008	mpgw (XMLthreat): rule (XMLthreat_Request): implied action Parse input as XML failed: document size limit of 4194304 bytes exceeded, aborting
07:05:21	xmlparse	error	5158177	request	172.16.80.11	0x80e003aa	mpgw (XMLthreat): document size limit of 4194304 bytes exceeded, aborting

## 8.4. Excessive attribute count threat protection

This section uses the XMLTest firewall that was created in the previous part to demonstrate the steps that are required to override the default XML parser values set by the XML manager. This feature is demonstrated by modifying the XML attribute count value.

- \_\_\_ 1. Modify the XMLTest firewall configuration to override the default XML parser limits.
  - \_\_\_ a. From the **Control Panel**, click the **multi-protocol gateway** icon.
  - \_\_\_ b. From the multi-protocol gateway Service list, select the **XMLthreat** multi-protocol gateway.
  - \_\_\_ c. To examine the current settings of the default XML manager, click [...] (edit).
  - \_\_\_ d. Click the **XML Parser** tab. Notice the value **4194304** for the **XML Bytes Scanned** limit, which triggered the message in the previous test. Also, notice that the default **Maximum Attribute Count** is **128**. Click **Help** for a description of each of these values.

These settings do not impact resource allocation, and are used only as part of your threat protection.

XML Bytes Scanned	<input type="text" value="4194304"/>	bytes
XML Element Depth	<input type="text" value="512"/>	
XML Attribute Count	<input type="text" value="128"/>	
XML Maximum Node Size	<input type="text" value="33554432"/>	bytes
XML Maximum Distinct Prefixes	<input type="text" value="0"/>	
XML Maximum Distinct Namespaces	<input type="text" value="0"/>	
XML Maximum Distinct Local Names	<input type="text" value="0"/>	
XML External Reference Handling	<input type="button" value="Forbid"/>	

- \_\_\_ e. Click **Cancel** to close the window.
- \_\_\_ f. To modify the default XML parser limits without modifying the default XML manager, click the **XML Threat Protection** tab.

- \_\_\_ g. In the Single Message XML Denial of Service (XDoS) Protection section of the page, select **On** for **Gateway parser limits**. This action shows the overridable values.

### Single Message XML Denial of Service (XDoS) Protection

<b>Maximum Message Size</b>	<input type="text" value="0"/> KB
<b>Gateway parser limits</b>	<input checked="" type="radio"/> on <input type="radio"/> off
<b>XML Attribute Count</b>	<input type="text" value="128"/> *
<b>XML Element Depth</b>	<input type="text" value="512"/> *
<b>XML Maximum Node Size</b>	<input type="text" value="33554432"/> bytes *
<b>XML Maximum Distinct Prefixes</b>	<input type="text" value="0"/>
<b>XML Maximum Distinct Namespaces</b>	<input type="text" value="0"/>
<b>XML Maximum Distinct Local Names</b>	<input type="text" value="0"/>
<b>Attachment Byte Count Limit</b>	<input type="text" value="2000000000"/> *
<b>Attachment Package Byte Count Limit</b>	<input type="text" value="0"/> *
<b>XML External Reference Handling</b>	<input type="button" value="Forbid"/> *
<b>Recursive Entity Protection</b>	<input checked="" type="radio"/> on <input type="radio"/> off

- \_\_\_ h. Change the **Max. XML Attribute Count** from the default, 128, to 10. This setting is changed to a low number to simplify the exercise.
- \_\_\_ i. Click **Apply** to save the modified multi-protocol gateway settings.
- \_\_\_ j. Click **Save Config**.
- \_\_\_ 2. Run the tests to demonstrate the XML threat protection features of the **XMLthreat** multi-protocol gateway.
- \_\_\_ a. In a terminal window, go to the `<lab_files>/XMLthreat` directory, and run the following command:

```
curl --data-binary @findByLocation-Attr.xml
http://<dp_public_ip>:<mpgw_threat_protect_port>/
```

Since the `findByLocation-Attr.xml` has an element with 11 attributes, you get a response fault string of Malformed content (from client).

- \_\_\_ b. Click **Refresh** on the System Log page.

07:14:29	mpgw	error	5171889	error	172.16.80.11	0x00030003	mpgw (XMLthreat): XML parser limits exceeded
07:14:29	multipstep	error	5171889	request	172.16.80.11	0x80c00008	mpgw (XMLthreat): rule (XMLthreat_Request): implied action Parse input as XML failed: attribute limit of 10 per element exceeded, aborting at offset 711 of http://172.16.78.44:6952/
07:14:29	xmlparse	error	5171889	request	172.16.80.11	0x80e003aa	mpgw (XMLthreat): attribute limit of 10 per element exceeded, aborting at offset 711 of http://172.16.78.44:6952/

The trace entries for the request failed because the attribute limit is being exceeded.

## 8.5. SQL injection attack prevention

SQL injection is a technique for using web applications that use client-supplied data in SQL queries, without stripping potentially harmful characters. SQL manipulation is perhaps the most popular type of SQL injection attack. SQL manipulation involves modifying the SQL statement through set operations (such as UNION) or altering the WHERE clause to return a different result. The most widely known attack is to modify the WHERE clause of the user authentication statement so the WHERE clause result is always TRUE.

The classic SQL manipulation is during the login authentication. A simplistic web application might check user identity by running the following query and checking to see whether any rows were returned:

```
SELECT * FROM users WHERE username='XXX'
```

The injection can occur when the user or attacker is prompted for the user name, and the user enters something like: 'Jim' or 'a' = 'a'. Depending on the programming language that is used in the server that does the authentication, the resulting string might run as:

```
SELECT * FROM users WHERE username='Jim' or 'a' = 'a'
```

Based on operator precedence, the WHERE clause is true for every row, and the attacker gains access to the application. This type of SQL injection vulnerability is known as "incorrectly filtered escape characters."

The best way to protect against this attack is to reconfigure the East multi-protocol gateway service that is created in Exercise 4 to protect against possible SQL injection attacks.

- \_\_\_ 1. Open and examine the default SQL injection patterns file.

SQL injection patterns are maintained in the external file `store:///SQL-Injection-Patterns.xml`. The filter scans for many commonly known methods of SQL injection, which are primarily data destruction attacks.

- \_\_\_ a. From the **Control Panel**, select the **File Management** icon.
- \_\_\_ b. Expand the `store` directory on the File Management page.
- \_\_\_ c. Scroll down, locate and open the `SQL-Injection-Filter.xsl` file, and then the `SQL-Injection-Patterns.xml` file by selecting each file name link. The files open in your default browser. Spend some time reading the contents of these two files. Close the browser when finished.

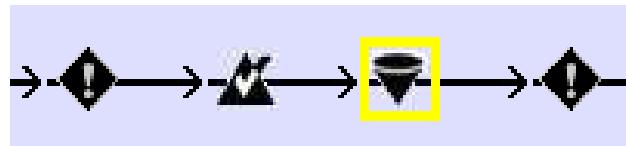


- \_\_\_ 2. Reconfigure the `EastAddressSearch` multi-protocol gateway policy with a new Filter action.
  - \_\_\_ a. From the Control Panel, select the **multi-protocol gateway** icon.
  - \_\_\_ b. From the multi-protocol gateway service catalog listing, select the `EastAddressSearch` multi-protocol gateway that was created in the previous exercise. Click the [...] (edit) to edit the multi-protocol gateway policy.
  - \_\_\_ c. Add a **Filter** action to the rule configuration area after the existing **Validate** action. The **Filter** action uses a style sheet to complete an accept or reject action on the incoming message.



### Note

The graphic might not exactly match your rule, depending on which exercises are completed. The graphic that is displayed here includes completion of the error reporting exercise.



- \_\_\_ d. Double-click the new **Filter** icon to modify the action.
- \_\_\_ e. On the Configure Filter action page, change the **Processing Control File** list selection from `local` to `store`
- \_\_\_ f. In the adjacent file list, select the `SQL-Injection-Filter.xsl` style sheet and click **Done**.

**Filter**

<b>Transform</b>	<input type="text" value="store:///"/> <input type="text" value="SQL-Injection-Filter.xsl"/> <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <i>Stylesheet Summary: Scan document for SQL injection attacks.</i>
<b>Asynchronous</b>	<input checked="" type="radio"/> on <input type="radio"/> off

- \_\_\_ g. Click **Apply Policy**, and then click **Close Window**.
- \_\_\_ h. On the main multi-protocol gateway creation screen, click **Apply**.

3. Send test messages to the multi-protocol gateway to test for SQL injection attacks.
- \_\_ a. Using an editor, open the <lab\_files>/XMLthreat/findByLocation-sql.xml file. Observe that <city> contains an SQL statement.
  - \_\_ b. In a terminal window, go to the <lab\_files>/XMLthreat> directory and run the following command:

```
curl -H "SOAPAction:\\" \" " -H "Content-Type: text/xml"
--data-binary @findByLocation-sql.xml
http://<dp_public_ip>:<mpgw_east_port>/EastAddress/services/Address
Search
```



### Note

You can also send this command to the AddressRouter multi-protocol gateway.

- \_\_ c. You get an HTML response generated by the filter-custom-error.xsl file in the error rule. The response resembles:

```
<body>
<h2>My Company Benefits</h2>
<p>Illegal operation attempted</p>
<p>This error will be reported to Application Security</p>
<p>Thank you.</p>
<p><font size="2">Transaction ID:88371</font></p>
<p><font size="2">Error code:0x00d30003</font></p>
<p><font size="2">Error-subcode:0x00d30003</font></p>
<p><font size="2">Error message:Message contains restricted
content</font></p>
</body>
```



### Information

If you did not finish the Handling Errors exercise, you receive a SOAP message that states:  
<faultstring>Message contains restricted content (from client)</faultstring>

- \_\_ d. From the Control Panel, click **View Logs** to open the System Log page.

Look for the traces of the SQL injection filter that is called and rejecting the request.

07:28:11	multistep	error	5201985	request	172.16.80.11	0x80c00009	mpgw (EastAddressSearch): request EastAddressRequest #4 filter: 'dpvar_1 store:///SQL-Injection-Filter.xsl' failed: Message contains restricted content
07:28:11	multistep	error	5201985	request	172.16.80.11	0x80c00078	mpgw (EastAddressSearch): Rejected by filter 'EastAddressRequest_filter_2' of rule 'EastAddressRequest'.
07:28:11	xslt	error	5201985	request	172.16.80.11	0x80c00010	mpgw (EastAddressSearch): Execution of 'store:///SQL-Injection-Filter.xsl' aborted: Message contains restricted content
07:28:11	xsltmsg	error	5201985	request	172.16.80.11	0x8180004d	mpgw (EastAddressSearch): SQL INJECTION: Message from '172.16.80.11' contains possible SQL Injection Attack of type 'SQL LIKE% Match'. Offending content: "%". Full Message: '<?xml version="1.0" encoding="UTF-8"?> <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:Body> <q1:findByLocation xmlns:q1="http://east.address.training.ibm.com"><city>SELECT USERS FROM DBA_USERS WHERE USERS LIKE '%CHAIR%'</city> <state>NC</state></q1:findByLocation> </soapenv:Body> </soapenv:Envelope>'.

- \_\_\_ e. Using an editor, open and view the `<lab_files>/XMLthreat/SQL-REJECT-STRINGS.txt` file. You see some other examples of SQL statements.

Copy one of these SQL statements into the `<city>` element of the `findByLocation-sql.xml` file. Rerun the cURL command and check the response and system logs again. You see results similar to what you received from the previous test.

## End of exercise

## Exercise review and wrap-up

In this exercise, you simulated some of the XML threats and configured a multi-protocol gateway to study the XML threat protection features offered by the DataPower SOA appliance. In the exercise, a recursive entity and the SQL injection attack are simulated. You then configured the XML threat protection capabilities of the multi-protocol gateway to demonstrate how the DataPower SOA appliance can protect valuable back-end resources from these potentially malicious XML attacks.



# Exercise 9. Configuring a web service proxy

## What this exercise is about

In this exercise, you create a web service proxy (WS-Proxy) service that virtualizes or proxies the East and West Address Search web service. A web service proxy allows you to mask the actual endpoint of the web service. The web service proxy configuration is done by uploading a WSDL document for each service. After you create a web service proxy, you can configure a policy with rules and actions for each service that is defined within the proxy.

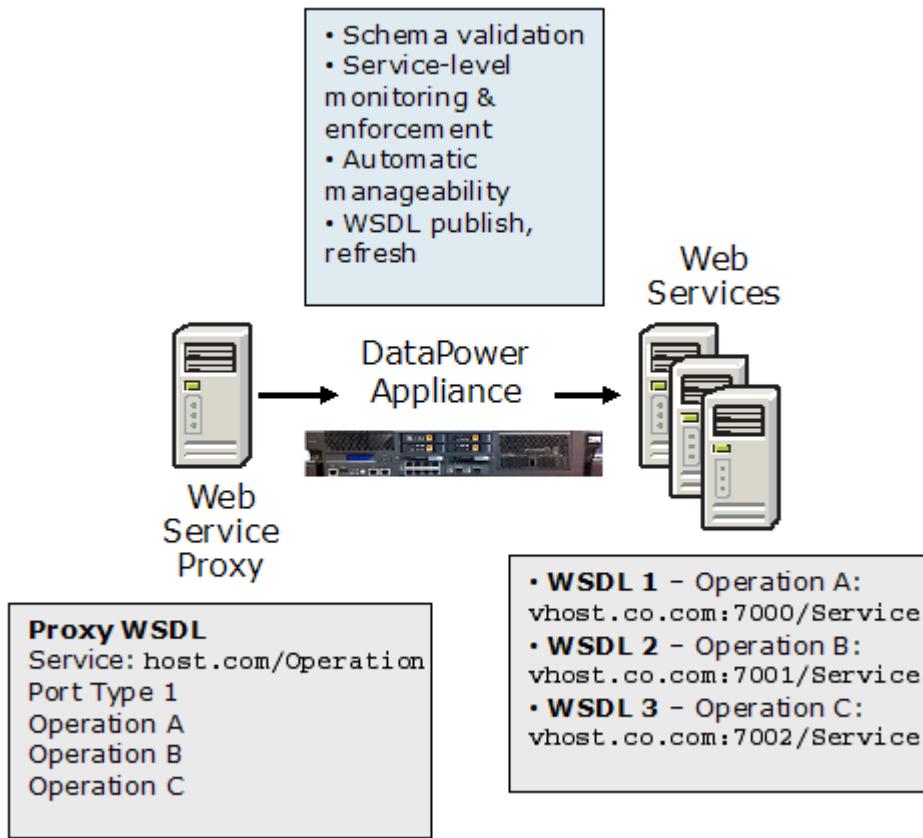
## What you should be able to do

At the end of the exercise, you should be able to:

- Configure a web service proxy to virtualize an existing set of web services
- Create a policy within the web service proxy

## Introduction

A web service proxy allows you 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 perform these tasks on the web service proxy.



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

In this exercise, you upload a WSDL file for both the East and West Address Search web services. The WSDL files contain the endpoint address of the respective web service. Using the web service proxy, you create a virtual address for these services, which the client calls to invoke the respective web service. In addition, the web service proxy validates both request and response messages and publishes your 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 `findByName` operation. The policy has a **Filter** action to verify that the `<title>` element in the message consists of valid values (Mr, Mrs, Ms, Miss, or Dr).

You use the Web Services Explorer in Eclipse to test your web service proxy.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **Eclipse**, to send requests to the DataPower appliance
- The **Address Search** web service that is running on WebSphere Application Server
- Access to the `<lab_files>` directory

## Exercise instructions

### Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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
  - *<dp\_public\_ip>*: IP address of the DataPower appliance development and administrative functions
  - *<backend\_server\_ip>*: IP address for the services that are running in WebSphere Application Server (AddressSearch web services, LDAP, registry)
  - *<wsp\_proxy\_port>*: port number for the AddressSearchProxy
  - *<was\_server\_port>*: port number for the services that are running in WebSphere Application Server (AddressSearch web services, LDAP, registry)

## 9.1. Create a web service proxy for EastAddressSearch web service

Configure a new web service proxy on the IBM WebSphere DataPower SOA Appliance to forward requests to the East Address Search web service.

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

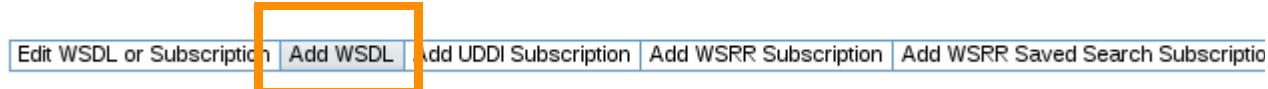


### Information

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.
  - \_\_\_ b. On the Configure Web Service Proxy page, click **Add**.
  - \_\_\_ c. For the Web Service Proxy Name, enter `AddressSearchProxy` and then click **Create Web Service Proxy**.
- \_\_\_ 3. Add the `EastAddressSearch` 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



- \_\_\_ b. Select `local:///` for the WSDL File URL.

- \_\_\_ c. Select **EastAddressSearch.wsdl** from the list.

**WSDL File URL**

local:///	<input type="button" value=""/>
EastAddressSearch.wsdl	<input type="button" value=""/>

**Use WS-Policy References**

on  off

**WS-Policy Parameter Set**

(none)

**WS-Policy Enforcement Mode**

Enforce

**SLA Enforcement Mode**

Allow

**Note**

This WSDL was loaded in Exercise 1. If it is not in local:///, use the **Upload** button to retrieve it from the <lab\_files>/setup directory.

- \_\_\_ d. Click **Next**.
- \_\_\_ 4. Create a front side handler to accept HTTP requests for the web service proxy.
- \_\_\_ a. Locate the entry for **AddressSearchService - AddressSearch** (it might be the only entry listed).
- \_\_\_ b. Click **+** (new button) beside the **Local Endpoint Handler** list.

**Web Service Proxy WSDLs**

AddressSearchService - AddressSearch

Local	
Local Endpoint Handler	URI
(none) <input type="button" value=""/> <input style="outline: 2px solid orange;" type="button" value="+"/> <input type="button" value="..."/>	/EastAddress/services/AddressSearch

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

- \_\_\_ d. Configure the new HTTP front side handler with the following values. Leave all other settings to the default values.
- **Name:** AddressSearchFSH
  - **Local IP Address:** Host alias of `dp_public_ip`
  - **Port Number:** `<wsp_proxy_port>`

#### HTTP Front Side Handler

<input type="button" value="Apply"/>	<input type="button" value="Cancel"/>
<hr/>	
<b>Name</b>	<input type="text" value="AddressSearchFSH"/> *
<hr/>	
<b>Administrative State</b>	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
<hr/>	
<b>Comments</b>	<input type="text"/>
<hr/>	
<b>Local IP Address</b>	<input type="text" value="dp_public_ip"/> <input type="button" value="Select Alias"/>
<hr/>	
<b>Port Number</b>	<input type="text" value="8975"/> *
<hr/>	
<b>HTTP Version to Client</b>	<input type="button" value="HTTP 1.1"/>

- \_\_\_ e. Click **Apply** to save the changes that are made to the HTTP front side handler.
- \_\_\_ 5. Complete the AddressSearch WSDL entry by changing the local URI to `/EastAddressSearch`.
- \_\_\_ a. In the **AddressSearchService – AddressSearch** WSDL entry, change the local (inbound) URI to: `/EastAddressSearch`

#### Web Service Proxy WSDLs

AddressSearchService - AddressSearch	
Local	
Local Endpoint Handler	URI
<input type="button" value="AddressSearchFSH"/>	
<input type="button" value="+"/>	
<input type="button" value="..."/>	
	<input type="text" value="/EastAddressSearch"/>

- \_\_\_ b. Click **Add** (green plus sign) in the Edit/Remove column to add it to the local endpoint handler list.

- \_\_\_ c. In the Remote (outbound) section, set the Remote Endpoint Host entry for the application server to: <backend\_server\_ip>

Remote			
Protocol	Remote Endpoint Host	Port	Remote URI
HTTP	treminelli.com	9080	/EastAddress/services/AddressSearch
Published	<input checked="" type="checkbox"/> Use Local		

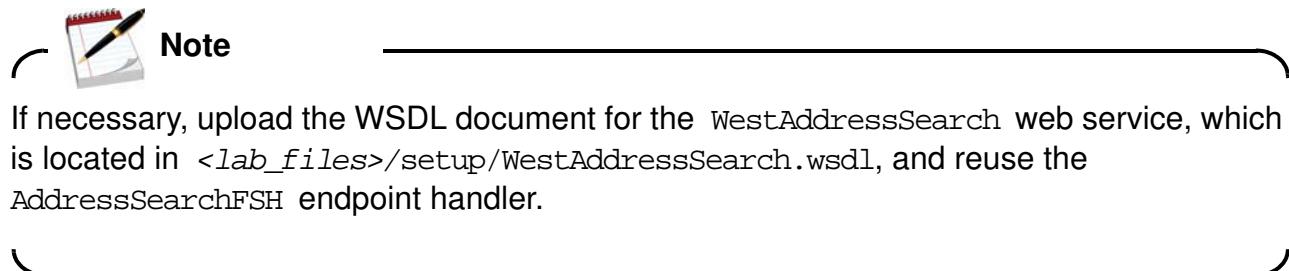
- \_\_\_ d. Set the Port value to: <was\_server\_port>  
\_\_\_ e. Click **Next** to continue the configuration.  
\_\_\_ f. Verify that the **WSDL entry** in the AddressSearchProxy has one active endpoint and that it is configured (1 up / 1 configured).

WSDL Source Location	Endpoint Handler Summary	WSDL Status
[+] local:///EastAddressSearch.wsdl	1 up / 1 configured	Okay

## 9.2. Add a WSDL to the web service proxy for the WestAddressSearch web service

Add another WSDL to the web service proxy to forward requests to the WestAddressSearch web service. This West Search service is configured in the same AddressSearchProxy that you created for the EastAddressSearch service.

- \_\_\_ 1. Repeat most of the steps from **Section 9.1, "Create a web service proxy for EastAddressSearch web service," on page 9-5** to add the WSDL file for the WestAddressSearch service within the existing AddressSearchProxy. You reuse the endpoint handler AddressSearchFSH that you created in the earlier section.



- \_\_\_ 2. Verify that you have both WS-Proxy WSDLs showing as 1 up / 1 configured.

### WSDLs

Edit WSDL or Subscription		
WSDL Source Location	Endpoint Handler Summary	WSDL Status
[+] local:///EastAddressSearch.wsdl	1 up / 1 configured	Okay
[+] local:///WestAddressSearch.wsdl	1 up / 1 configured	Okay

- \_\_\_ 3. Click **Save Config**.

## 9.3. Verify the generated components

In the last two sections, you used two WSDL documents for both the East and West address search web services. The DataPower appliance creates several components as a result of this action. In this section, you examine the components that the appliance creates.

- \_\_\_ 1. Examine the components in the **WSDL files** tab of the web service proxy web page.
  - \_\_\_ a. Expand both `local:///EastAddressSearch.wsdl` and `local:///WestAddressSearch.wsdl` to examine the local and remote proxy settings.
  - \_\_\_ b. Under **Local** for each WSDL file, you can see the URI and endpoint handlers.



### Information

The **URI** field specifies the URI for the web service. In addition, each local service has a local endpoint handler called `AddressSearchFSH`.

- \_\_\_ c. Click **Edit**, and then [...] (edit button) beside the local endpoint handler to open it.
- \_\_\_ d. Click **Cancel** to close the window.
- \_\_\_ e. Under **Remote**, you see the actual endpoint (URI) of the 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 that is listening for requests and various HTTP options for the HTTP connection.

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



### Information

The combination of the DataPower appliance host name + proxy port number + URI is used to call a service on a web service proxy. Take, for example, the following values:

- DataPower appliance host name: `dpedu.ibm.com`
- Proxy port number: 1234
- URI: `/WestAddressSearch`

If you used these values, you would invoke the `WestAddressSearch` web service with the URL (assuming that there is no SSL):

`http://dpedu.ibm.com:1234/WestAddressSearch`

2. Click the **Services** tab of the web service proxy. Verify that you see the services.

### Services

WSDL Name: EastAddressSearch.wsdl		
Service	Address Search Service	Publish to UDDI
WSDL Name: WestAddressSearch.wsdl		
Service	Address Search Service	Publish to UDDI



### Note

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

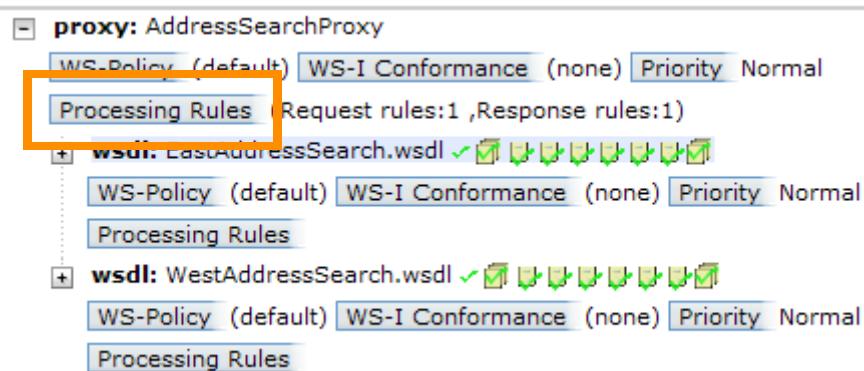
```
<wsdl:service name="AddressSearchService">
<wsdl:port binding="impl:AddressSearchSoapBinding"
name="AddressSearch">
<wsdlsoap:address
location="http://training.ibm.com:9080/EastAddress/services/AddressSe
arch"/>
</wsdl:port>
</wsdl:service>
```

Notice that the `wsdl:service name` inside the `EastAddressSearch.wsdl` 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 URL, user ID, and password. The DataPower appliance itself does not contain a UDDI registry.

3. Examine the **Policy** tab to view the Address Search proxy policy.
- \_\_ a. Click the **Policy** tab.
  - \_\_ b. Verify that you see the default request and response rules under the web service proxy policy.

- \_\_\_ c. Under AddressSearchProxy, click **Processing Rules**. A policy editor section opens beneath the WSDL policy tree.



The processing rules and actions to perform against requests and responses, and for error conditions, are displayed.



### Information

When you create a web service proxy, the appliance generates proxy-level request and response rules. The proxy-level request rule contains two actions: service level monitoring (SLM) and results.

- \_\_\_ d. Hover your mouse pointer over the SLM action. Notice that the SLM policy has the value `AddressSearchProxy`. This policy is defined in the **SLM** tab. The **SLM** tab is used to configure service monitoring.
- \_\_\_ e. Hover your mouse pointer over the **Results** action. This action returns the results to the client.



- \_\_\_ 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, click + to expand the hierarchical view until the ports are 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 performed and the publishing of the WSDL document to the web service proxy.

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

- wsdl: EastAddressSearch.wsdl ✓
- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules
- service: {http://east.address.training.ibm.com}AddressSearchService ✓
- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules
- port: {http://east.address.training.ibm.com}AddressSearch ✓
- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules
- port-operation: findByLocation ✓
- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules
- port-operation: findByName ✓
- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules
- port-operation: retrieveAll ✓
- WS-Policy (default) WS-I Conformance (none) Priority Normal
- Processing Rules

- \_\_ h. Click any of the icons to see the options.

The screenshot shows the configuration interface for a web service proxy. At the top, it displays the port information: `port {http://east.address.training.ibm.com}AddressSearch` with WS-Policy (default) set to WS-I Conformance (none) and Priority Normal. Below this, there are three sections for operations: `port-operation: findByLocation`, `port-operation: findByName`, and `port-operation: retrieveAll`. Each section has a "Processing Rules" link. On the right, a detailed view of the processing rules is shown for the first operation. It includes a table with columns "Effective Value" and "Local Value". The "Effective Value" column contains icons: a green checkmark for enabled components, a red X for disabled components, and a yellow question mark for schema validation. The "Local Value" column lists eight configuration options, each with a checked checkbox. A "Done" button is at the bottom right of the panel.

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 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. Click **Close**.
- \_\_ j. 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.

## 9.4. Test the EastAddressSearch web service

The `AddressSearchProxy` web service proxy is active and forwarding web service requests to the `EastAddressSearch` web service. Verify that the **findByLocation** operation properly retrieves the entire list of addresses from the service.



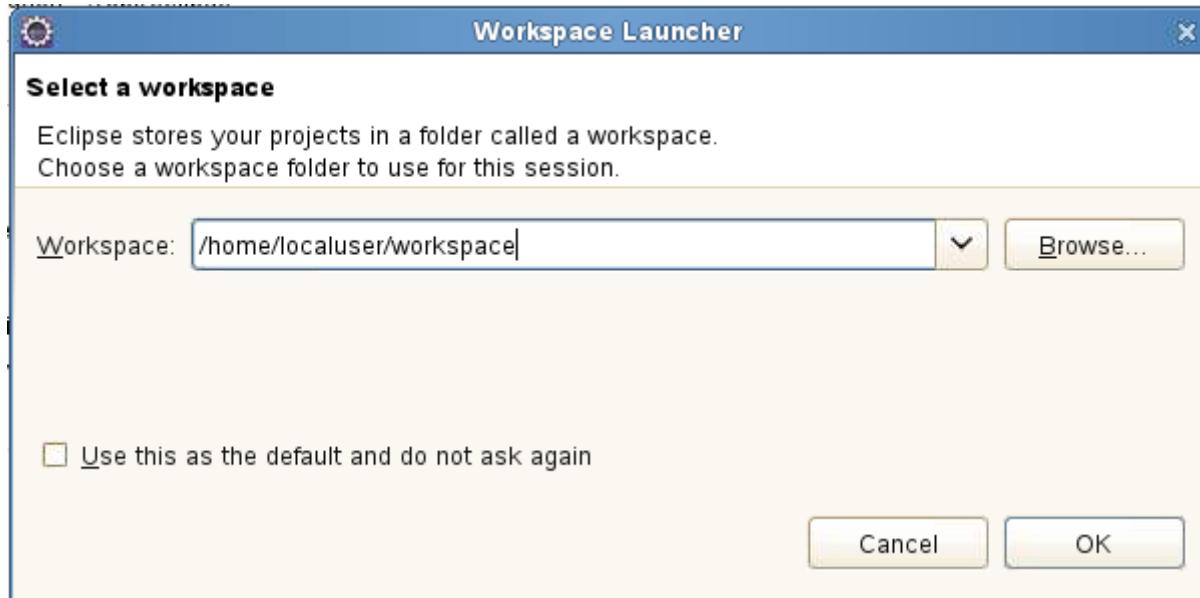
### Information

You use the Web Services Explorer in Eclipse to test the address web service proxy. You can still use cURL to test the web service by opening a Terminal window in the `<lab_files>/WSSecurity` subdirectory and running:

```
curl -H "SOAPAction: \"\" " --data-binary @findByLocation.xml
http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch
```

The input SOAP request is `findByLocation.xml`, and it must be in the current directory.

- 1. Load the Web Services Explorer from the Eclipse workbench to access the `EastAddressSearch` web service.
  - a. From the Linux desktop, open **Eclipse** to the `/home/localuser/workspace` directory. You see the project from the earlier exercise, and the `AddressClient` project that contains the WSDL files you imported earlier.

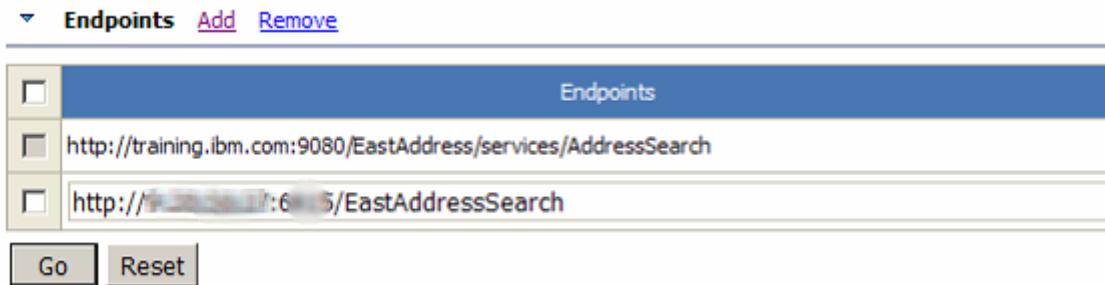


- b. Locate the `EastAddressSearch.wsdl` file in the workspace.
- c. If you are not in the Java EE perspective, switch to it now by clicking **Window > Open Perspective > Other** and then selecting **Java EE**.

- \_\_\_ d. Open the menu for `EastAddressSearch.wsdl` by right-clicking the selection. Select **Web Services > Test with Web Services Explorer**.



- \_\_\_ 2. Add an endpoint to send web service requests through the newly created web service proxy.
- \_\_\_ a. In the Web Services Explorer, locate the list of web service endpoint addresses in the Actions pane.
- \_\_\_ b. Click **Add** in the Endpoints list.



- \_\_\_ c. Enter `http://<dp_public_ip>:<wsp_proxy_port>/EmailAddressSearch`, the endpoint address for the WSDL entry, in the `AddressSearchProxy`.  
Recall from **section 3** how to construct the web service proxy URI.



### Important

The release level of Eclipse running in the image has a bug in the Web Service Explorer: maximizing a view might cause editable fields in the view to become non-editable. To fix the problem, restore the view or perspective to its regular size.

- \_\_\_ d. Click **Go** to accept the new endpoint. In the Status pane, you see a message that the endpoints were successfully updated.
- \_\_\_ 3. Execute the **findByLocation** operation on the newly added web service endpoint.
- \_\_\_ a. In the endpoints list, select the check box next to the newly created endpoint.

- \_\_\_ b. In the operations list, click **findByLocation**.

Name
<a href="#">findByName</a>
<a href="#">retrieveAll</a>
<a href="#">findByLocation</a>

- \_\_\_ c. In the Invoke a WSDL Operation page, make sure that the new endpoint is selected.
- \_\_\_ d. Enter **NY** for the **state** (not in the **city** field).
- \_\_\_ e. Click **Go** to execute the **findByLocation** web service operation.
- \_\_\_ f. In the Status pane directly below the Actions pane, confirm that the Web Services Explorer received a response from the web service, with an address of `<state>NY</state>`.

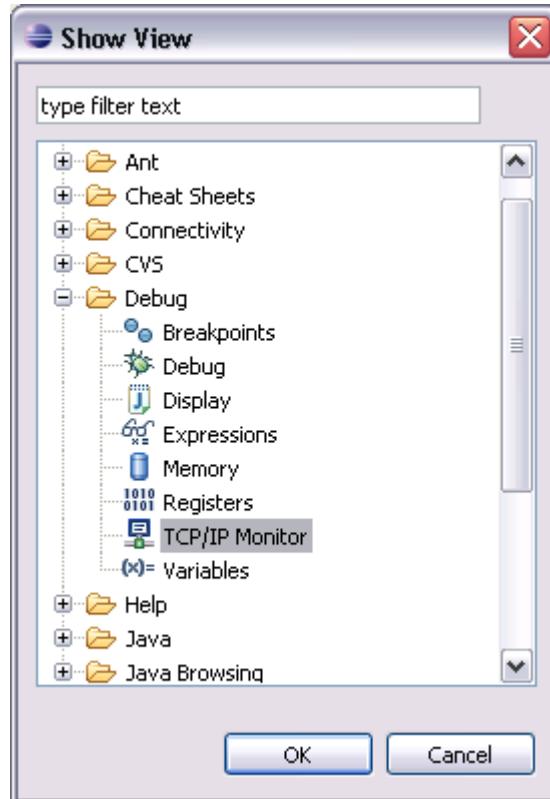


### Note

If you get an internal error on the request, verify that the URI specified in the endpoint is `/EastAddressSearch`, rather than `/EastAddress/services/AddressSearch`. `/EastAddressSearch` is what the web service proxy is looking for.

- \_\_\_ g. Click **Source** to view the SOAP message that the service returned.
- \_\_\_ 4. Use the TCP/IP monitor to capture and view the messages that are passed between the client and DataPower appliance.
- \_\_\_ a. Open the TCP/IP monitor view by selecting **Window > Show View > Other**.

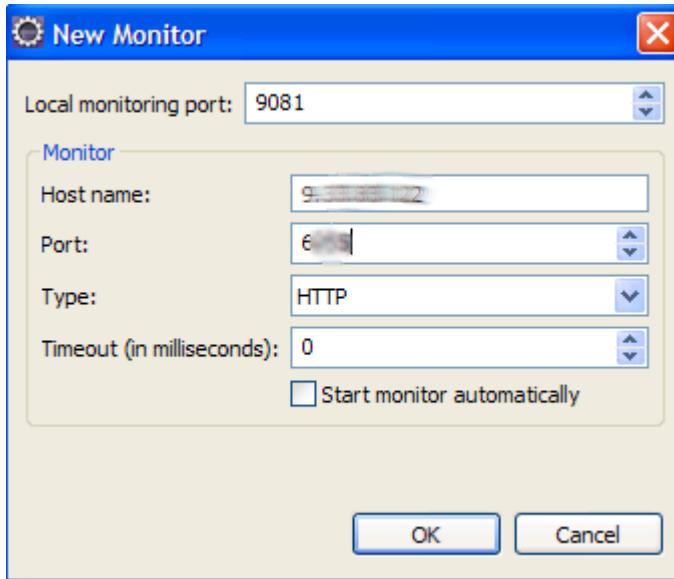
- \_\_\_ b. In the Show View window, expand **Debug** and select **TCP/IP Monitor**.



- \_\_\_ c. Click **OK**. The TCP/IP Monitor view opens.  
\_\_\_ d. Right-click inside the TCP/IP Monitor view and select **Properties**. This action opens the TCP/IP Monitor Preference page.

- \_\_\_ e. Click **Add** to create a TCP/IP monitor. Enter the following values:

- **Local monitoring port:** 9081
- **Host name:** <dp\_public\_ip>
- **Port:** <wsp\_proxy\_port>



This TCP/IP monitor listens to requests on the localhost (127.0.0.1) and port 9081 and forwards the request to the host name <dp\_public\_ip> and port <wsp\_proxy\_port>.

- \_\_\_ f. Click **OK** to close the New Monitor window and add an entry to the TCP/IP monitors list.
- \_\_\_ g. Click **Start** to start the TCP/IP monitor.
- \_\_\_ h. Click **OK** to close the preference page.
- \_\_\_ 5. Test the `EastAddressSearch` web service by submitting the request to the TCP/IP monitor.
- \_\_\_ a. In the Web Services Explorer, add an endpoint address:  
`http://127.0.0.1:9081/EastAddressSearch`
- \_\_\_ b. Invoke the operation **findByLocation**.
- \_\_\_ c. Verify that the TCP/IP monitor is populated with the request and response.



#### Note

If you get a response in the Status pane, but no entries in the TCP monitor, verify that the endpoint selected for the operation is the TCP monitor, and not the proxy.

## 9.5. Test the WestAddressSearch web service

Repeat the steps from **Section 9.4, "Test the EastAddressSearch web service," on page 9-15** to test the `WestAddressSearch` web service.

- \_\_\_ 1. Remember that the West Address Search web service is at `http://<dp_public_ip>:<wsp_proxy_port>/WestAddressSearch`.
- \_\_\_ 2. In your test, invoke the **findByLocation** operation with the state `CA`. You get a response that contains addresses for `<state>CA</state>`.



### Note

If you get an internal error response, make sure that you are initiating the operation from the `WestAddressSearch.wsdl` file in the Web Services Explorer, not the previous `EastAddressSearch.wsdl` file.

- \_\_\_ 3. In your final test, invoke the **findByLocation** operation with the state `NY`. This test causes the `WestAddressSearch` web service to generate a SOAP fault, since `NY` is outside the west address book.
  - \_\_\_ a. Make sure that the TCP/IP monitor is started and configured based on the instructions in **Section 9.4, "Test the EastAddressSearch web service"**.
  - \_\_\_ b. In the Web Services Explorer, verify that you added an endpoint address with the value `http://127.0.0.1:9081/WestAddressSearch`.
  - \_\_\_ c. Invoke the operation **findByLocation** with the state `NY`, and click **Go**.
- \_\_\_ 4. Examine the TCP/IP monitor for the `WestAddressSearch` request and response.
  - \_\_\_ a. In the TCP/IP monitor view, verify that you see an entry for the West Address request and response.
  - \_\_\_ b. View the message header. Click the down arrow at the upper right of the TCP/IP Monitor view (third from the right).
  - \_\_\_ c. Select **Show Header**.
  - \_\_\_ d. Under the request, you see the `WestAddressSearch` SOAP request.

```

Request: localhost:9081
Size: 374 (681) bytes
Byte ▾
POST /WestAddressSearch HTTP/1.1
Host: dpedu1:6775
Content-Type: text/xml; charset=utf-8
Content-Length: 374
Accept: application/soap+xml, application/dime, application/mshta, */*

```

You can use the list on the right side to format the request (byte, image, XML, and web browser).

- \_\_\_ e. Under the response, you see the following WestAddressSearch SOAP response.

The screenshot shows a window titled "Response: dpedu1:6775" with a size of "229 (327) bytes". A dropdown menu labeled "Byte" is open. The main content area displays the following XML response:

```

HTTP/1.0 500 Error
X-Backside-Transport: FAIL FAIL
Content-Type: text/xml
Connection: close

```

Notice that the WestAddressSearch web service generates a fault message that indicates that an address cannot be found.

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
<soapenv:Header/>
<soapenv:Body>
<soapenv:Fault>
<faultcode>AddressNotFoundException</faultcode>
<faultstring>AddressNotFoundException: Cannot find any address
entries located in NY.
</faultstring>
<detail>
<AddressNotFoundException>
<message>Cannot find any address entries located in NY.
</message>
</AddressNotFoundException>
</detail>
</soapenv:Fault>
</soapenv:Body>

```

- \_\_\_ f. The address search WSDL document defines the fault message to return when an address cannot be found.

```

<wsdl:fault message="impl:AddressNotFoundException"
name="AddressNotFoundException"/>

```

## 9.6. Add an operation-level rule to West Address Search web service proxy

In this section, you configure a proxy policy for all operations for the West Address Search web service (that is, an operation-level policy). The proxy policy consists of a Match action to match a set of URLs and a filter action that validates the title field of the input message to check for correct values.

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



### Note

Recall that the appliance generates a default proxy-level request and response policy. You can override these policies at a fine-grained level. You override the default proxy-level request policy with the `WestAddressSearch` operation-level policy.

- b. In the WSDL policy tree view, expand the plus sign (+) and locate the `findByName` operation for `WestAddressSearch`.

- c. Click **Processing Rules** underneath **port-operation: findByName**. The policy configuration area automatically reloads, and the policy editor displays.

- \_\_\_ d. Under Policy Configuration, click **New Rule**.
- \_\_\_ e. Enter a rule name (for example, `findByName_West_rule1`).

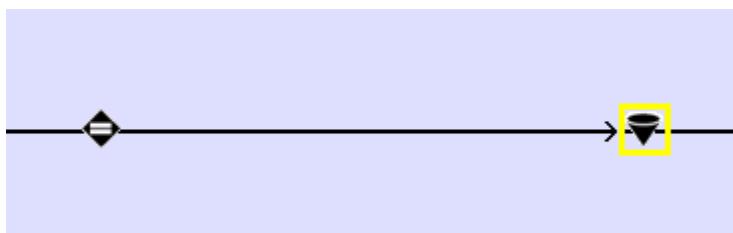
On the policy rule line, you see an automatically generated **Match** action.



#### Note

The **Match** action that is generated is configured with a matching rule that matches all URLs of incoming messages for this proxy.

- \_\_\_ 2. Add a **Filter** action to accept or reject the incoming message for invalid title values.
  - \_\_\_ a. Drag the filter icon after the **Match** action.



- \_\_\_ b. Double-click the **Filter** action to open the Configure Filter Action page. This web page allows you to specify an XSL style sheet for your **Filter** action.
- \_\_\_ c. Click **Upload** to upload an XSL style sheet that checks for valid title values in the incoming message.
- \_\_\_ d. In the File Management web page, click **Browse** and navigate to:  
`<lab_files>/WSProxy/Address-filter.xsl`
- \_\_\_ e. Select the file, click **Upload**, and then **Continue**.



#### Note

The `Address-filter.xsl` style sheet checks the contents of `<title>`. If the valid values of `mr`, `mrs`, `miss`, `ms`, or `dr` (case ignored) are found, the message is accepted (`<dp:accept>` extension function). Otherwise, the message is rejected (`<dp:xreject>`).

- \_\_\_ f. Click **Done** to close the Configure Filter Action page.
- \_\_\_ g. On the Policy Configuration page, for Rule Direction, select **Client to Server** from the list.
- \_\_\_ h. Click **Apply** at the top of the page to save the policy. This action also generates a **Results** action that sends the response message to the client.

3. Test the `WestAddressSearch` web service operation-level proxy policy.
- \_\_ a. Enable the probe so that you can view the actions that were executed. Click **Show Probe**. The multi-step probe window opens for the `AddressSearchProxy`.
  - \_\_ b. Click **Enable Probe**. When DataPower confirms you completed the action successfully, click **Close**.
  - \_\_ c. Switch back to Eclipse and use the Web Services Explorer to send a message to the **findByName** operation of the `WestAddressSearch` web service.
  - \_\_ d. Enter the following values:
    - **firstName**: Robin
    - **lastName**: Price
    - **title**: Mrs

**findByName**

**name**

**firstName** string  
Robin

**lastName** string  
Price

**title** string  
Mrs

**Go** **Reset**

- \_\_ e. Click **Go**.
- \_\_ f. If you submitted your message to the TCP/IP monitor, make sure that you get the correct response (Robin Price, Fort Worth, TX.)
- \_\_ g. Test the **findByName** operation again, except enter the title string `Mrss` instead of `Mrs`. The DataPower appliance generates a SOAP fault.

Response: dpedu1:6995  
Size: 232 (330) bytes

Byte

HTTP/1.0 500 Error  
Content-Type: text/xml  
X-Backside-Transport: FAIL FAIL  
Connection: close

- \_\_ h. Switch back to the multi-step probe window and click **Refresh** to view the list of transactions.

- \_\_\_ i. Click + next to one of the transactions. A transaction has both a request and a response.
- \_\_\_ j. Select the magnifying glass next to the **request** type of the same transaction. Another web page opens, showing the actions that are executed from this request.

Notice that the actions shown are similar to the actions configured for the WestAddressSearch **findByName** proxy policy: a filter and a results action. Use the **Next** and **Previous** arrows to move through the steps in the actions. You can examine the contents of the contexts and variables.

#### Input Context 'INPUT' of Step 1



Step 1: Filter Action: Input=INPUT, Transform=local:///Address-filter.xsl, OutputType=default, SOAPValidation=body

- \_\_\_ k. Close the request transaction web page.
- \_\_\_ l. Back on the list of transactions, select the magnifying glass next to the **response** type of the same transaction.

Another web page opens, showing the action that is executed from this response: a results action that moves the input to the output.



#### Note

You did not code a response rule for the **findByName** operation. So where did this response rule originate? Recall that there are default request and response rules that are created at the proxy level. Since the **findByName** operation had a request rule at the operation level, that request rule was executed. Since there was no response rule that is coded at the operation, port, service, or WSDL level, the default response rule at the proxy level was executed. If you click **Processing Rules** under the **AddressSearchProxy**, the Policy Configuration for it opens, displaying a request and a reply rule.

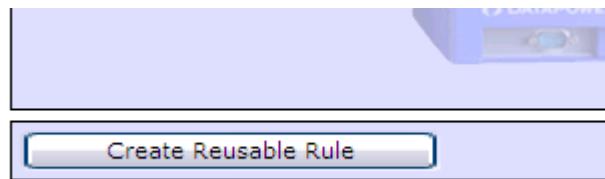
Configured Rules			
Order	Rule Name	Direction	Actions
↑ ↓	AddressSearchProxy_default_request-rule	Client to Server	
↑ ↓	AddressSearchProxy_default_response-rule	Server to Client	

- \_\_\_ m. Close the response transaction web page.

- \_\_\_ 6. Click the magnifying glass next to the transaction without a + (plus sign), which is red. This transaction is a request only. It corresponds to the transaction with the invalid <title> contents.
- \_\_\_ 7. A page opens with a single **Filter** action displayed, and no **Results** action. You see the invalid `Mrss` in the <title> element.
- \_\_\_ 8. Use the **Next** arrow to move to the next step in the processing. Notice that the header text indicates: Transaction aborted in Step 1. Hence, there is no response.
- \_\_\_ 9. Select the **Service Variables** tab and look for  
`var://service/formatted-error-message`. The value is Incomplete input (from client), which is the same response you get in your TCP monitor.
- \_\_\_ 10. Close the request transaction web page.
- \_\_\_ 11. Click **Disable Probe**, and then click **Close**.
- \_\_\_ 12. On the multi-step probe window, click **Close**.
- \_\_\_ 13. Click **Save Config** to save your configuration to the appliance.
- \_\_\_ 14. Log out from the WebGUI when you are finished.
- \_\_\_ 15. Close Eclipse.



Remember, only one policy can be executed for a request or response. However, you can reuse a rule or reuse actions within a rule by creating a reusable rule in the policy configuration. As soon as you create the reusable rule, it can be dragged from that rule into the rule configuration area for other rules.



## End of exercise

## Exercise review and wrap-up

In this exercise, you created a web service proxy for both the East and West Address Search web service. Using the Web Services Explorer in Eclipse, you tested the web service proxy. The system logs were used to verify the requests and responses to the DataPower appliance.

Finally, you added an operation-level policy to the **findByName** operation and tested it using the Web Services Explorer in Eclipse.



# Exercise 10. Web service encryption and digital signatures

## What this exercise is about

In this exercise, you learn how to perform web services security functions by using the IBM WebSphere DataPower SOA Appliance. The DataPower appliance supports security-related tasks that both a client and a server must perform. You play the role of a client by using a multi-protocol gateway to generate an encrypted and signed message. You play the role of the server by decrypting and verifying the digital signature of the message on the web service proxy.

## What you should be able to do

At the end of the exercise, you should be able to:

- Create a multi-protocol gateway to generate a message with XML encryption
- Create a multi-protocol gateway to generate a message with an XML digital signature
- Perform field-level encryption and decryption on XML messages
- Create a rule to decrypt messages and verify digital signatures that are contained in a message within a web service proxy policy

## Introduction

In this exercise, you play two roles, the client and server.

In a typical scenario that involves XML encryption, a client sends a message encrypted using the server certificate to the server (the DataPower appliance). The server, such as the DataPower appliance, uses the associated private key to decrypt the message. The private key is kept confidential by the server, and the certificate is publicly available.

Since the lab environment does not include a client runtime to generate a message with XML encryption, you create a multi-protocol gateway to generate an encrypted message. This message is then decrypted on the web service proxy.

Similarly, you use a multi-protocol gateway to create a signed message that is verified on the web service proxy.

Two crypto objects, **AliceKey** and **AliceCert**, are used for the security tasks. The **AliceKey** is the private key and the **AliceCert** is the certificate.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- The **Address Search** web service that is running on WebSphere Application Server
- Access to the `<lab_files>` directory
- 
- 
-

# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the completion of “Exercise 9: Configure a web service proxy service.”
- 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
  - *<dp\_public\_ip>*: IP address of the public services on the appliance
  - *<wsp\_proxy\_port>*: port number of the web service proxy that was created in the previous exercise
  - *<mpgw\_crypto\_port>*: port number of the multi-protocol gateway that encrypts and signs messages

## 10.1.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. These same key and certificate objects are used for signing and validating messages.

- \_\_\_ 1. Log on to the IBM WebSphere DataPower SOA Appliance WebGUI.
- \_\_\_ 2. Create a crypto key object called `AliceKey`.
- \_\_\_ a. Select **Objects > Crypto Configuration > Crypto Key**.
- \_\_\_ b. Click **Add** to create a crypto key object.
- \_\_\_ c. In the Crypto Key web page, enter the following information and leave the remaining fields with their default values:
  - **Name:** `AliceKey`
  - **File name:** `cert:/// Alice-privkey.pem`

Name  \*

Administrative State  enabled  disabled

File Name

- \_\_\_ d. Click **Apply**.



### Information

The `cert: Alice-privkey.pem` file is already uploaded to the appliance. If you do not see this file in the file name list, notify your instructor. The steps to execute this task can be found in **Exercise 1**.

- \_\_\_ e. Verify that the `AliceKey` object operational state is **up**.

Crypto Key: AliceKey [up]



## Information

The `AliceKey` object provides a reference to the private key file that the DataPower appliance uses to decrypt messages that were encrypted by using the associated certificate. This associated certificate is set up in the next step.

- \_\_\_ 3. Create a certificate object called `AliceCert`.
  - \_\_\_ a. Select **Objects > Crypto Configuration > Crypto Certificate**.
  - \_\_\_ b. Scroll down and click **Add** to create a crypto certificate object.
  - \_\_\_ c. In the Crypto Certificate web page, enter the name `AliceCert` and select the `cert:/// Alice-sscert.pem` file. Leave the remaining fields with their default values:

The screenshot shows a configuration interface for a crypto certificate. At the top, there is a 'Name' field containing 'AliceCert'. Below it, an 'Administrative State' section has two radio buttons: 'enabled' (selected) and 'disabled'. Underneath, a 'File Name' section shows a dropdown menu with 'cert:///'. A sub-menu is open, displaying 'Alice-sscert.pem' with a red box highlighting it. To the right of the dropdown are 'Details...' and a close button.

- \_\_\_ d. Click **Apply**.
- \_\_\_ e. Verify that the `AliceCert` operational status is **up**.



## Information

The `AliceCert` object provides a reference to the certificate that is used to encrypt and sign messages that are sent to the DataPower appliance. For simplicity, you send a plaintext message to the DataPower appliance to be encrypted, which the DataPower appliance echoes back. You use the echoed message to test message decryption on the DataPower appliance.

The `AliceCert` and `AliceKey` objects are used throughout this exercise for security-related tasks.

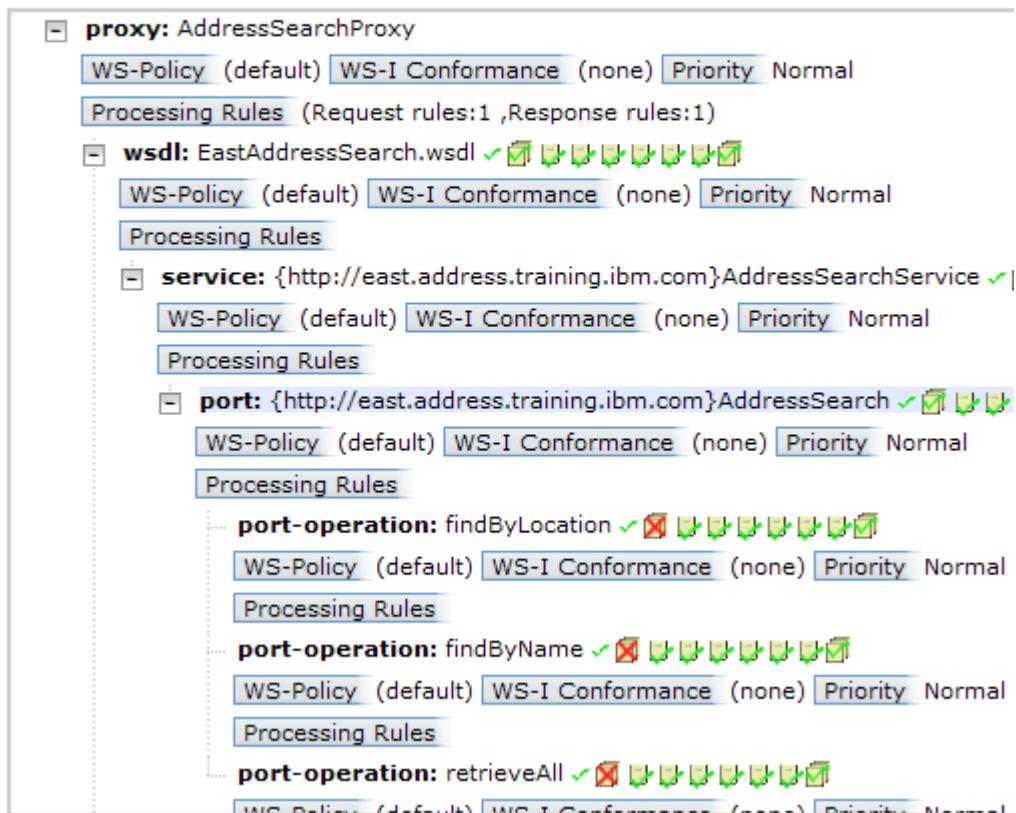
- \_\_\_ f. Click **Save Config**.

## 10.2.Examine the East Address Search web service proxy

- \_\_\_ 1. On the Control Panel, click the **Web Service Proxy** icon.
- \_\_\_ 2. In the Configure Web Service Proxy page, click **AddressSearchProxy**.
- \_\_\_ 3. Click the **Policy** tab.
- \_\_\_ 4. Click + (plus sign) to expand **wsdl:** EastAddressSearch.wsdl > **service:** AddressSearchService > **port:** AddressSearch.

### WSDL Policy Tree Representation

Define the policies to apply in the tree.



### Note

Recall that you can add rules at many levels of the web service proxy. Only one set of rules is executed per request or response.

In a later section, you create a rule to *decrypt* the message payload for the findByLocation operation, and a rule to *validate* the message payload for the findByLocation operation.

Similarly, you create a rule to *decrypt* messages for the findByName operation, except that you use a document crypto map to decrypt specific elements that are contained within the message.

## 10.3.Create a multi-protocol gateway to encrypt messages

In this section, you create a multi-protocol gateway to encrypt a plaintext message that is later decrypted by using the web service proxy that you created in the previous exercise.

In a real world scenario that involves web services security, a client that sends a request to a web service needs access to a web services security runtime that performs security functions, such as WebSphere Application Server. You would use WebSphere Application Server to generate XML encryption on the message and send it to the underlying web service.

The lab environment is not configured to use WebSphere Application Server to generate XML encryption. Instead, you simulate these tasks by using encryption functions within the DataPower appliance.

The multi-protocol gateway that you create to encrypt messages can be thought of as your web services security client runtime. You use it to generate and save an encrypted message.

- \_\_\_ 1. Create a multi-protocol gateway to apply XML encryption to a message.
  - \_\_\_ a. In the WebGUI, click **Control Panel > Multi-Protocol Gateway**.
  - \_\_\_ b. In the Configure Multi-Protocol Gateway catalog list page, click **Add**.
  - \_\_\_ c. In the Configure Multi-Protocol Gateway page, enter a Multi-Protocol Gateway Name of **CryptoMPGW**.
  - \_\_\_ d. Select **dynamic-backend** for the **Type**.



### Information

All of the functions that are needed in this service happen in request rule processing, so no response rules are needed. Since there is no loopback option for multi-protocol gateways in the request rule, you set a DataPower variable that tells the service that no response processing is needed. As part of this technique, you must set the service type as **dynamic-backend**.

- \_\_\_ e. Retain **SOAP** as the Response Type and the Request Type.
- \_\_\_ f. In the Front Side Protocol section, click the **+** (New) to start the creation of the front side handler.
- \_\_\_ g. In the drop-down list, select **HTTP Front Side Handler**.
- \_\_\_ h. In the Configure HTTP Front Side Handler dialog, enter a Name of **CryptoMPGW\_HTTP\_FSH**.
- \_\_\_ i. For the Local IP Address field, select a host alias of **dp\_public\_ip**.

- \_\_\_ j. Set the Port Number to: <mpgw\_crypto\_port>

HTTP Front Side Handler

Name

CryptoMPGW\_HTTP\_FSH \*

Administrative State

enabled  disabled

Comments

Local IP Address

dp\_public\_ip

Port Number

7973 \*

HTTP Version to Client

HTTP 1.1

- \_\_\_ k. Click **Apply**. The dialog closes, and the new CryptoMPGW\_HTTP\_HSH handler displays in the Front Side Protocol list.

- \_\_\_ 2. Create the processing policy to encrypt the message.

- \_\_\_ a. Click + (New) in the Multi-Protocol Gateway Policy section to create a processing policy.
- \_\_\_ b. In the policy editor window, enter a Policy Name of **CryptoMPGW\_Policy**, and click **Apply Policy**.
- \_\_\_ c. Click **New Rule**.
- \_\_\_ d. Set the rule direction to **Client to Server**.
- \_\_\_ e. Double-click the highlighted **Match** action.
- \_\_\_ f. Create a Matching rule named `match_url_encrypt` that matches on a URL of `/encrypt`.
- \_\_\_ g. Click **Done** to complete the Match action configuration.
- \_\_\_ h. In the policy editor, drag the **Advanced** action to the rule configuration path.
- \_\_\_ i. Double-click the icon to configure this action.
- \_\_\_ j. Select **Set Variable** and click **Next**.
- \_\_\_ k. The advanced action page becomes a Configure Set Variable Action page.
- \_\_\_ l. For the Variable Name field, use the **Var Builder** to select the service variable **service/mpgw/skip-backside**.

- \_\_\_ m. For the Variable Assignment field, set a value of **1**.
  - \_\_\_ n. Click **Done**. This setting tells the DataPower service that no response rule processing is to occur. This variable is the DataPower variable that allows a multi-protocol gateway to emulate a service type of Loopback.
  - \_\_\_ o. In the policy editor, drag an **Encrypt** action to the right of the set variable action.
  - \_\_\_ p. Double-click the action icon.
  - \_\_\_ q. Since you are using WS-Security encryption on the complete SOAP message, retain **WSSec Encryption** as the Envelope Method and **SOAP Message** as the Message Type.
  - \_\_\_ r. For the recipient Certificate, select the **AliceCert** certificate you created previously.
  - \_\_\_ s. Click **Done** to complete this action.
  - \_\_\_ t. Hover the mouse over the encrypt action, and notice that the Output setting is **OUTPUT**. The results of the encryption are placed in the OUTPUT context, so no results action is needed.
  - \_\_\_ u. Click **Apply Policy**, and close the policy editor window.
- \_\_\_ 3. Complete the gateway service.
- \_\_\_ a. Click **Apply** to complete the gateway configuration.

Multi-Protocol Gateway status: [up]

### General Configuration

<b>Multi-Protocol Gateway Name</b>	<b>XML Manager</b>
<input type="text" value="CryptoMPGW"/> *	<input type="text" value="default"/> *
<b>Summary</b>	<b>Multi-Protocol Gateway Policy</b>
<input type="text"/>	<input type="text" value="CryptoMPGW_Policy"/> *
<b>Type</b>	<b>URL Rewrite Policy</b>
<input checked="" type="radio"/> dynamic-backend <input type="radio"/> static-backend <small>*</small>	<input type="text" value="(none)"/>

#### Back side settings

With a dynamic proxy back end type, the back end server address and port are determined by a stylesheet in a policy action.

#### Front side settings

<b>Front Side Protocol</b>
<input type="text" value="CryptoMPGW_HTTP_FSH (HTTP Front Side Handler)"/>

- \_\_\_ b. Click **Save Config**.

- \_\_\_ 4. Use cURL to test the CryptoMPGW multi-protocol gateway.
- \_\_\_ a. Open a Terminal window and navigate to the <lab\_files>/WSSecurity directory.
- \_\_\_ b. Enter the following command:
- ```
curl --data-binary @findByLocation.xml  
http://<dp_public_ip>:<mpgw_crypto_port>/encrypt >  
findByLocation-enc.xml
```
- The plain-text message body that is contained in the URI file is encrypted and saved in the findByLocation-enc.xml file. The > (greater-than symbol) is used to redirect output to a specific location rather than the console.
- \_\_\_ c. Open the findByLocation-enc.xml to view the results of the encrypted message. You send this file as an encrypted request later in this exercise.

**Hint**

You can use **gedit** to view the file, but the file contents display unformatted, which is difficult to read. To make it easier to read the XML, use Firefox instead (**File > Open File**).

- \_\_\_ 5. Create a rule in the CryptoMPGW multi-protocol gateway to apply field-level encryption.
- \_\_\_ a. In the Configure Multi-Protocol Gateway page, click [...] (Edit) beside the CryptoMPGW\_Policy firewall policy field.
- \_\_\_ b. In the policy editor window, click **New Rule** in the rule configuration area.
- \_\_\_ c. Set the Rule Direction to **Client to Server**.
- \_\_\_ d. Double-click the **Match** action icon to configure it.
- \_\_\_ e. Click **Add** to create a matching rule `match_url_encrypt_f1` that matches the URL `/encrypt_f1`.
- \_\_\_ f. Click **Done** to complete the Match action configuration.
- \_\_\_ g. As before, drag an **Advanced** action to the rule configuration path.
- \_\_\_ h. Double-click it, select **Set Variable**, and click **Next**.
- \_\_\_ i. Set the Variable Name field to the service variable `service/mpgw/skip-backside`.
- \_\_\_ j. For the Variable Assignment field, set a value of **1**.
- \_\_\_ k. Click **Done**.
- \_\_\_ l. Drag an **Encrypt** action after the **Set Variable** action.
- \_\_\_ m. Double-click the **Encrypt** action.

- \_\_\_ n. In the **Recipient Certificate** field, select **AliceCert**.
- \_\_\_ o. For **Message Type**, select **Selected Elements (Field-Level)**. The web page reloads, and shows a **Document Crypto Map** field.

**Encrypt**

|                                                                                                                                    |                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Envelope Method</b>                                                                                                             | <input checked="" type="radio"/> WSSec Encryption<br><input type="radio"/> Standard XML Encryption<br><input type="radio"/> Advanced<br>*                                               |
| <b>Message Type</b>                                                                                                                | <input checked="" type="radio"/> Selected Elements (Field-Level)<br><input type="radio"/> SOAP Message<br><input type="radio"/> Raw XML Document<br><input type="radio"/> Advanced<br>* |
| <b>Document Crypto Map</b>                                                                                                         | (none) <input type="button" value="+"/> ... *                                                                                                                                           |
| <b>Asynchronous</b>                                                                                                                | <input type="radio"/> on <input checked="" type="radio"/> off                                                                                                                           |
| {http://www.datapower.com/param/config}recipient <input type="button" value="AliceCert"/> <input checked="" type="checkbox"/> Save |                                                                                                                                                                                         |

- \_\_\_ p. Click the + (plus sign) to create a document crypto map.



### Information

The **document crypto map** is used to specify the elements to encrypt by using an XPath expression.

- \_\_\_ q. Enter the name: **Encrypto\_DCM**
- \_\_\_ r. In the **XPath Expression** field, click **XPath Tool**.
- \_\_\_ s. Click **Upload**, and then **Browse**.
- \_\_\_ t. Navigate to the `findByName.xml` file in `<lab_files>/WSSecurity` and click **Open**.
- \_\_\_ u. Click **Upload** and **Continue**.
- \_\_\_ v. Under **Namespace Handling**, select **local**. This setting generates a shorter XPath expression. Click **Refresh** if you do not see the contents of the `findByName.xml` file.

- \_\_\_ w. Click the <name> element to generate an XPath expression.

**XPath \***

`/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByName']/*[local-name()='name']`

---

**Content of sample XML file.**  
Click on an element, attribute name, or attribute value to select

- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:q0="http://east.address.training.ibm.com" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
- <SOAP-ENV:Body>
- <q0:findByName>
- <name>  
    <firstName>**Sarah**</firstName>  
    <lastName>**Chan**</lastName>  
    <title>**Mrs**</title>  
    </name>
- </q0:findByName>
- </SOAP-ENV:Body>
- </SOAP-ENV:Envelope>

- \_\_\_ x. Click Done.

You are back in the Configure Document Crypto Map web page. The XPath expression that you generated is populated in the **XPath Expression** field.

### Document Crypto Map

Name: Encrypto\_DCM \*

Administrative State:  enabled  disabled

Comments:

Operation: Encrypt (WS-Security) \*

XPath Expression:

```
/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByName']/*[local-name()='name']
```

Add XPath Tool \*

- \_\_\_ y. Click **Apply**.
- \_\_\_ z. Click **Done** in the Configure Encrypt Action page.
- \_\_\_ aa. Click **Apply Policy**. Close this window.
- \_\_\_ ab. Click **Apply** to apply the multi-protocol gateway changes.
- \_\_\_ 6. Use cURL to test the new rule.
  - \_\_\_ a. Open a Terminal window and navigate to the <lab\_files>/WSSecurity directory.
  - \_\_\_ b. Enter the following command:  
`curl --data-binary @findByName.xml  
http://<dp_public_ip>:<mpgw_crypto_port>/encrypt_f1 >  
findByName-enc.xml`

The <name> element that is contained in the `findByName.xml` file is encrypted and saved in the `findByName-enc.xml` file.
  - \_\_\_ c. Open the `findByName-enc.xml` to view the results of the encrypted message.



**Note**

Notice that the <name> element is replaced with an <EncryptedData> element. When you decrypt this message, you cannot use the same XPath expression in the document crypto since the <name> element does not exist. Think about the XPath expression that you must write for the field-level decrypt action.

## 10.4.Create a rule to sign messages

In a similar way as the previous section, you create a rule that signs messages.

The multi-protocol gateway that you create to sign messages can be thought of as your web services security client runtime. You use it to generate and save a signed message.

- \_\_\_ 1. Create a request rule to apply an XML signature to a message.
  - \_\_\_ a. In the Configure Multi-Protocol Gateway page, click [...] (Edit) beside the `CryptoMPGW_Policy` firewall policy field.
  - \_\_\_ b. In the policy editor window, click **New Rule** in the rule configuration area.
  - \_\_\_ c. Set the Rule Direction to **Client to Server**.
  - \_\_\_ d. Double-click the **Match** action icon to configure it.
  - \_\_\_ e. Click **Add** to create a matching rule **match\_url\_sign** that matches the URL `/sign`.
  - \_\_\_ f. Click **Done** to complete the Match action configuration.
  - \_\_\_ g. As before, drag an **Advanced** action to the rule configuration path.
  - \_\_\_ h. Double-click it, select **Set Variable**, and click **Next**.
  - \_\_\_ i. Set the Variable Name field to the service variable `service/mpgw/skip-backside`.
  - \_\_\_ j. For the Variable Assignment field, set a value of **1**.
  - \_\_\_ k. Click **Done**.
  - \_\_\_ l. Drag a **Sign** action after the **Set Variable** action.
  - \_\_\_ m. Double-click the **Sign** action.
  - \_\_\_ n. Retain **WSSec Method** as the Envelope Method, and **SOAP Message** as the Message Type.
  - \_\_\_ o. Select **AliceKey** for the Key and **AliceCert** for the Certificate.

- \_\_\_ p. Retain the other defaults.

| Sign                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Envelope Method</b>    | <input type="radio"/> Enveloped Method<br><input type="radio"/> Enveloping Method<br><input type="radio"/> SOAPSec Method<br><input checked="" type="radio"/> WSSec Method<br><input type="radio"/> Advanced<br><br>*                                                                                                                                                                                                                                                                                                                                               |
| <b>Message Type</b>       | <input checked="" type="radio"/> SOAP Message<br><input type="radio"/> SOAP With Attachments<br><input type="radio"/> Raw XML Document, including SAML for Enveloped<br><input type="radio"/> Selected Elements (Field-Level)<br><input type="radio"/> Advanced<br><br>*                                                                                                                                                                                                                                                                                            |
| <b>Asynchronous</b>       | <input type="radio"/> on <input checked="" type="radio"/> off<br><br><input checked="" type="radio"/> on <input type="radio"/> off<br><br><input type="checkbox"/> Save                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Use Asymmetric Key</b> | <b>Signing Algorithm</b> : rsa <input type="button" value="Save"/> <input type="checkbox"/> Save<br><b>Key</b> : AliceKey <input type="button" value="Save"/> <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/> Save<br><b>Certificate</b> : AliceCert <input type="button" value="Save"/> <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/> Save<br><b>WS-Security Version</b> : 1.0 <input type="button" value="Save"/> <input type="checkbox"/> Save |

- \_\_\_ q. Click **Done**.
- \_\_\_ r. Hover the mouse over the sign action, and notice that the Output setting is **OUTPUT**. The results of the signing are placed in the OUTPUT context, so no results action is needed.
- \_\_\_ s. Click **Apply Policy**, and close the policy editor window.
- \_\_\_ t. Click **Apply** to complete the gateway configuration.
- \_\_\_ 2. Use cURL to test the signing rule in the multi-protocol gateway.
- \_\_\_ a. Open a Terminal window and navigate to the `<lab_files>/WSSecurity` directory.
- \_\_\_ b. Enter the following command:

```
curl --data-binary @findByLocation.xml
http://<dp_public_ip>:<mpgw_crypto_port>/sign >
findByLocation-sign.xml
```

The message body that is contained in the `findByLocation.xml` file is signed and saved in the `findByLocation-sign.xml` file.

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

In this section, you perform the actions of the server (web service proxy). In the previous two sections, when you created a multi-protocol gateway, you were performing the actions of the client that would send encrypted and signed requests to the web service proxy.

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.

You next must decrypt the message payload for the **findByLocation** operation by specifying a proxy-wide crypto key. This decryption occurs **before** any validation or policy is executed.

- \_\_\_ 1. Examine the encrypted **findByLocation** message.



### Note

In the previous section, you used the `CryptoMPGW` multi-protocol gateway to create a message named `findByLocation-enc.xml` whose message body is encrypted. Open the file with an editor, such as `gedit`.

Notice that under the `<soapenv:Body>` tag, the operation name is encrypted.

```
<soapenv:Envelope>
  ...
  <soapenv:Body xmlns:q0="http://east.address.training.ibm.com"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <xenc:EncryptedData Id="body"
      Type="http://www.w3.org/2001/04/xmlenc#Content">
      ...
      </xenc:EncryptedData>
    </soapenv:Body>
  </soapenv:Envelope>
```

- \_\_\_ a. Change the `soapenv:mustUnderstand` attribute within the `wsse-Security` tag from `1` to `true` so that the attribute reads: `soapenv:mustUnderstand="true"`.
- \_\_\_ b. Save the file.



### Information

There is a bug in your WebSphere Application Server regarding the **mustUnderstand** attribute. The DataPower service generates the correct value of `1`, while the back-end server is looking for `true`.

2. Configure the web service proxy to decrypt the message body by using a crypto key object.
- a. In the WebGUI, click the **Control Panel > Web Service Proxy** icon.
  - b. Click the `AddressSearchProxy` link in the catalog list to view the web service proxy settings.
  - c. Click the **Proxy Settings** tab. This page contains general configuration information.
  - d. In the **Decrypt Key** list, select **AliceKey**.

### General Configuration

|                                        |                                                                                                                                                                   |  |  |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| <b>Comments</b>                        |                                                                                                                                                                   |  |  |
| <b>Type</b>                            | <input type="radio"/> Dynamic Backend<br><input type="radio"/> Static Backend<br><input checked="" type="radio"/> Static from WSDL<br><small>*</small>            |  |  |
| <b>Decrypt Key</b>                     | <input type="text" value="AliceKey"/> <span style="border: 1px solid #ccc; padding: 2px;">+</span> <span style="border: 1px solid #ccc; padding: 2px;">...</span> |  |  |
| <b>EncryptedKeySHA1 Cache Lifetime</b> | <input type="text" value="0"/>                                                                                                                                    |  |  |



#### Note

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

- e. Click **Apply**.

3. Use cURL to test the **Decrypt Key** specified by using the encrypted message: `findByLocation-enc.xml`.
- a. Open a Terminal window and navigate to the `<lab_files>/WSSecurity` directory.



#### Hint

You can open the `findByLocation-enc.xml` file in an editor to view the encrypted message.

- \_\_ b. Enter the following command:

```
curl --data-binary @findByLocation-enc.xml
http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch
```

- \_\_ c. Verify that the unencrypted response is returned correctly.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header/>
    <soapenv:Body>
        <p796:findByLocationResponse
            xmlns:p796="http://east.address.training.ibm.com">
            <findByLocationReturn>
                <Address>
                    <details>
                        <city>New York</city>
                        <state>NY</state>
                        <street>4944 Broadway Street</street>
                        <zipCode>10145</zipCode>
                    </details>
                    <name>
                        <firstName>Steve</firstName>
                        <lastName>Anderson</lastName>
                        <title>Mr</title>
                    </name>
                </Address>
            </findByLocationReturn>
        </p796:findByLocationResponse>
    </soapenv:Body>
</soapenv:Envelope>
```



### Information

In this step, you sent an encrypted message to the web service proxy that is decrypted by using the AliceKey crypto object. The original message was encrypted by using the AliceCert certificate object.

## 10.6. Configure field-level message decryption on the web service proxy

In this section, you decrypt specific XML elements of incoming requests. In the previous section, you configured a crypto key to decrypt the message payload. However, that key cannot decrypt specific elements. Therefore, you must configure a policy that does field-level decryption. In addition, you must turn off automatic validation of request messages because validation occurs before the policy (which contains the decrypt action) is executed.

- \_\_\_ 1. Create a rule for the `EastAddressSearch findByName` operation and turn off automatic request message validation.
  - \_\_\_ a. On the Configure Web Service Proxy page, click the **Policy** tab.
  - \_\_\_ b. Expand **wsdl:** EastAddressSearch.wsdl > **service:** AddressSearchService > **port:** AddressSearch > **port-operation:** findByName.
  - \_\_\_ c. Click the green and red check mark icon beside the **findByName** operation. In the user policy window that opens, clear the **Schema validate request messages** check box.

The screenshot shows the IBM DataPower configuration interface. The left pane displays a tree structure of policies and rules. The right pane shows a detailed view of the selected policy. A context menu is open over the 'findByName' entry under 'port-operation'. The 'Effective Value' column shows a red 'X' next to the 'Schema validate request messages' checkbox, indicating it is disabled. The 'Local Value' column lists several checkboxes, with 'Schema validate request messages' being the one highlighted with a blue border.

| 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 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 |

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

**Important**

Schema validation of the request message occurs before the policy is executed. Leaving it selected would result in failed schema validation (encrypted elements in message that is not specified in schema), and your policy would never be executed.

- \_\_\_ 2. Click **Processing Rules** under the **findByName** operation. The rule configuration area reloads and the policy editor displays.
- \_\_\_ 3. Click **New Rule** in the policy editor.
- \_\_\_ 4. In the rule configuration area, change the Rule Direction to **Client to Server**.
- \_\_\_ 5. On the rule configuration line, you see an automatically generated **Match** action. It is not necessary to configure this action.
- \_\_\_ 6. Add a **Decrypt** action that decrypts the message by using the `AliceKey` object.
  - \_\_\_ a. In the rule configuration area, drag the **Decrypt** action after the **Match** action.
  - \_\_\_ b. Double-click the **Decrypt** action to configure the decryption settings.
  - \_\_\_ c. In the **Message Type** field, check that **Selected Elements (Field-Level)** is selected. The page reloads with the Document Crypto Map field.
  - \_\_\_ d. Click **+** (plus sign) to create a document crypto map.

The document crypto map is used to specify an XPath expression of the elements to decrypt.

- \_\_\_ e. Enter the name: `Name_DCM`
- \_\_\_ f. In the **XPath Expression** field, click **XPath Tool**.
- \_\_\_ g. Select the `findByName.xml` file in the drop-down list.
- \_\_\_ h. Under **Namespace Handling**, select **local**. Click **Refresh** if you do not see the contents of the `findByName.xml` file.
- \_\_\_ i. Click the `<name>` element to generate an XPath expression.

**Note**

Recall that the encrypted message replaces the `<name>` element with an `<EncryptedData>` element. On the decrypt side, you must point to this `<EncryptedData>` element to indicate which element to decrypt, rather than the `<name>` element.

- \_\_\_ j. Change the 'name' in the XPath expression to: 'EncryptedData'

**XPath \***

```
/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByName']/*[local-name()='EncryptedData']
```

- \_\_\_ k. Click **Done**. You are back in the Configure Document Crypto Map page. The XPath expression that you generated is populated in the **XPath Expression** field.
- \_\_\_ l. Click **Apply** to save the crypto map.
- \_\_\_ m. Back on the Configure Decrypt Action dialog, select `AliceKey` from the Decrypt Key list.

**Decrypt**

|                            |                                                                                                                                                                           |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Message Type</b>        | <input type="radio"/> Entire Message/Document<br><input checked="" type="radio"/> Selected Elements (Field-Level)<br><input type="radio"/> Advanced<br><br><code>*</code> |
| <b>Document Crypto Map</b> | <input type="text" value="Name_DCM"/> <input type="button" value="+"/> <input type="button" value="..."/> *                                                               |
| <b>Asynchronous</b>        | <input type="radio"/> on <input checked="" type="radio"/> off                                                                                                             |
| <b>Decrypt Key</b>         | <input type="text" value="AliceKey"/> <input type="button" value="+"/> <input type="button" value="..."/> <input checked="" type="checkbox"/> Save                        |

- \_\_\_ n. Click **Done**.
- \_\_\_ o. At the top of the page, click **Apply**.
- \_\_\_ 7. Use cURL to test the field-level decryption.
- \_\_\_ a. Open a Terminal window and navigate to the `<lab_files>/WSSecurity` directory.
- \_\_\_ b. Using a text editor, open the `findByName-enc.xml` file. Notice that only the `<name>` element is encrypted. You send this message to test against the rule that you configured.

- \_\_\_ c. Change the value of the `mustUnderstand` attribute from `1` to `true` as was done previously in `findByLocation-enc.xml`.

```
<soapenc:Envelope ... >
<soapenv:Body>
<q0:findByName>
<xenc:EncryptedData Id="G232471f8-8D"
Type="http://www.w3.org/2001/04/xmlenc#Element">
...
</xenc:EncryptedData>
</q0:findByName>
</soapenv:Body>
</soapenv:Envelope>
```

- \_\_\_ d. Compare this file against the original `findByName.xml` file, which is in the `<lab_files>/WSSecurity` directory, to examine the elements that are encrypted. You receive a response with the address information of this person.

```
<SOAP-ENV:Envelope>
<SOAP-ENV:Body>
<q0:findByName>
<name>
<firstName>Sarah</firstName>
<lastName>Chan</lastName>
<title>Mrs</title>
</name>
</q0:findByName>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- \_\_\_ e. Enter the following command:

```
curl --data-binary @findByName-enc.xml
http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch
```

- \_\_\_ f. Verify that the response is returned correctly in the terminal window. You get a response that contains Sarah Chan in Cleveland, OH.



### Information

In this step, you sent a message that had specific elements encrypted to the web service proxy. The `findByName...` rule that contains the `decrypt` action decrypts the message.

## 10.7. Configure message verification on the web service proxy

In this section, you use a validation credential to verify the message payload of the `findByLocation` operation.

- \_\_\_ 1. Create a rule to verify the digital signature that is contained in a message and turn off schema validation for the request message.
  - \_\_\_ a. Click the **Policy** tab.
  - \_\_\_ b. Under web service proxy policy, expand **wsdl:** EastAddressSearch.wsdl > **service:** AddressSearchService > **port:** AddressSearch > **port-operation:** `findByLocation`.
  - \_\_\_ c. Click the green check mark beside the **findByLocation** operation. In the window that opens, clear the **Schema validate request messages** check box.
  - \_\_\_ d. Click **Done**.



### Note

Schema validation of the request message occurs before the policy is executed. When a message is signed, an additional attribute in the `<soapenv:Body>` element is added. This attribute would cause schema validation of the request message to fail.

```
<soapenv:Body wsu:Id="Body-7eaf5ae0-19ae-4f58-abf2-b4fea040f85a">
    <q0:findByLocation>
        <city/>
        <state>NY</state>
    </q0:findByLocation>
</soapenv:Body>
```

- \_\_\_ e. Click **Processing Rules** under the **findByLocation** operation.
- \_\_\_ f. Click **New Rule** to create a rule.
- \_\_\_ g. Set the Rule Direction to **Client to Server**.
- \_\_\_ h. The rule configuration area opens with a **Match** action.
- \_\_\_ i. In the rule configuration area, drag a **Verify** action after the **Match** action.
- \_\_\_ j. Double-click the **Verify** action to configure it.
- \_\_\_ k. The **Verify** action requires a validation credential object. Click **+** (plus sign) to create a **Validation Credential** object.



## Information

A **validation credential object** is used to validate a signer certificate. This object is created by referencing an existing crypto certificate object.

- \_\_ l. In the Configure Crypto Validation Credentials window, enter the name: AliceValidCred
- \_\_ m. In the **Certificates** field, select **AliceCert** from the list and click **Add**.
- \_\_ n. Leave the default values for the remaining fields and click **Apply**.
- \_\_ o. Verify that the **Validation Credential** field in the Configure Verify Action window is populated with the `AliceValidCred` object that you created.

## Verify

The screenshot shows the 'Verify' configuration window with several tabs: Asynchronous, Signature Verification Type (RSA/DSA Signatures), Optional Signer Certificate, and Validation Credential. The Validation Credential tab is active, showing a dropdown menu with 'AliceValidCred' selected, a '+' button, a '...' button, and a 'Save' checkbox checked. An orange box highlights the 'AliceValidCred' dropdown.

- \_\_ p. Click the **Advanced** tab.
- \_\_ q. Select **off** for **Check Timestamp Expiration**.



## Important

The signed message that you generated earlier inserts a timestamp into the message. If you wait too long to verify the signature, the timestamp expires. In most scenarios, you would leave this option on because it protects your service from repeated attacks that use the same message.

- \_\_ r. Click **Done**.
- \_\_ s. Click **Apply** at the top of the page.
- \_\_ 2. Use cURL to test the verify action.
  - \_\_ a. Open a Terminal window and navigate to the `<lab_files>/WSSecurity` directory.

- \_\_\_ b. Using a text editor, open the `findByLocation-sign.xml` file to examine the digital signature. The `<soapenv:Header>` contains the digital signature. Change the value of the attribute `mustUnderstand` from `1` to `true` as was done previously.

```
<soapenc:Envelope ... >
<soapenv:Header>
<wsse:Security soapenv:mustUnderstand="1">
[XML Digital Signature elements]
</wsse:Security>
</soapenv:Header>
<soapenv:Body wsu:Id="Body-7eaf5ae0-19ae-4f58-abf2-b4fea040f85a">
<q0:findByLocation>
<city/>
<state>NY</state>
</q0:findByName>
</soapenv:Body>
</soapenv:Envelope>
```

- \_\_\_ c. Enter the following command:

```
curl --data-binary @findByLocation-sign.xml
http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch
```

- \_\_\_ d. Verify that the response is returned correctly (Steve Anderson in New York).



### Information

In this step, you sent a signed message to the web service proxy. The AliceKey crypto object is used to verify the message.

- \_\_\_ e. In the `findByLocation-sign.xml` file, add an extra character in the `<state>` element. Changing the message causes the verify action to fail.
- \_\_\_ f. Use cURL to send the request again to the web service proxy.

You get a SOAP fault because of failed signature validation.

```
<env:Envelope  
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">  
    <env:Body>  
        <env:Fault>  
            <faultcode>env:Client</faultcode>  
            <faultstring>Hash values do not match. (from  
client)</faultstring>  
        </env:Fault>  
    </env:Body>  
</env:Envelope>
```

- \_\_\_ g. As soon as you are done testing, remove the character that you inserted and save the file.

## 10.8. Send a signed and encrypted message to the web service proxy

In section 5, you decrypted the entire message body for the `findByLocation` operation. In the previous section, you verified a signature for the `findByLocation` operation.

In this section, you combine message decryption and signature verification in the same rule. In fact, these actions are already configured in the same rule. Recall that the `decrypt` key you specified in the web service proxy settings applies to all messages that are decrypted. As soon as the message is decrypted, the policy for that operation is invoked.

In this example, the **findByLocation** message is decrypted and its digital signature is verified.

The web service proxy decrypts and then verifies the signature for the **findByLocation** message. You must reverse these steps when generating the message to encrypt and sign.

The `findByLocation-sign.xml` file is already signed, so you can use cURL to encrypt this message.

- \_\_\_ 1. Use the CryptoMPGW multi-protocol gateway to encrypt the `findByLocation-sign.xml` file.

Enter the following command to encrypt the `findByLocation-sign.xml` file and save the output into the `findByLocation-sign-enc.xml` file.

```
curl --data-binary @findByLocation-sign.xml  
http://<dp_public_ip>:<mpgw_crypto_port>/encrypt >  
findByLocation-sign-enc.xml
```

- \_\_\_ 2. Use cURL to test the signed and encrypted XML message.

- \_\_\_ a. Enter the following command to invoke the `findByLocation` operation with a signed and encrypted message:

```
curl --data-binary @findByLocation-sign-enc.xml  
http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch
```



### Warning

If the execution of the cURL command returns an error, open the `findByLocation-sign-enc.xml` file by using a text editor and look for the `mustUnderstand` attribute. Change the value of this attribute from `1` to `true` and test again.

- \_\_\_ b. Verify that the response is returned correctly in the Terminal window (Steve Anderson in New York).

## **End of exercise**

## Exercise review and wrap-up

In this exercise, you used multi-protocol gateways to create messages that were encrypted and signed. These messages were decrypted and validated on the web service proxy. The **encryption**, **decryption**, **sign**, and **verify** steps are actions that are configured within a rule. You used cURL to test these web service security functions.

# Exercise 11. Web service authentication and authorization

## What this exercise is about

This exercise covers the authentication, authorization, and auditing (AAA) capabilities of the IBM WebSphere DataPower SOA Appliances. Enforcing client authentication and authorization means that access to services is restricted to permitted clients.

## What you should be able to do

At the end of the exercise, you should be able to:

- Configure an action to enforce authentication and authorization policies
- Configure an action to verify a SAML assertion token for authentication and authorization purposes

## Introduction

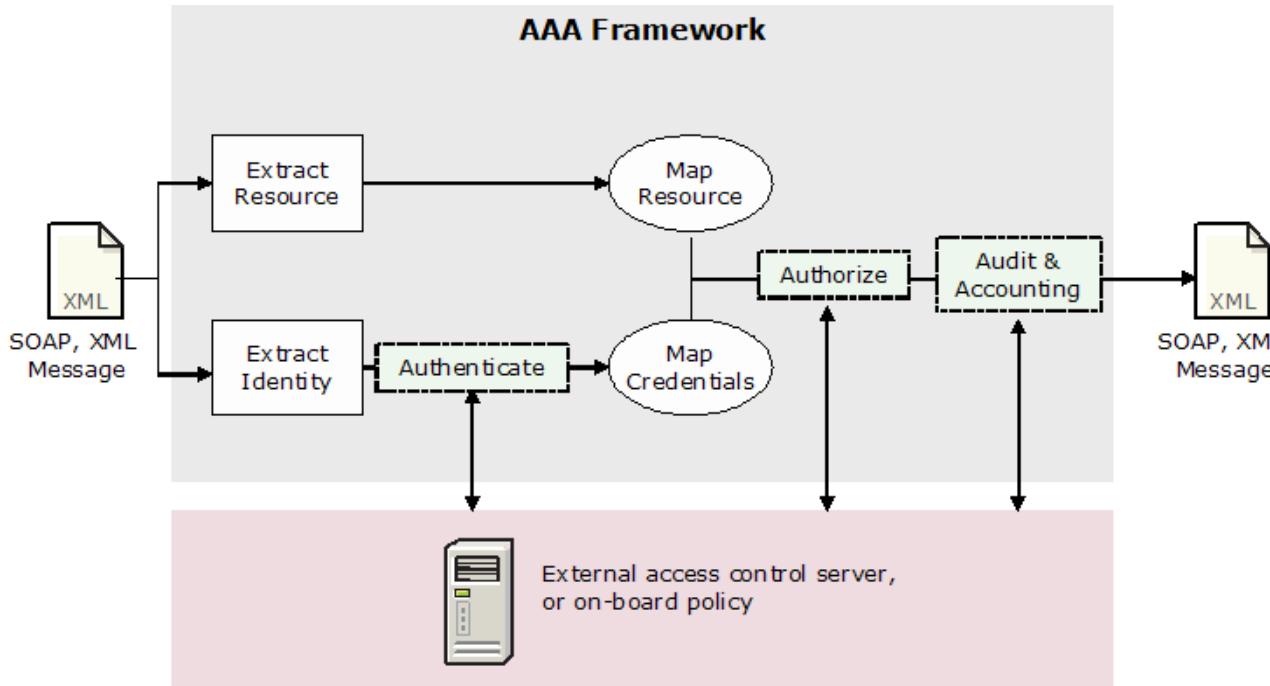
Every instance of the Address Search web service includes an operation to download all address entries. The retrieveAll operation returns a sequence of entries with name and address details. For privacy concerns, normal users should not be allowed to call this operation. Only developers should have access to this web service operation for testing and maintenance.

To enforce this condition, configure a AAA policy action in the web service proxy for the East Address Search web service to restrict access to the retrieveAll operation.

With the new AAA policy, web service clients supply user credentials in the request message. The WS-Security specification defines how to encode this information as a security token. As a policy enforcement point, the AAA policy verifies the user credentials and determines whether the user is authorized to access the operation.

Another option is to defer the authentication and authorization task to a separate security server. The security server can vouch for an incoming request by adding a security assertion to the request message with the Security Assertion Markup Language (SAML).

Finally, the AAA policy provides logging and auditing features. Certain company policies, and existing laws, mandate that the system track any access to private customer information.



## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- **Eclipse**, to send requests to the DataPower appliance
- The **Address Search** web services that run on WebSphere Application Server
- Access to the `<lab_files>` directory

# Exercise instructions

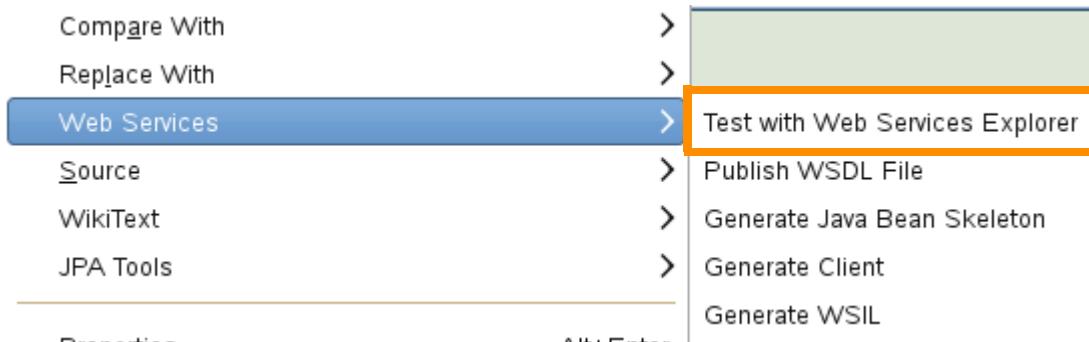
## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 9: Configure a web service proxy.”
- 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
  - *<dp\_internal\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<dp\_public\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<backend\_server\_ip>*: instructor-assigned address for the enterprise application server that hosts the Address Search web services
  - *<wsp\_proxy\_port>*: port number of the web service proxy that is created in a previous exercise

## 11.1. Test the East Address Search web service

After completing the previous exercise, the `AddressSearchProxy` web service proxy is active and forwarding web service requests to the `EastAddressSearch` web service. Verify that the **retrieveAll** operation properly retrieves the entire list of addresses from the service.

- \_\_\_ 1. You use the Web Services Explorer from the Eclipse workbench to access the `EastAddressSearch` web service.
  - \_\_\_ a. From the Linux desktop, open Eclipse to the following workspace directory:  
`/home/localuser/workspace`
  - \_\_\_ b. In the Project Explorer, right-click `EastAddressSearch.wsdl` and select **Web Services > Test with Web Services Explorer**.



- \_\_\_ 2. Add an endpoint to send web service requests through the `AddressSearchProxy`.
  - \_\_\_ a. In the Web Services Explorer, locate the list of web service endpoint addresses in the Actions pane.

- \_\_\_ b. Click **Add** in the Endpoints list.

**WSDL Binding Details**

Shown below are the details for this **SOAP <binding>** element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

**Operations**

Name	Documentation
<a href="#">retrieveAll</a>	--
<a href="#">findByName</a>	--
<a href="#">findByLocation</a>	--

**Endpoints**

	Endpoints
<input type="checkbox"/>	<a href="http://training.ibm.com:9080/EastAddress/services/AddressSearch">http://training.ibm.com:9080/EastAddress/services/AddressSearch</a>

**Add** **Remove**

Go Reset

- \_\_\_ c. In the new Endpoint entry, enter

`http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch`

Remember to replace the references to the DataPower appliance host address and the web service proxy port with the values assigned to you by the instructor.

**Endpoints**

	Endpoints
<input type="checkbox"/>	<a href="http://training.ibm.com:9080/EastAddress/services/AddressSearch">http://training.ibm.com:9080/EastAddress/services/AddressSearch</a>
<input type="checkbox"/>	<a href="http://172.16.78.44:6555/EastAddressSearch">http://172.16.78.44:6555/EastAddressSearch</a>

**Add** **Remove**



### Important

The release level of Eclipse running in the image has a bug in the Web Service Explorer: maximizing a view might cause editable fields in the view to become non-editable. To fix the problem, restore the view or perspective to its regular size.

- \_\_\_ d. Click **Go** to add the new endpoint.

3. Run the **retrieveAll** operation on the new web service endpoint.
- \_\_ a. In the Endpoints list, select the check box next to the new endpoint.
  - \_\_ b. In the **Operations** list, select **retrieveAll**.
  - \_\_ c. In the Invoke a WSDL Operation page, make sure that the new endpoint is shown.
  - \_\_ d. Click **Go** to run the **retrieveAll** web service operation.
  - \_\_ e. In the Status pane, confirm that the Web Services Explorer received a list of 50 addresses from the web service. Click the **Source** link to view the SOAP message that the service returns.

The screenshot shows the 'Status' pane of the IBM Web Services Explorer. At the top, there is a 'Source' link. Below it, the XML response for the 'retrieveAll' operation is displayed. The XML structure includes 'retrieveAllResponse' and 'retrieveAllReturn' elements, which contain 'Address' and 'details' sub-elements. The details for one address are expanded, showing 'city (string): Atlanta', 'state (string): GA', 'street (string): 1477 Highland Boulevard', and 'zipCode (string): 30309'. Another 'Address' element is present, with its 'details' sub-element also expanded, showing 'city (string): Milwaukee'.

```
<?xml version="1.0"?>
<ns1:retrieveAllResponse xmlns:ns1="http://www.ibm.com/websphere/webservices/soa/1.0">
  <ns1:retrieveAllReturn>
    <ns1:Address>
      <ns1:details>
        <ns1:city>Atlanta</ns1:city>
        <ns1:state>GA</ns1:state>
        <ns1:street>1477 Highland Boulevard</ns1:street>
        <ns1:zipCode>30309</ns1:zipCode>
      </ns1:details>
      <ns1:name>
        <ns1:firstName>Timothy</ns1:firstName>
        <ns1:lastName>Green</ns1:lastName>
        <ns1:title>Mr</ns1:title>
      </ns1:name>
      <ns1:Address>
        <ns1:details>
          <ns1:city>Milwaukee</ns1:city>
        </ns1:details>
      </ns1:Address>
    </ns1:Address>
  </ns1:retrieveAllReturn>
</ns1:retrieveAllResponse>
```

## 11.2. Configure a AAA policy action for a web service operation

Certain web service operations are not accessible by all users. For example, running a **retrieveAll** operation from a real-life address directory places significant strain on the network if the payload is a list of 100,000 address entries. However, such an operation might be necessary between different departments in a company. The **retrieveAll** operation is also useful for developers that want to test the service against a sample data set.

IBM WebSphere DataPower SOA Appliances can enforce an authentication, authorization, and auditing policy in a pipeline. In this section, add a AAA policy action to reject **retrieveAll** operation requests for all but a few users.

- \_\_\_ 1. In the IBM WebSphere DataPower SOA Appliance WebGUI, click **Control Panel > Web Service Proxy**.
- \_\_\_ 2. Click the `AddressSearchProxy` from the list.
- \_\_\_ 3. In the Configure Web Service Proxy page, click the **Policy** tab.
- \_\_\_ 4. Expand the WSDL policy tree of the `EastAddressSearch` web service to the **port-operation** level (**findByName**, **findByLocation**, and **retrieveAll**).



### Information

The Web Service Proxy Policy page enforces one or more processing rules at the proxy, WSDL document definition, WSDL service, WSDL port, and WSDL operation levels. There are two default rules for any service that is associated with the web service proxy:

- The **default request rule** matches all incoming messages from this proxy and applies the service level monitoring (SLM) rules that are defined in the **SLM** tab. The rule direction is set to **Client to Server**.
- The **default response rule** matches all outgoing messages from this proxy and returns the result to the user. The rule direction is set to **Server to Client**.

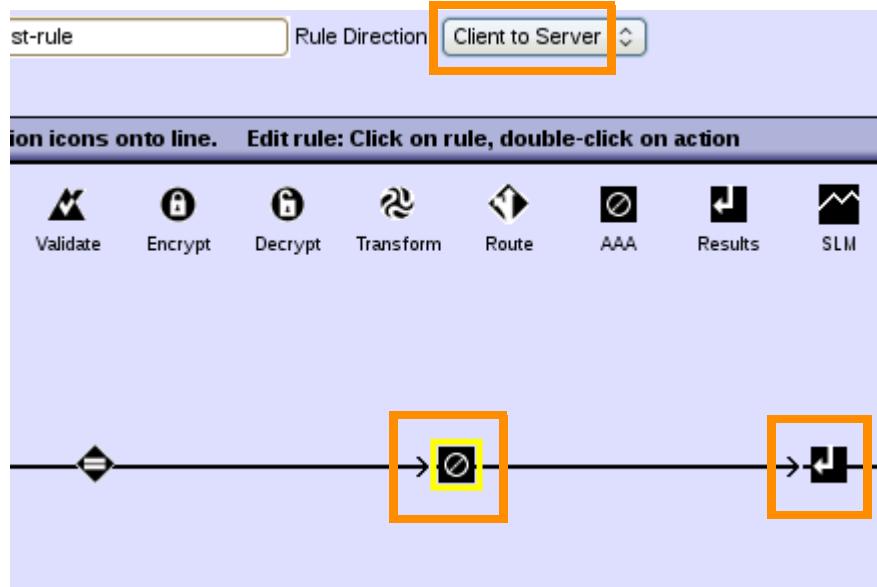
Keep in mind that only one policy rule is applied for each request or response message that passes through the web service proxy. That is, if a request message matches a user-defined rule at the WSDL port level, the proxy does not run the default request rule.

- \_\_\_ 5. Create a request processing rule, with a client-to-server AAA action, for the `EastAddressSearch` **retrieveAll** web service operation.
  - \_\_\_ a. Under the **port-operation**: `retrieveAll` entry in the Web Service Proxy Policy page, click **Processing Rules**.

- \_\_\_ b. In the processing rule editor, under the Web Service Proxy Policy list, click **New Rule**.

The screenshot shows the IBM DataPower processing rule editor's policy tree. The top level is 'wsdl: EastAddressSearch.wsdl'. Below it is 'service: {http://east.address.training.ibm.com}AddressSearchService'. Under the service, there is a 'port: {http://east.address.training.ibm.com}AddressSearch'. Under the port, three port operations are listed: 'port-operation: findByLocation', 'port-operation: retrieveAll', and 'port-operation: findByName'. The 'port-operation: retrieveAll' node is currently selected and highlighted with a red box. Its 'Processing Rules' section is also highlighted with a red box.

- \_\_\_ c. Enter `retrieveAll_AAA_Request_Rule` as the new **rule name**.  
 \_\_\_ d. Drag a **AAA** action after the **Match** action.  
 \_\_\_ e. Drag a **Results** action to the right of the **AAA** action.  
 \_\_\_ f. Change the rule direction to **Client to Server**.



- \_\_\_ 6. Double-click the **AAA** action in the new rule.

- \_\_\_ 7. In the Configure AAA Action page, create a AAA policy by clicking + (plus sign).
- \_\_\_ 8. In the Configure an Access Control Policy page, enter a name of AddressAdmin, and click **Create**.
  - \_\_\_ a. Select **Password-carrying UsernameToken from WS-Security header**.

**AAA Policy Name:** AddressAdmin

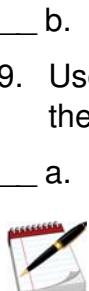
**Define how to extract a user's identity from an incoming request.**

- HTTP Authentication Header
- Password-carrying UsernameToken Element from WS-Security Header
- Derived-key UsernameToken Element from WS-Security Header
- BinarySecurityToken Element from WS-Security Header



**Note**

This page determines how the access control policy identifies the client from the incoming request message. Click the **Help** link on the upper right corner of the page for a summary of all of the different identification methods that the **AAA** action supports.



**Note**

The “Define how to authenticate the user” page determines how the policy validates the user identity that is stated within the incoming request message.

In a production environment, an authorized users list usually exists in a corporate directory server, such as a Lightweight Directory Access Protocol (LDAP) server. The identity information in the request message must be verified against the corporate directory server.

For the purposes of this case study, the list of authorized users exists in an XML file, `AAAInfo.xml`. Since the file is included with every DataPower appliance, it is used only for testing.

- \_\_\_ b. In the URL section at the bottom of the page, select the `store:///` file location and select `AAAInfo.xml` from the menu.

**Define how to authenticate the user.**

Method
<input type="radio"/> Accept a SAML Assertion with a Valid Signature
<input type="radio"/> Accept an LTPA token
<input type="radio"/> Bind to Specified LDAP Server
<input type="radio"/> Contact a SAML Server for a SAML Authentication Statement
<input type="radio"/> Contact a WS-Trust Server for a WS-Trust Token
<input type="radio"/> Contact ClearTrust Server
<input type="radio"/> Contact Netegrity SiteMinder
<input type="radio"/> Contact NSS for SAF Authentication
<input type="radio"/> Contact Tivoli Access Manager
<input type="radio"/> Custom Template
<input type="radio"/> Pass Identity Token to the Authorize Step
<input type="radio"/> Retrieve SAML Assertions Corresponding to a SAML Browser Art
<input type="radio"/> Use an Established WS-SecureConversation Security Context
<input type="radio"/> Use certificate from BinarySecurityToken
<input checked="" type="radio"/> Use DataPower AAA Info File
<input type="radio"/> Use specified RADIUS Server
<input type="radio"/> Validate a Kerberos AP-REQ for the Correct Server Principal
<input type="radio"/> Validate the Signer Certificate for a Digitally Signed Message.
<input type="radio"/> Validate the SSL Certificate from the Connection Peer
*

**URL**

store:///

`AAAInfo.xml`

- \_\_\_ c. Leave the “Define how to map credentials” **Method** set to **None**.
- \_\_\_ d. Click **Next**.
- \_\_\_ 10. Configure the access control policy to allow calls to the **retrieveAll** web service operation for any authenticated user.

**Note**

The “Define how to extract the user” page specifies how the access control policy determines what resource the client requested. In this scenario, the resource in question is the **retrieveAll** web service operation.

- \_\_\_ a. Select the **Local name of the request element** box to retrieve the child element from the SOAP request message.

According to the SOAP specification, the first child element in a SOAP request message must be the web service operation name.

### Configure an Access Control Policy

**AAA Policy Name:** AddressAdmin

#### Define how to extract the resources.

##### Resource Identification Methods

- URL Sent to Back End
- URL Sent by Client
- URI of Toplevel Element in the Message
- Local Name of Request Element
- HTTP Operation (GET/POST)
- XPath Expression
- Processing Metadata
- \*

#### Define how to map resources.

##### Method

\*

[Back](#) [Next](#) [Cancel](#)

- \_\_\_ b. Leave the “Define how to map resources” **Method** set to **None**.
- \_\_\_ c. Click **Next**.
- \_\_\_ d. Select **Allow Any Authenticated Client** to ensure that only authenticated users can call the **retrieveAll** web service operation.

**Information**

The “Define how to authorize a request” page determines the access rules that are based on the resource and the user.

- \_\_\_ e. Click **Next**.
- \_\_\_ 11. Configure a counter to track all requests to **retrieveAll** that the access control policy rejects. To protect against denial-of-service attacks, block the operation if two failed attempts occur within the span of 10 seconds.
- \_\_\_ a. Under Monitors, click **Reject Counter Tool**.

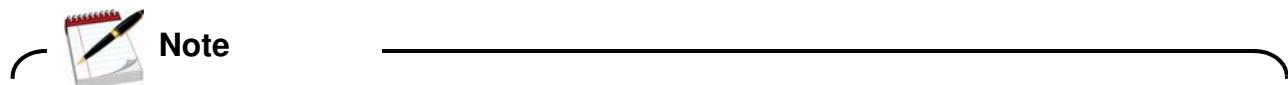
The screenshot shows the 'Monitors' configuration screen. Under the 'Rejected Counter' section, there is a dropdown menu set to '(none)' and a button labeled 'Reject Counter Tool'. The 'Reject Counter Tool' button is highlighted with an orange rectangle.

- \_\_\_ b. Enter `retrieveAllCounter` as the count monitor name.
- \_\_\_ c. Change the **interval for measuring rate of authentication failures** to 10000 milliseconds.
- \_\_\_ d. Change the **maximum rate of authentication failures** to two messages per interval.
- \_\_\_ e. Leave the **block interval** at 60000 milliseconds, the equivalent of 1 minute.
- \_\_\_ f. Change the **authentication failure log level** to **emergency**.
- \_\_\_ g. Click **Next**.

The screenshot shows the 'Count Monitor Name' configuration dialog. Several fields are highlighted with orange rectangles: the 'Count Monitor Name' field containing 'retrieveAllCounter', the 'Interval for Measuring Rate of Authentication Failures' field containing '10000', the 'Max. Rate of Authentication Failures' field containing '2', the 'Authentication Failure Log Level' field containing 'emergency', and the 'Next' button at the bottom.

- \_\_\_ h. Click **Commit**, and then click **Done**.

- 
- \_\_\_ 12. On the Configure An Access Control Policy page, examine the monitoring, logging, and postprocessing options available with the access control policy.



On the Configure An Access Control Policy page, under **Monitors**, the authorized counter and the rejected counter track how many requests were allowed or denied access. The rejected counter has an additional feature to block further requests if a threshold is reached.

The **Logging** section tracks the request and response message details for any authorized or rejected access attempts. The amount of message detail that is logged depends on the log level that was configured.

The postprocessing section describes actions that can be applied to the message after the authentication, authorization, and auditing steps. For example, the DataPower appliance can generate a security token that is based on the decision that the access control policy makes.

- 
- \_\_\_ 13. Click **Commit** to save the AddressAdmin access control policy, and then click **Done**.

- \_\_\_ 14. Click **Done** to save the settings for the AAA action in the **retrieveAll** policy rule.



The **Results** action takes the result from the **AAA** action and forwards it to the application server. It is using the PIPE DataPower variable.

- 
- \_\_\_ 15. Click **Done** to confirm the settings within the action.

- \_\_\_ 16. Click **Apply** on the top of the Web Service Proxy Policy page.

## 11.3. Test the access control policy for a web service operation

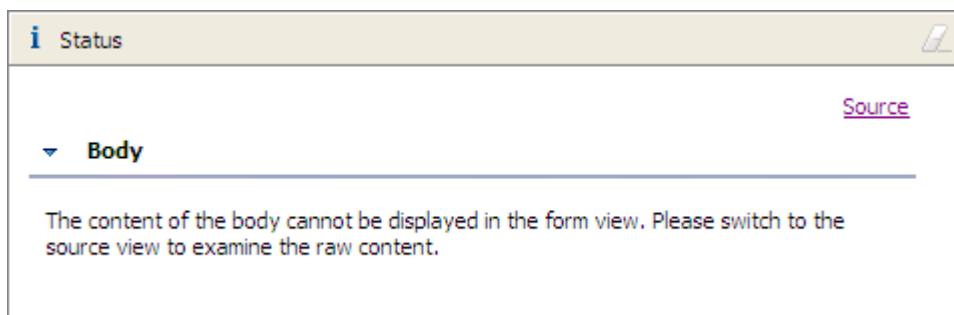
The new `retrieveAll_AAA_Request_Rule` completes the following actions when a client requests access to the **retrieveAll** web service operation:

- Decide whether to accept or reject the web service operation request according to the **AddressAdmin** access control policy.
- Use the `<wsse:Username>` element in the WS-Security declaration within the SOAP message header to identify the client.
- Use the `AAAInfo.xml` sample DataPower appliance AAA information XML file to authenticate the identity of the client.
- Determine the requested resource by extracting the web service operation name as the child element within a SOAP message body.
- Allow any authenticated client to call the **retrieveAll** web service operation.
- Use the **retrieveAllCounter** rejected access counter object to keep a count of all rejected attempts to access the web service operation.
- For two failed access attempts within the span of 10 seconds, block access to the web service operation for 60 seconds.

Follow the steps that are detailed in Section **11.1: Test the East Address Search web service** to test the **retrieveAll** web service operation by using the Web Services Explorer in Eclipse.

- 1. Using the Web Services Explorer in Eclipse, call the **retrieveAll** operation against the web service proxy on the DataPower appliance.
  - a. Open Eclipse, if necessary.
  - b. Run the Web Services Explorer from the pop-up menu of the `EastAddressSearch.wsdl` WSDL document.
  - c. Add an endpoint to call the `AddressSearchProxy` web service proxy. (It might still be there from your previous test.)
  - d. Run the **retrieveAll** web service operation in the same manner as you did earlier.
  - e. In the Invoke a WSDL Operation page, make sure that the endpoint address matches the DataPower `AddressSearchProxy` URI.
  - f. Click **Go** to run the **retrieveAll** operation.

2. Examine the results from calling the `AddressSearchProxy` **retrieveAll** operation.
- a. In the Web Services Explorer, locate the Status pane directly below the Actions pane. Instead of receiving a list of address entries, a warning message is shown in the form view.



- b. Click the **Source** link to view the actual SOAP request and response messages in XML form.
- c. Verify that the SOAP Response Envelope includes a web services SOAP fault message, `Rejected by policy`.

**SOAP Request Envelope:**

```
- <soapenv:Envelope xmlns:q0="http://east.address.training.ibm.com"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <soapenv:Body>
  <q0:retrieveAll />
</soapenv:Body>
</soapenv:Envelope>
```

**SOAP Response Envelope:**

```
- <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
- <env:Body>
- <env:Fault>
  <faultcode>env:Client</faultcode>
  <faultstring>Rejected by policy. (from client)</faultstring>
</env:Fault>
</env:Body>
</env:Envelope>
```

The **faultcode** value `env:Client` indicates that the problem originated from the web service requester. The **faultstring** provides a human-readable description; the **AddressAdmin** access control policy rejected the request message.

- \_\_\_ 3. Examine the log file that is associated with the web service proxy to determine which part of the access control policy caused the appliance to reject the request message.
- \_\_\_ a. In the DataPower WebGUI, click **Control Panel > Web Service Proxy**.
- \_\_\_ b. Select the magnifying glass (logs) icon for `AddressSearchProxy`.
- \_\_\_ c. Examine the error log entries, which are sorted by most recent time:
- 1) The operation fails **AAA Authentication** because no WS-Security user name element is shown in the SOAP request message header.
  - 2) The operation also fails **AAA Authorization** as only authorized clients are allowed to access the specified resource.
  - 3) The second entry from the top identifies the AAA policy rule that rejected the message: **retrieveAll\_AAA\_Request\_Rule**.
  - 4) The first entry from the top indicates that the `AddressSearchProxy` sent back a SOAP fault message to the client.

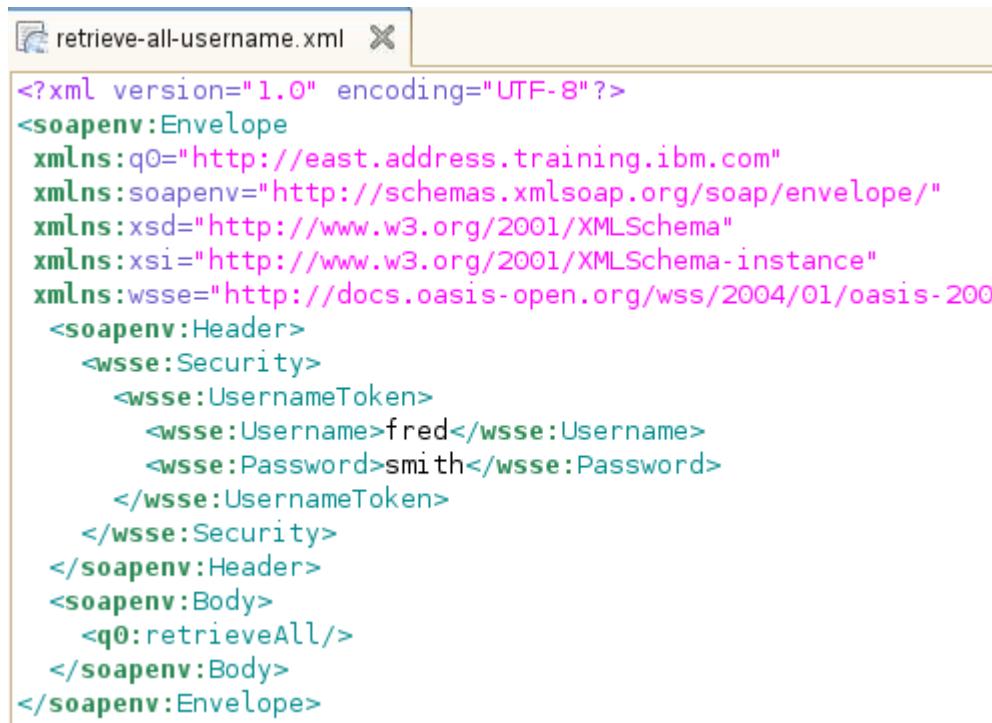
30.11	0x00d30003	wsgw (AddressSearchProxy): Rejected by filter; SOAP fault sent
30.11	0x80c00009	wsgw (AddressSearchProxy): request retrieveAll_AAA_Request_Rule #1 aaa: 'INPUT AddressAdmin stored in PIPE' failed: Rejected by policy.
30.11	0x80c00010	wsgw (AddressSearchProxy): Execution of 'store:///dp/aaapolicy.xsl' aborted: Rejected by policy.
30.11	0x8380000e	wsgw (AddressSearchProxy): Message rejected
30.11	0x80c00037	wsgw (AddressSearchProxy): Reject set: Rejected by policy.
30.11	0x01d30002	wsgw (AddressSearchProxy): AAA Authorization Failure
30.11	0x838000f2	wsgw (AddressSearchProxy): anyauthenticated authorization failed with credential 'SPECIAL-FORMAT-NOT-PRINTED' for resource 'retrieveAll'

- \_\_\_ d. Close the log window.

## 11.4. Test the access control policy with a web services security user name token

To satisfy the `AddressAdmin` access control policy, every request to the **retrieveAll** web service operation must include a WS-Security Username Token. The token itself must represent an authorized user according to the `AAAInfo.xml` DataPower appliance AAA information XML file. Use the cURL command-line utility to send a sample SOAP message with the correct user credentials.

- \_\_\_ 1. Examine the sample SOAP request message with a WS-Security Username Token, `retrieve-all-username.xml`.
  - \_\_\_ a. From the Eclipse workbench, select **File > Open File** from the main menu bar.
  - \_\_\_ b. Select `<lab_files>/AAA/retrieve-all-username.xml`.
  - \_\_\_ c. Click the **Source** tab at the bottom of the editor.
  - \_\_\_ d. Examine the sample SOAP request message in the XML editor. This message is identical to the one sent from the Web Services Explorer, with an additional `<wsse:UsernameToken>` element in the WS-Security header.



```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:q0="http://east.address.training.ibm.com"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-2004-03-wss-wssecurity-username-token-profile-1.1.xsd">
  <soapenv:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>fred</wsse:Username>
        <wsse:Password>smith</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <q0:retrieveAll/>
  </soapenv:Body>
</soapenv:Envelope>

```

- \_\_\_ e. Close the editor. Be careful **not** to save any changes to the file.
- \_\_\_ 2. Use the cURL command-line utility to send the new **retrieveAll** web service operation request.
  - \_\_\_ a. Open a terminal window and go to the `<lab_files>/AAA/` directory.

- \_\_\_ b. Run the cURL command-line utility with the following parameters:

```
curl --data-binary @retrieve-all-username.xml  
http://<dp_appliance_ip>:<wsp_proxy_port>/EastAddressSearch  
--include --header "Content-type: text/xml; charset=utf-8"
```

- \_\_\_ c. Verify that the SOAP response message succeeds, returning the entire list of address entries.

## 11.5. Configure the access control policy to handle SAML assertions

In the current scenario, the DataPower appliance completes three distinct security tasks: it processes the client security claims, verifies the identity of the client, and determines whether the client has the authority to access the requested resources. If the request requires a series of resources, these three security tasks can be repeated at each policy enforcement point.

One alternative is to use a security server at the edge of the network as a gatekeeper. When the security server authenticates the client and authorizes access to certain resources, the server adds a security assertion to the request. In other words, the security server vouches for the claims in the original messages. The internal resources can trust the assertion instead of checking the original claims each time.

The Security Assertion Markup Language (SAML) provides a framework for creating and exchanging security information in an XML format. DataPower appliances can use SAML tokens, and generate new SAML assertions after performing a AAA action.

For information about SAML, refer to the OASIS Security Services Technical Committee website at: <http://www.oasis-open.org/committees/security/>

Modify the existing `AddressAdmin` access control policy to accept SAML assertions for authenticating clients and authorizing clients to internal resources.

- \_\_\_ 1. Locate the `retrieveAll_AAA_Request_Rule` proxy rule within the `AddressSearchProxy` web services proxy.
  - \_\_\_ a. In the WebGUI, click **Control Panel > Web Service Proxy**.
  - \_\_\_ b. Click `AddressSearchProxy`.
  - \_\_\_ c. Click the **Policy** tab.
  - \_\_\_ d. Expand the tree to view the **port-operations** level.
  - \_\_\_ e. Under `EastAddressSearch.wsdl`, **port-operation:**`retrieveAll` click **Processing Rules** to gain access to the `retrieveAll_AAA_Request_Rule`.
- \_\_\_ 2. Create an access control policy named `AddressSAML`.
  - \_\_\_ a. In the `retrieveAll_AAA_Request_Rule`, edit the AAA action by double-clicking the icon.
  - \_\_\_ b. Create an access control policy by clicking + (plus sign) beside the `AddressAdmin` AAA policy.
  - \_\_\_ c. Enter `AddressSAML` as the name of the new access control policy.
  - \_\_\_ d. Click **Create** to configure the new access control policy.

- \_\_\_ 3. Use the SAML assertion signature to configure the new access control policy to authenticate the client.
- \_\_\_ a. In the “Define how to extract a user’s identity from an incoming request” page, select **Name from SAML Attribute Assertion**.

**Identification Methods**

- Kerberos AP-REQ from SPNEGO Token
- Subject DN of the SSL Certificate from the Connection Peer
- Name from SAML Attribute Assertion
- Name from SAML Authentication Assertion
- SAML Artifact
- Client IP Address

- \_\_\_ b. Click **Next**.
- \_\_\_ c. In the “Define how to authenticate the user” page, select **Accept a SAML Assertion with a Valid Signature**. This policy accepts the name that is given in the SAML assertion if the digital signature for the assertion is valid.

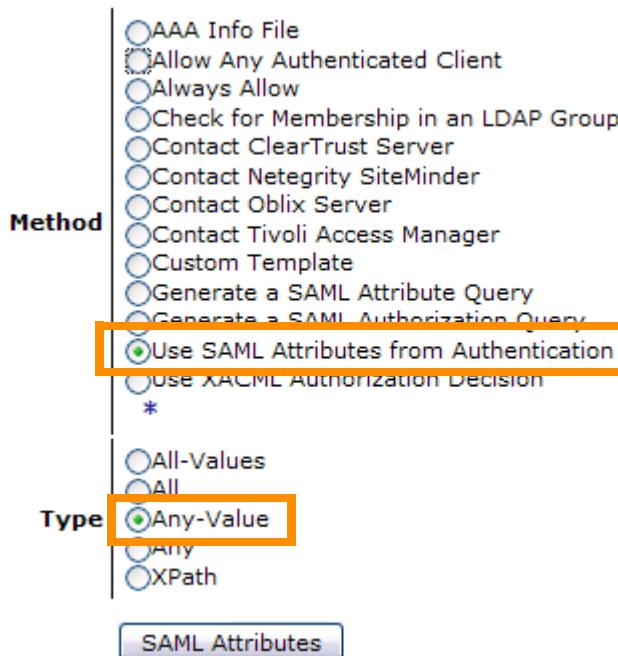
**Define how to authenticate the user.**

**Method**

- Accept a SAML Assertion with a Valid Signature
- Accept an LTTPA token
- Bind to Specified LDAP Server
- Contact a SAML Server for a SAML Authentication Statement
- Contact a WS-Trust Server for a WS-Trust Token
- Contact ClearTrust Server
- Contact Netegrity SiteMinder
- Contact Oblix server
- Contact Tivoli Access Manager
- Custom Template
- Pass Identity Token to the Authorize Step
- Retrieve SAML Assertions Corresponding to a SAML

- \_\_\_ d. Click **Next**.
- \_\_\_ 4. Determine the resources that the client requests by examining the SAML assertion attributes. Authorize authenticated clients that have the following SAML attributes in the assertion:
- **Namespace URI:** `http://address.training.ibm.com`
  - **Local Name:** `EastAddressSearch`
  - **Attribute Value:** `Query`
- \_\_\_ a. In the “Define how to extract the resources” page, select **Local name of request element**. The requested element is the local name of the child element within the SOAP message body.
- \_\_\_ b. Click **Next**.
- \_\_\_ c. In the “Define how to authorize a request” page, select **Use SAML Attributes from Authentication** as the method.

- \_\_\_ d. For the type, select **Any-Value**.



- \_\_\_ e. Click **SAML Attributes** to specify the name and value that are expected in the SAML assertion.
- \_\_\_ f. Click **Add** to create a SAML Attribute entry.
- \_\_\_ g. Enter the values as shown:
- **Namespace URI:** `http://address.training.ibm.com`
  - **Local Name:** `EastAddressSearch`
  - **Attribute Value:** `Query`
- \_\_\_ h. Click **Submit**.
- \_\_\_ i. Click **Done** to return to the “Define how to authorize a request” page.
- \_\_\_ j. Click **Next**.
- \_\_\_ 5. Accept the default values for monitoring, logging, and postprocessing.
- \_\_\_ 6. Click **Commit** to save the changes to the `AddressSAML` access control policy, and click **Done**.
- \_\_\_ 7. Click **Done** to return to the Web Service Proxy Policy page.

- \_\_\_ 8. Save the newly created rule in the web service proxy.
- \_\_\_ a. Click **Apply** on the top of the Web Service Proxy Policy page in the Web Service Proxy editor.

**Web Service Proxy Name** [up]  \*

[View Log](#) | [View Status](#) | [View C](#)

## 11.6. Test the access control policy with a SAML assertion

To satisfy the AddressSAML access control policy, every request to the retrieveAll web service operation must include a digitally signed SAML assertion element. The SAML assertion must include at least one qualified name that matches `http://address.training.ibm.com/EastAddressSearch`, with a value of `Query`. Use the cURL command-line utility to send a sample SOAP message with the correct user credentials.

- \_\_\_ 1. Examine the sample SOAP request message with a WS-Security Username Token, `retrieve-all-saml.xml`.
  - \_\_\_ a. From the Eclipse workbench, select **File > Open File** from the main menu bar.
  - \_\_\_ b. Select `<lab_files>/AAA/retrieve-all-saml.xml`.

- \_\_\_ c. Examine the sample SOAP request message in the XML editor. This example is the same as the previous two SOAP requests, except for the digitally signed SAML assertion element in the header.



```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"><SOAP
<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="#ID85ed3d35-a093-4e85-b4c5-6a192719c09e">
        <Transforms>
            <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <DigestValue>euGybUspqaiXqxVzvmPDbdkqcGM=</DigestValue>
    </Reference>
</SignedInfo>
<SignatureValue>jHnnWNV2PQO/63QyEWmR2Dz9oJtv+gAt49R7Unt1LPMQxthTbSRmaza
QTELMAkGA1UECBMCT04xEDA0BgNVBAcTB1Rvcm9udG8xDDAKBgNVBAoTA01CTTEX
MBUGA1UECxMOU29mdHdhcmUgR3JvdXAxDjAMBgNVBAMTBUFsaWN1MB4XDTA2MDcz
MTE4NTIyMFoXDTA5MDczMDE4NTIyMFowYzELMAkGA1UEBhMCQOEExCzAJBgNVBAgT
Ak9OMRAwDgYDVQQHEwdUb3JvbnRvMQwwCgYDVQQKEwNJQkOxFzAVBgNVBAstD1Nv
ZnR3YXJ1IEdyb3VwMQ4wDAYDVQQDEwVBbG1jZTCBnzANBgkqhkiG9wOBAQEFAAOB
jQAwgYkCgYEAxhppa/OJyJf4+DiPSfjZWAh6rvBRYtpk62iN2SLxdyeWUUQU/BHF
13qBBKincIEhBDKJvjGCTDhsQqKUVmKO2pkaV/Av6WXxHVSmcMRp32y7a7atnp5
+1hLJmDyD86zH+HSosEL9tq5hBw6riTdhBW8N68nYSpqRftCXUokSY8CAwEAAaOB
ODCbzTAMBgNVHRMEBTADAQH/MBOGA1UdDgQWBBSeAYCmEf/9jUrf5pr8kcOWoU2n
eTCBkAYDVROjBIGIMIGfGBSeAYCmEf/9jUrf5pr8kcOWoU2neaFnpuGUwYzELMAkG
A1UEBhMCQOEExCzAJBgNVBAgTAK9OMRAwDgYDVQQHEwdUb3JvbnRvMQwwCgYDVQQK
EwNJQkOxFzAVBgNVBAstD1NvZnR3YXJ1IEdyb3VwMQ4wDAYDVQQDEwVBbG1jZYIE
UI2vzTALBgNVHQ8EBAMCArwDQYJKoZIhvcNAQEFBQADgYEAtuSSf1i7wLUEqmTt
mysvv6/MgiIW/9F01VmMgCZxhkMS70OESi+NwIKpBv3sOCYLu6951GkSIHa6dET9Q
FpaYim3IjkUCFWDE1AkpJbcH45V1wkRkfNhOsuX1QSEUWFMtXYdz861E+BaN2ekV
Pn1E06kPdPL7QO63wcoGO+pgN78=
</X509Certificate><X509IssuerSerial><X509IssuerName>CN=Alice, OU=Software G

```

- \_\_\_ d. Close the editor. ***Do not*** save any changes to the file.
- \_\_\_ 2. Use the cURL command-line utility to send the new **retrieveAll** web service operation request.
- \_\_\_ a. Open a terminal window and go to the `<lab_files>/AAA/` directory.
  - \_\_\_ b. Run the **cURL** command-line utility with the following parameters:
- ```

curl --data-binary @retrieve-all-saml.xml
http://<dp_appliance_ip>:<wsp_proxy_port>/EastAddressSearch
--include --header "Content-type: text/xml; charset=utf-8"

```
- \_\_\_ c. Verify that the SOAP response message succeeds, returning the entire list of address entries.

\_\_\_ d. Save your configuration in the DataPower appliance.

## **End of exercise**

## Exercise review and wrap-up

The first part of the exercise reviewed the concept of the web service proxy. This service virtualizes one or more web services through one policy enforcement point on the DataPower SOA appliance.

To restrict access to the retrieveAll web service operation, an authentication, authorization, and auditing (AAA) action was applied to the operation processing rule. The first access control policy, AddressAdmin, allowed only clients with the correct user name and password in a WS-Security UsernameToken element.

To demonstrate other forms of authentication and authorization, a second access control policy named AddressSAML handled SAML assertions. The digital signature for the SAML assertion prevents tampering and identified the client as well. The policy authorized access to the retrieveAll operation when certain SAML attributes were included in the assertion.

# Exercise 12.Creating message counter monitors for a AAA policy

## What this exercise is about

Message monitors, although not as configurable as service level monitors, still have a place in the design and configuration of a DataPower service. One use for them is the counters for the AAA policy. In this exercise, you create message count monitor objects. You attach the message monitors as controls for the “authorized counter” and the “rejected counter” in a AAA policy object. These counters are used to manage the traffic through the AAA policy, dependent on whether the AAA policy authorizes or rejects the request.

## What you should be able to do

At the end of the exercise, you should be able to:

- Create message count monitors
- Specify message count monitors as the “authorized counter” and the “rejected counter” in a AAA policy

## Introduction

Every instance of the Address Search web service includes an operation to download all address entries. The retrieveAll operation returns a sequence of entries with name and address details. For privacy concerns, normal users should not be allowed to call this operation. Only developers should have access to this web service operation for testing and maintenance.

To enforce this condition, configure a AAA policy action in the web service proxy for the East Address Search web service to restrict access to the retrieveAll operation.

Message counter monitors are used to track users who are successfully authenticating when calling the AddressSearchProxy. The count monitor is configured in the AAA policy action.

A second count monitor is created to track users who are rejected or unsuccessfully authenticated attempting to access the AddressSearchProxy web service. The second count monitor is also configured from within the AAA policy action.

A shell script is used to stress the system that simulates multiple sequential logins. The shell script saves the student from having to type the same information multiple times. The shell script requires certain parameters to run. The parameters include the DataPower appliance public IP address, the processing port, and the number of times to submit a AddressSearchProxy login to the appliance.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- The **Address Search** web services that run on WebSphere Application Server
- Access to the `<lab_files>/` directory

# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 9: Configuring a web service proxy” and “Exercise 11: Web service authentication and authorization.”
- 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
  - *<dp\_internal\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<dp\_public\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<backend\_server\_ip>*: instructor-assigned address for the enterprise application server that hosts the Address Search web services
  - *<wsp\_proxy\_port>*: port number of the web service proxy that is created in a previous exercise

## 12.1. Test the East Address Search web service

After completing the previous exercise, the `AddressSearchProxy` web service proxy is active and forwarding web service requests to the `EastAddressSearch` web service. Verify that the `retrieveAll` operation properly retrieves the entire list of addresses from the service.

Test the `EastAddressSearch` configuration by opening a terminal window and running cURL. The following steps ensure that the back-end web service is 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. Open a terminal window and go to the `<lab_files>/counter` directory.

\_\_\_ 2. Run the shell script command-line utility with the following parameters:

```
./sendMsgs.sh <dp_public_ip> <wsp_proxy_port> 1
```

\_\_\_ 3. Verify that the SOAP response message succeeds, returning the entire list of address entries.



### Note

The shell script `sendMsgs.sh` file is sending a request to the `retrieveAll` operation of the `EastAddressSearch` web service. The third parameter identifies the number of times the operation call is called. Here the operation is called only one time. Later in this exercise the operation is called multiple times to stress the system to reach thresholds that are defined in the policy and web service count monitors.

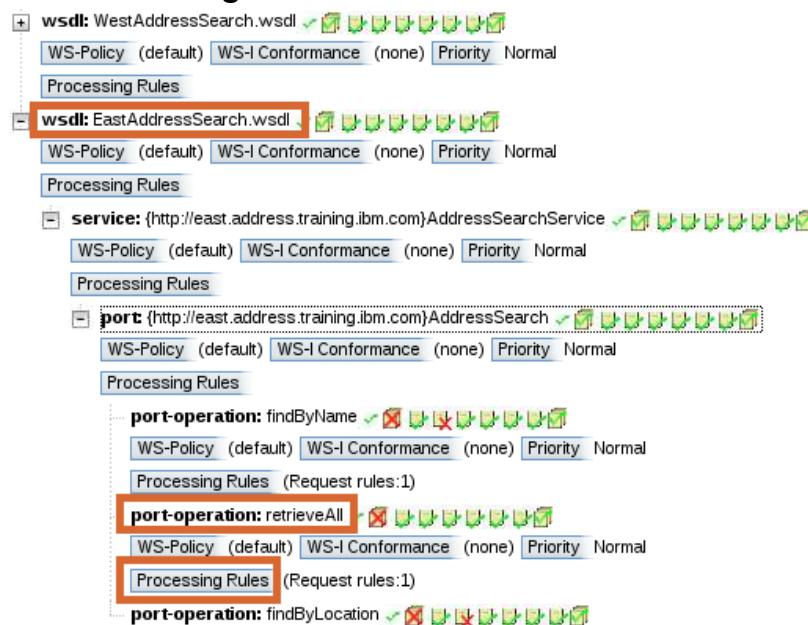
\_\_\_ 4. Verify in the terminal window that an XML list of addresses is returned.

## 12.2. Configure a AAA policy action for message counter

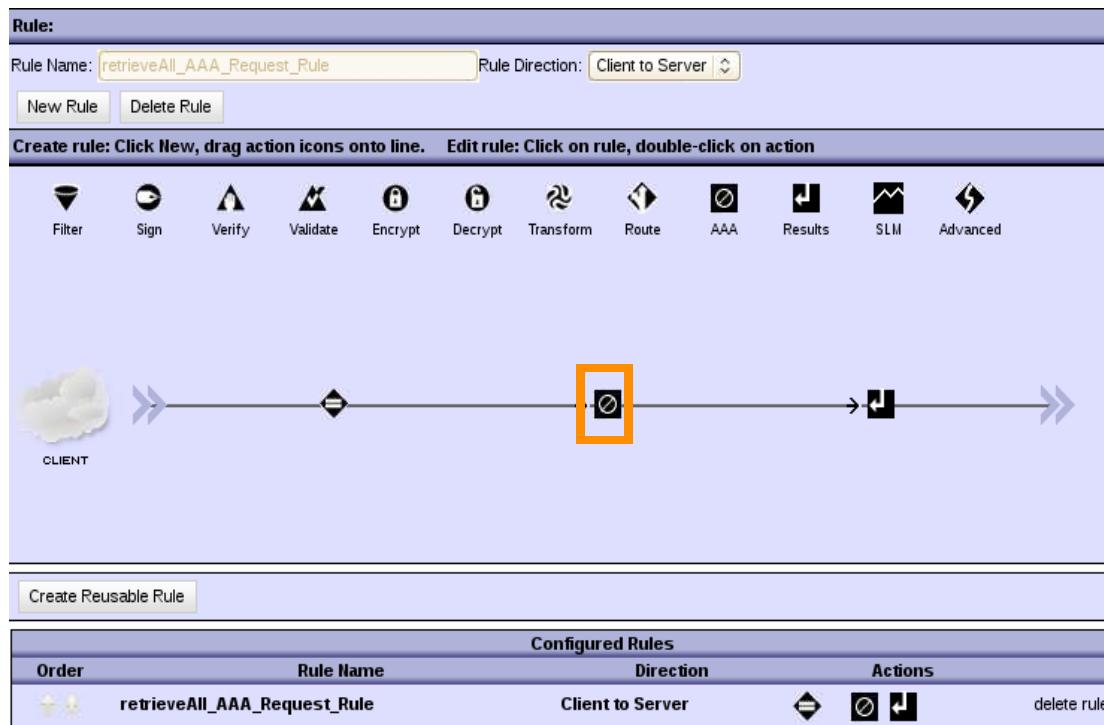
Certain web service operations are not accessible by all users. For example, running a **retrieveAll** operation from a real-life address directory places significant strain on the network if the payload is a list of 100,000 address entries. However, such an operation might be necessary between different departments in a company. The **retrieveAll** operation is also useful for developers that want to test the service against a sample data set.

IBM WebSphere DataPower SOA Appliances can enforce an authentication, authorization, and auditing policy in a pipeline. In this section, you modify the existing AAA policy action to monitor all message requests to the **retrieveAll** operation. Two types of monitoring are configured. The first type of monitoring that is configured counts all the successful authentications that the AAA policy action processes. The second type of monitor counts all the rejections of all messages over a certain threshold.

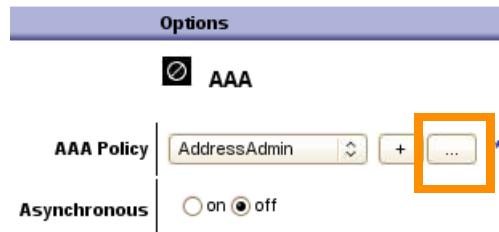
- \_\_\_ 1. In the IBM WebSphere DataPower SOA Appliance WebGUI, click **Control Panel > Web Service Proxy**.
- \_\_\_ 2. Click the `AddressSearchProxy` from the list.
- \_\_\_ 3. In the Configure Web Service Proxy page, click the **Policy** tab.
- \_\_\_ 4. Expand the WSDL policy tree of the `EastAddressSearch` web service to the **port-operation** level (**findByName**, **findByLocation**, and **retrieveAll**).
- \_\_\_ 5. Edit the request processing rule, with a client-to-server AAA action, for the `EastAddressSearch` **retrieveAll** web service operation.
  - \_\_\_ a. Under the **port-operation: retrieveAll** entry in the Web Service Proxy Policy page, click **Processing Rules**.



- \_\_\_ b. Scroll down to the policy editor and double-click the AAA action that is on the retrieveAll\_AAA\_Request\_Rule processing rule.

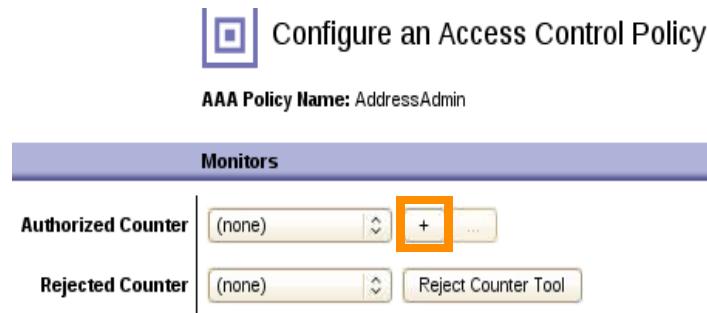


- \_\_\_ 6. Select the AddressAdmin AAA policy for editing.



- \_\_\_ 7. Click **Next** to accept the current Define how to extract a user's identity from an incoming request configuration.
- \_\_\_ 8. Click **Next** to accept the current Define how to authenticate the user configuration.
- \_\_\_ 9. Click **Next** to accept the current Define how to extract the resources configuration.
- \_\_\_ 10. Click **Next** to accept the current Define how to authorize a request configuration.

- \_\_\_ 11. Select the + to create a new Authorized counter for the AAA policy.



- \_\_\_ 12. Enter the name `retrieveAll-Counter` for the message count monitor.  
 \_\_\_ 13. Ensure the Thresholds/Filters tab is selected.  
 \_\_\_ 14. Click Add to define a new Thresholds/Filters.

| Name    | Interval | Rate Limit | Burst Limit | Action |
|---------|----------|------------|-------------|--------|
| (empty) |          |            |             |        |

\_\_\_ 15. Enter the following information in the appropriate fields:

- **Name:** retrieveAll-log
- **Interval:** 60000
- **Rate Limit:** 10
- **Burst Rate:** 10

|             |  |
|-------------|--|
| Name        | <input type="text" value="retrieveAll-log"/> * |
| Interval    | <input type="text" value="6000"/> msec *       |
| Rate Limit  | <input type="text" value="10"/> messages *     |
| Burst Limit | <input type="text" value="10"/> messages *     |
| Action      | <input type="text" value="(none)"/> *          |

\_\_\_ 16. Click + to create an Action.

- Set the action **Name** to: retrieveAll-Notify
- Leave the **Type** set to Notify.
- Set the **Log Priority** to: error

Message Filter Action

|                      |   |
|----------------------|---|
| Name                 | <input type="text" value="retrieveAll-Notify"/> *                       |
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments             | <input type="text"/>  |
| Type                 | <input type="button" value="Notify"/> *                                 |
| Log Priority         | <input type="button" value="error"/> *                                  |



Setting the log priority to error prints the message in the color red in the system log. The red messages are easier to locate during the lab.

- \_\_\_ 17. Click **Apply** to save the Action.
- \_\_\_ 18. Click **Apply** to save the Thresholds/Filters.
- \_\_\_ 19. The message counter should be displayed with the following parameters.

 Configure Message Count Monitor

Main Thresholds/Filters

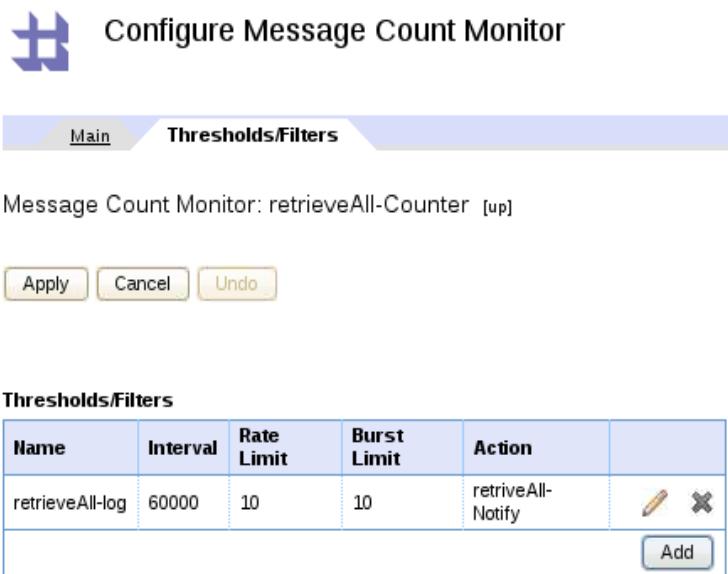
Message Count Monitor: retrieveAll-Counter [up]

Apply Cancel Undo

**Thresholds/Filters**

| Name            | Interval | Rate Limit | Burst Limit | Action             |   |
|-----------------|----------|------------|-------------|--------------------|---|
| retrieveAll-log | 60000    | 10         | 10          | retrieveAll-Notify |   |

Add



- \_\_\_ 20. Click **Done** to save the AAA Policy Action.
- \_\_\_ 21. Add the monitor to the Monitors section of the web service proxy.
  - \_\_\_ a. Click the Monitor tab of the web service proxy.

 Configure Web Service Proxy

Proxy Settings Advanced Proxy Settings Headers/Params WS-Addressing WS-ReliableMessaging **Monitors** XML T



- \_\_\_ b. Select the `retrieveAll-Counter` from the list.
- \_\_\_ c. Click **Add**.

**Monitors**

**Count Monitors**  
(empty)

`retrieveAll-Counter` **Add** + ...



- \_\_\_ 22. The `retrieveAll-Counter` monitor now is shown in the Count Monitors list.

**Monitors**

**Count Monitors**

|                                  |  |
|----------------------------------|--|
| <code>retrieveAll-Counter</code> |   |
| <code>retrieveAll-Counter</code> | Add + ...  |



\_\_\_ 23. Click **Apply** to save the web service proxy.

## 12.3. Test the East Address Search web service count monitor

Test the `EastAddressSearch` configuration by opening a terminal window and running cURL. The following steps ensure that the back-end web service is 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. Open a terminal window and go to the `<lab_files>/counter` directory.
  - \_\_\_ 2. Run the shell script command-line utility with the following parameters:
- ```
./sendMsgs.sh <dp_public_ip> <wsp_proxy_port> 12
```
- \_\_\_ 3. Verify that the SOAP response message succeeds, returning the entire list of address entries. The message happens 12 times since 12 messages are sent to the web service proxy on the DataPower appliance. Mainly you are reviewing the information in the Terminal window as returned addresses from the web service proxy and not SOAP Fault Errors.
  - \_\_\_ 4. Locate the error message that is the count monitor printed. Check the system log to locate the log message created by the `retrieveAll-log` monitoring. Since the action type of log was defined as an error, the message is shown in red.

debug	3434171953	request	172.16.80.11	0x80e0038d	wsgw (AddressSearchProxy): Monitor retrieveAll-Counter Incremented.
error	3434171953		172.16.80.11	0x80e00183	monitor-action (retrieveAll-Notify): Message monitor retrieveAll-Counter triggers filter retrieveAll-log on credential fred
info	3434171953	request	172.16.80.11	0x8380000f	wsgw (AddressSearchProxy): Message allowed
info	3434171953	request	172.16.80.11	0x838000f1	wsgw (AddressSearchProxy): anyauthenticated authorization succeeded with credential 'OutputCredential=admin' for resource 'retrieveAll'

The message counter was configured to log an error type message to the system log when more than 10 messages were received within 60 seconds (60000 milliseconds). The count monitor is successfully working.

## 12.4. Configure a AAA policy action for message rejection that is based on message counter

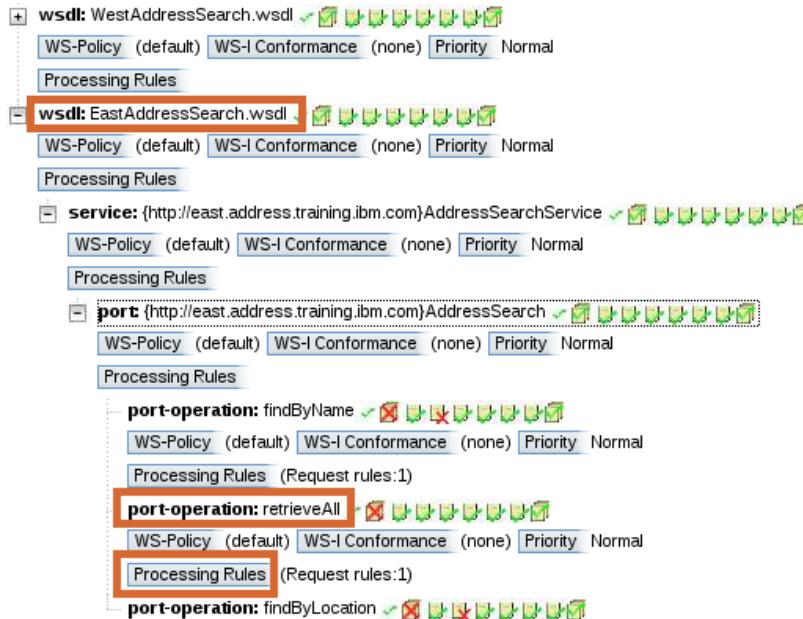
The count monitor is created to count and log all messages that enter the AAA policy action when a rate limit of 10 is exceeded in 60 seconds. Next, a count monitor is created to reject messages when a threshold of 5 failed AAA authentications are exceeded in a 60-second window.

Since the threshold is set for 5, AAA authentication attempts to process the authentication and fails during the AAA policy action. However, when 5 is exceeded, that is, with the sixth failed authentication, the reject counter takes control of authentication. This control causes rejection of any additional authentications from the requesting IP address before the request even enters the service.

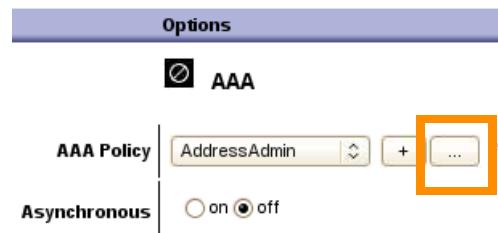
Tracking this process is easiest by reviewing the system log. The transaction ID in the system log views the first five authentications that fail. The sixth authentication failure displays the counter that takes control of the processing and rejects the failed authentication. All subsequent failed authentications are rejected before entering the service policy. The message in the system log is *No error rule is matched*.

- \_\_\_ 1. In the IBM WebSphere DataPower SOA Appliance WebGUI, click **Control Panel > Web Service Proxy**.
- \_\_\_ 2. Click the `AddressSearchProxy` from the list.
- \_\_\_ 3. In the Configure Web Service Proxy page, click the **Policy** tab.
- \_\_\_ 4. Expand the WSDL policy tree of the `EastAddressSearch` web service to the **port-operation** level (`findByName`, `findByLocation`, and `retrieveAll`).

- \_\_\_ 5. Edit the request processing rule, with a client-to-server AAA action, for the EastAddressSearch **retrieveAll** web service operation.
- \_\_\_ a. Under the **port-operation:** `retrieveAll` entry in the Web Service Proxy Policy page, click **Processing Rules**.



- \_\_\_ b. Scroll down to the policy editor and double-click the AAA action that is on the `retrieveAll_AAA_Request_Rule` processing rule.
- \_\_\_ 6. Select the AddressAdmin AAA policy for editing.



- \_\_\_ 7. Click **Next** to accept the current Define how to extract a user's identity from an incoming request configuration.
- \_\_\_ 8. Click **Next** to accept the current Define how to authenticate the user configuration.
- \_\_\_ 9. Click **Next** to accept the current Define how to extract the resources configuration.
- \_\_\_ 10. Click **Next** to accept the current Define how to authorize a request configuration.

- \_\_\_ 11. Select **Reject Counter Tool** create a new Rejected Counter.



- \_\_\_ 12. Enter the name `retrieveAll-RejectCounter` for the message count monitor.  
 \_\_\_ 13. Change the **Interval for Measuring Rate of Authentication Failures** to: 30000  
 \_\_\_ 14. Change the **Max. Rate of Authentication Failures** to: 5

This is a configuration dialog titled 'Create a Rejected Access Counter for use with AAA and XML Threat Protection'. It includes a 'Help' link. The form contains the following fields:

Count Monitor Name	<input type="text" value="retrieveAll-RejectCounter"/> *
Interval for Measuring Rate of Authentication Failures	<input type="text" value="30000"/> msec *
Max. Rate of Authentication Failures	<input type="text" value="5"/> messages/interval *
Block Interval	<input type="text" value="60000"/> msec
Authentication Failure Log Level	<input type="button" value="error"/>

At the bottom are 'Next' and 'Cancel' buttons.

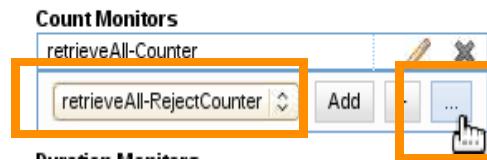
- \_\_\_ 15. Click Next.  
 \_\_\_ 16. Click Commit.  
 \_\_\_ 17. The `retrieveAll-RejectCounter` now exists as the Rejected Counter name. Click **Commit** to accept the Access Control Policy configuration.



- \_\_\_ 18. Click **Done**.  
 \_\_\_ 19. Click **Done**.  
 \_\_\_ 20. Click **Apply** to save the changes to the web service proxy.  
 \_\_\_ 21. Select the **Monitors** tab in the AddressSearchProxy web service proxy.

- \_\_\_ 22. Select **retrieveAll-RejectCounter** from the list box, and then click ... to edit the monitor.

### Monitors



- \_\_\_ 23. Select the **Thresholds/Filter** tab; then click the edit icon, which is the **pencil** to edit the `retrieveAll-RejectCounter-count-monitor-filter`.



### Configure Message Count Monitor

This configuration has been modified, but not yet saved.

Name	Interval	Rate Limit	Burst Limit	Action
retrieveAll-RejectCounter-count-monitor-filter	30000	5	10	retrieveAll-RejectCounter-monitor-action

- \_\_\_ 24. Change the **Burst Limit** to: 7

Name	<input type="text" value="retrieveAll-RejectC"/>
Interval	<input type="text" value="30000"/>
Rate Limit	<input type="text" value="5"/>
Burst Limit	<input type="text" value="7"/>
Action	<input type="text" value="retrieveAll-RejectC"/>

**Information**

The burst limit is the maximum number of messages that are allowed during short increases of message traffic. Normally the burst limit is set to a value twice the value of Rate Limit. In this exercise, the limit is reduced to 7, creating a more manageable output in the system log to review.

\_\_\_ 25. Click **Apply**.

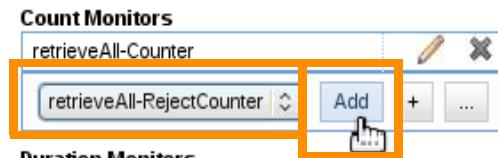
\_\_\_ 26. The **Thresholds/Filter** is shown as:

**Thresholds/Filters**

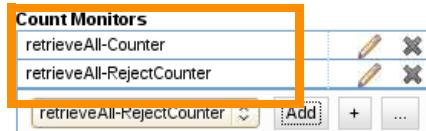
Name	Interval	Rate Limit	Burst Limit	Action
retrieveAll-RejectCounter-count-monitor-filter	30000	5	7	retrieveAll-RejectCounter-monitor-action

\_\_\_ 27. Click **Apply**.

\_\_\_ 28. Click **Add** to add the rejected counter to the monitors that apply to this web service proxy.

**Monitors**

\_\_\_ 29. The `retrieveAll-RejectCounter` monitor now is shown in the Count Monitors list

**Monitors**

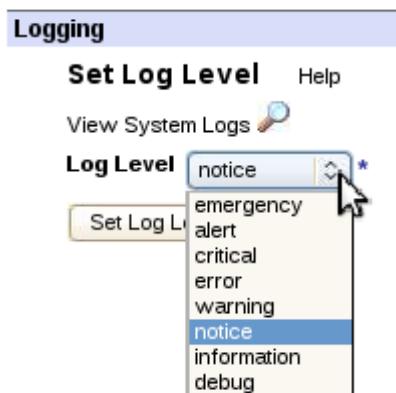
\_\_\_ 30. Click **Apply** to update the web service proxy.

\_\_\_ 31. Click **Save Config** to persist your student domain.

## 12.5. Test the East Address Search web service reject count monitor

Test the `EastAddressSearch` configuration by opening a terminal window and running cURL. The following steps ensure that the back-end web service is 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. Set the log level to **notice** so that the system log messages are easier to review.
  - \_\_\_ a. Click **Control Panel** in the DataPower WebGUI.
  - \_\_\_ b. Click the **TroubleShooting** icon.
  - \_\_\_ c. Select **notice** from the list.



- \_\_\_ d. Click **Set Log Level**.
- \_\_\_ e. Click **Confirm**.
- \_\_\_ f. Click **Close**.
- \_\_\_ 2. Open a terminal window and go to the `<lab_files>/counter` directory.
- \_\_\_ 3. Run the shell script command-line utility with the following parameters:  
`./sendNoAuthMsgs.sh <dp_public_ip> <wsp_proxy_port> 9`
- \_\_\_ 4. SOAP faults and failed messages are displayed on the screen as the authentication values sent to the DataPower appliance are invalid.
- \_\_\_ 5. Review the system log to trace the steps of the Reject Counter in the AAA policy action of the web service proxy.
- \_\_\_ 6. Open the system log.

7. Scroll down to the beginning of the red messages. The messages are reported as error messages, which show up in red for easier identification. Trace the messages by transaction ID (tid), which is the fourth column from the left. AAA authorization failure is occurring, and the reject count monitor takes no action. Since the burst is set to 7, there is no rejection for the first six messages. Using transaction ID, trace the first six messages. In the single example that is listed here, the transaction ID is 3435575409.

Notice that your transaction ID is different from the one listed here.

notice	3435575409		172.16.80.11	0x80c0007b	wsm-stylepolicy (AddressSearchProxy): No error rule is matched.
error	3435575409	error	172.16.80.11	0x00d30003	wsgw (AddressSearchProxy): Rejected by filter; SOAP fault generated.
error	3435575409	request	172.16.80.11	0x80c00009	wsgw (AddressSearchProxy): request 'retrieveAll_AAA_Request_Rule #1 aaa: 'INPUT AddressAdmin stored in PIPE' failed: Rejected by policy.
error	3435575409	request	172.16.80.11	0x80c00010	wsgw (AddressSearchProxy): Execution of 'store:///dp/aaapolicy.xsl' aborted: Rejected by policy.
warn	3435575409	request	172.16.80.11	0x8380000e	wsgw (AddressSearchProxy): Message rejected
error	3435575409	request	172.16.80.11	0x01d30002	wsgw (AddressSearchProxy): AAA Authorization Failure
warn	3435575409	request	172.16.80.11	0x838000f2	wsgw (AddressSearchProxy): anyauthenticated authorization failed with credential 'SPECIAL-FORMAT-NOT-PRINTED' for resource 'retrieveAll'
error	3435575409	request	172.16.80.11	0x01d30001	wsgw (AddressSearchProxy): AAA Authentication Failure
warn	3435575409	request	172.16.80.11	0x83800015	wsgw (AddressSearchProxy): xmlfile authentication failed with ('wssec-username, username='fred' password='*****')

8. Scrolling up the system log, the first message in which the reject message counter takes control is identified. As configured by setting the burst limit to 7, the log is the seventh authentication message failure. The transaction ID here is 3435575425.

notice	3435575425		172.16.80.11	0x80c0007b	wsm-stylepolicy (AddressSearchProxy): No error rule is matched.
error	3435575425	error	172.16.80.11	0x00d30003	wsgw (AddressSearchProxy): Rejected by filter; SOAP fault seen.
error	3435575425	request	172.16.80.11	0x80c00009	wsgw (AddressSearchProxy): request retrieveAll_AAA_Request_Rule #1 aaa: 'INPUT AddressAdmin stored in PIPE' failed: Rejected by policy.
error	3435575425	request	172.16.80.11	0x80c00010	wsgw (AddressSearchProxy): Execution of 'store:///dp/aaapolicy.xls' aborted: Rejected by policy.
error	3435575425		172.16.80.11	0x80e00600	monitor-count (retrieveAll-RejectCounter): Rejected by Count Monitor filter (Measure: XPath) retrieveAll-RejectCounter.
error	3435575425		172.16.80.11	0x80e00183	monitor-action (retrieveAll-RejectCounter-monitor-action) Message monitor retrieveAll-RejectCounter triggers filter retrieveAll-RejectCounter-count-monitor-filter on credential 172.16.80.11
warn	3435575425	request	172.16.80.11	0x8380000e	wsgw (AddressSearchProxy): Message rejected
error	3435575425	request	172.16.80.11	0x01d30002	wsgw (AddressSearchProxy): AAA Authorization Failure
warn	3435575425	request	172.16.80.11	0x838000f2	wsgw (AddressSearchProxy): anyauthenticated authorization failed with credential 'SPECIAL-FORMAT-NOT-PRINTED' for resource 'retrieveAll'
error	3435575425	request	172.16.80.11	0x01d30001	wsgw (AddressSearchProxy): AAA Authentication Failure
warn	3435575425	request	172.16.80.11	0x83800015	wsgw (AddressSearchProxy): xmlfile authentication failed with ('wssec-username', username='fred' password='*****')

9. Scrolling up the system log reveals the last two authentication failures that list the processing as "No error rule is matched." Errors are not longer reported in the log since the web service proxy is not run. Execution of the web service proxy is where the AAA authentication is failing. Instead, the reject counter monitor is rejecting all messages, since the threshold/filter is realized. Authentication failures are being rejected before the service is called.

tid	direction	client	msgid	message	Show last	50	100	all
-----	-----------	--------	-------	---------	-----------	----	-----	-----

3435575457		172.16.80.11	0x80c0007b	wsm-stylepolicy (AddressSearchProxy): No error rule is matched.
3435575441		172.16.80.11	0x80c0007b	wsm-stylepolicy (AddressSearchProxy): No error rule is matched.

## End of exercise

## Exercise review and wrap-up

The first part of the exercise reviewed the concept of the web service proxy. The retrieveAll operation of the EastAddressSearch web service proxy was called.

Count monitors were configured on the retrieveAll operation that counts both usage of the operation and failed authentication calls to the operation. Two count monitors are configured. The first count monitor is configured in the AAA policy action to count successful message calls to the retrieveAll operation.

A second count monitor was created to track users who were rejected or unsuccessfully authenticated while attempting to access the AddressSearchProxy web service. The second count monitor was also configured from within the AAA policy action.

# Exercise 13.Creating a AAA policy by using LDAP

## What this exercise is about

In this exercise, you play the role of an LDAP user and a DataPower developer. You create a AAA policy that validates a credential by using a configured LDAP directory service.

## What you should be able to do

At the end of the exercise, you should be able to:

- Add entries to the IBM Tivoli Directory Server LDAP server
- Authenticate and authorize users on an LDAP server by configuring a AAA policy

## Introduction

In an earlier exercise, you created a AAA policy for the `EastAddressSearch` web service to validate the credentials of clients that access the `retrieveAll` operation. Suppose that an IT department implements an LDAP directory service that all DataPower services must use to authenticate and authorize users. The existing `EastAddressSearch` `retrieveAll` policy must be modified to conform to these changes. The new AAA policy that is created extracts the user name and password from the header of an HTTP basic authentication message. The policy authenticates by using an LDAP directory service with the user name and password that are extracted. Finally, authorization is done by checking for membership of the authenticated identity within a specific group.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- The Address Search web service that is running on WebSphere Application Server

- Access to the <lab\_files> directory
- IBM Tivoli Directory Server V6.3 to support LDAP services

# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 11: Web service authentication and authorization.”
- 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
  - *<dp\_public\_ip>*: IP address of the public services on the appliance
  - *<backend\_server\_ip>*: IP address for the services that is running in WebSphere Application Server (AddressSearch web services, LDAP, registry)
  - *<wsp\_proxy\_port>*: port number for the AddressSearchProxy
  - *<ldap\_server\_port>*: port number for the LDAP administrative console

## 13.1. Obtain the IBM Tivoli Directory Server connectivity information

The LDAP server is configured with the root suffix of `dc=ibm,dc=com`. The root DN of the LDAP server is `cn=root,dc=ibm,dc=com` with the password `websphere`.

In most classroom situations, you access an LDAP server that is running on a back-end server, rather than on your own image.



### Information

In the next section, you play the role of the DataPower user who connects to the directory service setup with the information.

- LDAP host name: `<backend_server_ip>`
- LDAP port: 389
- LDAP Server Name: WCE
- LDAP bind DN: `cn=root`
- LDAP bind password: `websphere`



## 13.2.Add directory entries into IBM Tivoli Directory Server

In this section, you use the IBM Tivoli Directory Server web administration application to examine the existing entries and create new entries in this LDAP server. You first create a user, and then create a group to which you add the user. The user and group are unique to you.

- \_\_\_ 1. Open the IBM Tivoli Directory Server web administrative application.
- \_\_\_ a. Open a web browser and click the bookmark in the toolbar, **ITDS Admin Console**, or enter the following URL: `http://<backend_server_ip>:<ldap_server_port>/IDSWebApp/IDSjsp/Login.jsp`
- \_\_\_ b. In the Login page, select **WCE** for the LDAP Server Name.
- \_\_\_ c. Enter the IBM Tivoli Directory Server user DN `cn=root` and password `web1sphere` as the login information.

The screenshot shows the 'Tivoli Directory Server Web Administration Tool' interface. At the top, it says 'Tivoli Directory Server Web Administration Tool'. Below that, it says 'Directory server login'. There is a form with three fields: 'Enter user name and password'. The first field is 'LDAP Server Name:' with the value 'localhost:389'. The second field is 'User DN:' with the value 'cn=root'. The third field is 'Password:' with several redacted dots. Below the form are two buttons: 'Login' and 'Login to Console admin'.

- \_\_\_ d. Click **Login**.
- \_\_\_ 2. Examine the existing entries in directory server.
- \_\_\_ a. Expand **Directory management > Manage entries**.
- \_\_\_ b. The main web page reloads with the root contents of the directory server.



### Information

The root entry that is configured for this exercise is `dc=ibm,dc=com`. This entry represents the root suffix in the directory server hierarchy. You add entries to this domain.

- \_\_\_ 3. Add a user entry.
- \_\_\_ a. Click **Add** at the top of the table.

- \_\_\_ b. Scroll down the object class list and select **inetOrgPerson**. Click **Next**.

Filter object classes:

All Refresh

Structural object classes:

tbs-instance  
INamingService  
**inetOrgPerson**  
ipNetwork  
ipProtocol  
inService

- \_\_\_ c. In the Select auxiliary object classes page, it is not necessary for you to add any classes. Click **Next**.
- \_\_\_ d. In the Required attributes page, enter the following values:

- **Relative DN:** `cn=student<nn>`, where `nn` = student number
- **Parent DN:** `dc=ibm,dc=com` (alternatively, you can browse to this domain by clicking **Browse**)
- **cn:** `student<nn>`, where `nn` = student number
- **sn:** `student<nn>sn`, where `nn` = student number

Object class inheritance:

### Distinguished name (DN)

Relative DN:

\*

Parent DN:

### Required attributes

Enter the values for the attributes of the entry. For multiple values click **Multiple values** next to the attribute.

cn:

\*

sn:

\*

- \_\_\_ e. Click **Next**.
- \_\_\_ f. On the “Optional attributes” page, scroll down and enter the following value:
- **userPassword:** `web1sphere`
- \_\_\_ g. Click **Finish**.

- \_\_\_ h. You are asked if you want to add a similar entry. Click **No**.

You are returned to the DN directory. A + (plus sign) is displayed next to the dc=ibm,dc=com domain.

- \_\_\_ i. Click the check box next to dc=ibm,dc=com and click **Expand** at the top of the table. You are brought to the subentries, where your newly created person object now displays.

Select	Expand	RDN	Object class	Created	Last modified
<input type="checkbox"/>	<a href="#">+</a>	<a href="#">cn=configuration</a>	ibm-slapdTop		
<input type="checkbox"/>	<a href="#">+</a>	<a href="#">cn=ibmpolicies</a>	container	Dec 2, 2010	Dec 2, 2010
<input type="checkbox"/>	<a href="#">+</a>	<a href="#">cn=localhost</a>	container	Dec 2, 2010	Dec 2, 2010
<input checked="" type="checkbox"/>	<a href="#">+</a>	<a href="#">dc=ibm,dc=com</a>	domain	Dec 3, 2010	Dec 3, 2010

Page 1 of 1      Total: 4    Filtered: 4    Displayed: 4

[Close](#)

- \_\_\_ 4. Add the group grpDPnn and add cn=student<nn> to it.

- \_\_\_ a. Click **Add**.

- \_\_\_ b. On the Select object class page, select **groupOfNames**, and click **Next**.

Filter object classes:

All    Refresh

Structural object classes:

- friendlyCountry
- groupOfCertificates
- groupOfNames**
- groupOfUniqueNames
- groupOfURLs

- \_\_\_ c. In the Select auxiliary object classes page, it is not necessary for you to add any classes. Click **Next**.

- \_\_\_ d. In the Required attributes page, enter the following values:

- **Relative DN:** `cn=grpDP<nn>`, where `nn` = your student number
- **Parent DN:** `dc=ibm,dc=com` (this entry is already completed for you)
- **cn:** `grpDP<nn>`, where `nn` = your student number
- **member:** `cn=student<nn>,dc=ibm,dc=com`, where `nn` = student number (no blanks in entry). This action adds the previously created student to the group.

Object class inheritance:

groupOfNames 

### Distinguished name (DN)

Relative DN:

\*

`cn=grpDP88`

Parent DN:

\*

`dc=ibm,dc=com`

### Required attributes

Enter the values for the attributes of the entry. For multiple values click **values** next to the attribute.

cn:

\*

`grpDP`

Multiple values

member:

\*

`cn=student88,dc=ibm,dc=com`

Multiple values

- \_\_\_ e. Click **Next**.
- \_\_\_ f. No entries are needed on the Optional attributes page. Click **Finish**.
- \_\_\_ g. You are asked if you want to add a similar entry. Click **No**.
- \_\_\_ h. The `dc=ibm,dc=com` domain reopens.

All necessary objects are now completed and the `dc=ibm,dc=com` domain is displayed as follows:

Current location :

<ldap://localhost:389> > dc=ibm,dc=com

Select	Expand	RDN	Object class	Created	Last modified
<input type="checkbox"/>		<a href="#">cn=admin</a>	inetOrgPerson	Dec 3, 2010	Dec 3, 2010
<input type="checkbox"/>		<a href="#">cn=grpDP</a>	groupOfNames	Jul 6, 2011	Jul 6, 2011
<input type="checkbox"/>		<a href="#">cn=student</a>	inetOrgPerson	Jul 6, 2011	Jul 6, 2011
Page 1 of 1		Total: 3 Filtered: 3 Displayed: 3			
<a href="#">Close</a>					



### Note

If the students are sharing a back-end LDAP server, entries for other students might also be in the list.

- 5. Log out of the Administration tool by clicking **Logout** in the navigation menu on the left.



### Information

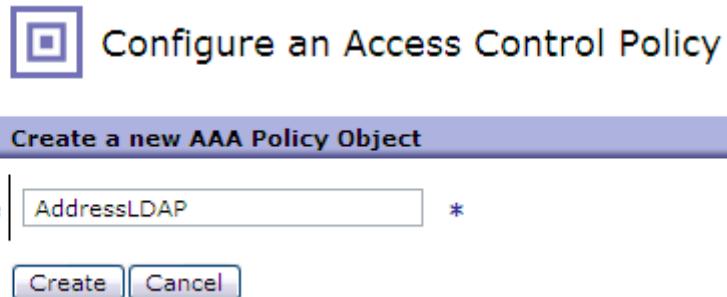
In this section, you added the user account information that DataPower uses to authenticate.

### 13.3. Use LDAP to create a AAA policy

The AddressSearch team requests a policy change. All users must now authenticate against an LDAP server.

In this section, you modify the existing web service proxy policy to use a AAA policy that authenticates and authorizes users by using LDAP.

- \_\_\_ 1. In the DataPower WebGUI, click **Control Panel > Web Service Proxy**.
- \_\_\_ 2. Click **AddressSearchProxy**, and then click the **Policy** tab.
- \_\_\_ 3. Expand **wsdl:EastAddressSearch.wsdl > service: EastAddressSearchService > port: AddressSearch > port-operation: retrieveAll**.
- \_\_\_ 4. Click **Processing Rules** to edit the existing **retrieveAll** rule.
- \_\_\_ 5. Double-click the **AAA** action to open the AAA policy.
- \_\_\_ 6. Create a AAA policy.
  - \_\_\_ a. Click + (plus sign) next to AAA Policy.
  - \_\_\_ b. Enter the policy name: **AddressLDAP**



- \_\_\_ c. Click **Create**.
- \_\_\_ d. For the extract identity step, select **HTTP Authentication Header**.
- \_\_\_ e. Click **Next**.
- \_\_\_ f. For the authentication step, select **Bind to Specified LDAP Server**. The web page reloads with the LDAP configuration options.



#### Hint

Refer to the end of section 1 for the relevant LDAP information that is needed for the next steps.

\_\_ g. Scroll down and enter the LDAP configuration information:

- **Host:** <backend\_server\_ip>
- **LDAP Bind DN:** cn=root
- **LDAP Bind Password:** web1sphere (twice)
- **LDAP Suffix:** dc=ibm,dc=com (no blank spaces)

<b>Method</b>	
<input type="radio"/> Accept a SAML Assertion with a Valid Signature <input type="radio"/> Accept an STS token <input checked="" type="radio"/> Bind to Specified LDAP Server <input type="radio"/> Contact a SAML Server for a SAML Authentication Statement <input type="radio"/> Contact a WS-Trust Server for a WS-Trust Token <input type="radio"/> Contact ClearTrust Server <input type="radio"/> Contact Netegrity SiteMinder <input type="radio"/> Contact NSS for SAF Authentication <input type="radio"/> Contact Tivoli Access Manager <input type="radio"/> Custom Template <input type="radio"/> Pass Identity Token to the Authorize Step <input type="radio"/> Retrieve SAML Assertions Corresponding to a SAML Browser Artifact <input type="radio"/> Use an Established WS-SecureConversation Security Context <input type="radio"/> Use certificate from BinarySecurityToken <input type="radio"/> Use DataPower AAA Info File <input type="radio"/> Use specified RADIUS Server <input type="radio"/> Validate a Kerberos AP-REQ for the Correct Server Principal <input type="radio"/> Validate the Signer Certificate for a Digitally Signed Message. <input type="radio"/> Validate the SSL Certificate from the Connection Peer  *	
<b>LDAP Load Balancer Group</b>	
<b>Host</b>	(none) <input type="button" value="+"/> <input type="button" value="..."/>
<b>Port</b>	389
<b>SSL Proxy Profile</b>	
<b>LDAP Bind DN</b>	<input type="text" value="cn=root"/>
<b>LDAP Bind Password</b>	<input type="password"/> <input type="password"/>
<b>LDAP Search Attribute</b>	
<b>LDAP Version</b>	v2 <input type="button" value="+"/> <input type="button" value="..."/>
<b>LDAP Search for DN</b>	<input type="radio"/> on <input checked="" type="radio"/> off
<b>LDAP Prefix</b>	
<b>LDAP Suffix</b>	<input type="text" value="dc=ibm,dc=com"/>
<b>User Auxiliary LDAP Attributes</b>	

\_\_ h. Click **Next**.

- \_\_\_ i. In the extract resource step, select **Local Name of Request Element** and click **Next**.
- \_\_\_ j. For authorizing the request, select **Check for Membership in an LDAP Group**.

The web page reloads with the LDAP configuration options.

\_\_ k. Enter the following information:

- **Host:** <backend\_server\_ip> (this field is already completed for you, from the information that is provided in a previous step)
- **Port:** 389
- **Group DN:** cn=grpDP<nn>,dc=ibm,dc=com, where *nn* = your student number
- **LDAP Bind DN:** cn=root
- **LDAP Bind Password:** websphere

<b>Method</b> <input type="radio"/> AAA Info File <input type="radio"/> Allow Any Authenticated Client <input type="radio"/> Always Allow <input checked="" type="radio"/> Check for Membership in an LDAP Group <input type="radio"/> Contact ClearTrust Server <input type="radio"/> Contact Netegrity SiteMinder <input type="radio"/> Contact NSS for SAF Authorization <input type="radio"/> Contact Tivoli Access Manager <input type="radio"/> Custom Template <input type="radio"/> Generate a SAML Attribute Query <input type="radio"/> Generate a SAML Authorization Query <input type="radio"/> Use SAML Attributes from Authentication <input type="radio"/> Use XACML Authorization Decision  <font>*</font>
<b>Host</b> <input type="text" value="192.168.1.100"/>
<b>Port</b> <input type="text" value="389"/>
<b>SSL Proxy Profile</b> <input type="text" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>
<b>Group DN</b> <input type="text" value="cn=grpDP&lt;nn&gt;,dc=ibm,dc=com"/> *
<b>LDAP Bind DN</b> <input type="text" value="cn=root"/>
<b>LDAP Bind Password</b> <input type="password" value="*****"/> <input type="password" value="*****"/>
<b>LDAP Load Balancer Group</b> <input type="text" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>
<b>LDAP Group Attribute</b> <input type="text" value="member"/>
<b>LDAP Version</b> <input type="text" value="v2"/> <input type="button" value="▼"/>
<b>LDAP Search Scope</b> <input type="text" value="Subtree"/> *
<b>LDAP Search Filter</b> <input type="text" value="(objectClass=*)"/>
<b>User Auxiliary LDAP Attributes</b> <input type="text"/>
<input type="button" value="Back"/> <input type="button" value="Next"/> <input type="button" value="Advanced"/> <input type="button" value="Cancel"/>

\_\_ l. Click **Next**.

\_\_ m. Click **Commit** and then click **Done**. The AAA policy object is created.

- \_\_\_ n. In the AAA action window, make sure the **AddressLDAP** AAA policy is selected and click **Done**.
  - \_\_\_ o. Click **Apply** to save the web service proxy configuration.
- \_\_\_ 7. Use cURL to test the LDAP AAA policy
- \_\_\_ a. Open a terminal window and navigate to the <lab\_files>/ldap directory.
  - \_\_\_ b. Enter the following cURL command:

```
curl --data-binary @retrieve-all.xml  
http://<dp_public_ip>:<wsp_proxy_port>/EastAddressSearch -u  
studentnn:websphere -H "Content-type: text/xml"
```



**Note**

The `-u` flag inserts the user name and password into the HTTP header. This policy can also be tested by using a user name token. When using HTTP basic authentication, you always use SSL to send the user name and password. For simplicity, you do not use SSL here.

- \_\_\_ c. Verify that you get the correct response (east coast addresses and people).
- \_\_\_ d. Change the password to an invalid value. The response displays a SOAP fault of “rejected by policy”.

## End of exercise

## Exercise review and wrap-up

In this exercise, you modified the East Address Search retrieveAll policy to authenticate and authorize users by using LDAP. The scenario was tested by invoking the East Address Search web service with HTTP basic authentication.



# Exercise 14. Implementing an SLM monitor in a web service proxy

## What this exercise is about

In this exercise, you specify SLM criteria to a web service proxy. You then send a series of web service requests, and observe the responses and log entries. To receive SLM-only log messages, you create a custom log target.

## What you should be able to do

At the end of the exercise, you should be able to:

- Specify service level monitoring criteria for a web service proxy
- Inspect and edit an SLM policy object
- Explain the need for an operation-level SLM action in a web service proxy
- Create a custom log target for SLM events

## Introduction

In an earlier exercise, you created an AddressSearchProxy web service proxy. A test script is provided to send cURL commands to the `findByName` and `findByLocation` operations multiple times. In the first section, you run the script to see the non-monitored responses. Since the SLM-related log messages might be difficult to find in the plethora of log messages, you then create a custom log target that captures only the SLM-related log messages. In another section, you use the SLM Policy tab of the web service proxy to specify specific traffic rates and sanctions to the `findByName` and `findByLocation` operations.

Again, you run the test script to observe the new behavior. As expected, the SLM monitoring manages the message traffic according to the specifications, but only for the `findByLocation` operation. In the subsequent section, an SLM action is added to the `findByName` request rule. The test script is run again, and all of the expected SLM monitoring occurs. Next, the SLM policy object is examined as a stand-alone object from the navigation bar of the WebGUI. A final optional section has you run the test script multiple times, and display the SLM Policy tab graph window.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- **cURL**, to send requests to the DataPower appliance
- The Address Search web services that are running on WebSphere Application Server
- Access to the `<lab_files>` directory

# Exercise instructions

## Preface

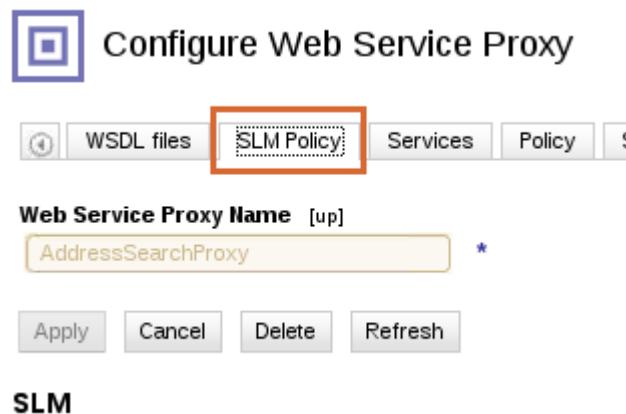
- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 9: Configuring a web service proxy.”
- Remember to use the domain and port address that you were assigned in the exercise setup. **Do not** use the default domain.
- The references in exercise instructions refer to the following values:
  - *<lab\_files>*: location of the student lab files
  - *<dp\_public\_ip>*: IP address of the public services on the appliance
  - *<wsp\_proxy\_port>*: port number of the web service proxy

## 14.1. Test the existing AddressSearchProxy by using the test script

A testing script `driveSLM.sh` is provided to simulate numerous invocations of the **findByName** and **findByLocation** operations in the **WestAddress** web service. The script sends a cURL command to each of the operations, “x” number of times. This script is a simple version of a load tester, which you must test for service level monitoring configuration.

In this section, you use the script to load the web service without any SLM statements in effect.

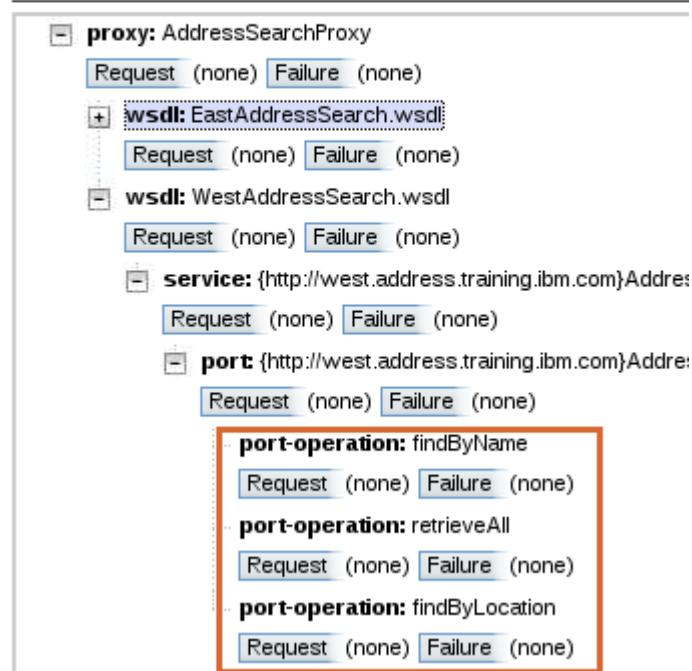
- \_\_\_ 1. Use **Troubleshooting** on the Control Panel to set the Log Level to **debug**. This level gives you the most detailed log entries.
- \_\_\_ 2. Open **AddressSearchProxy** from the Control Panel of the WebGUI.
- \_\_\_ 3. Select the **SLM Policy** tab.



4. Scroll down to the **Auto Generated SLM Statements** section. Expand the **WestAddressSearch** WSDL down to the port-operation level. Notice that there are no SLM statements entered.

#### Auto Generated SLM Statements

Define the request or failure SLM policy.



5. On the Linux desktop, open a file browser, and navigate to the `<lab_files>/SLM` folder.  
6. Use gedit to examine `driveSLM.sh`.

- \_\_\_ 7. The script takes three parameters: the IP address of the appliance public services, the port for the AddressSearchProxy service, and the number of times to send the two cURL commands.

```
echo "<script>"  
COUNT=$3  
MAX=$3  
INDEX=1  
while [[ $COUNT -gt 0 ]]  
do  
echo "<text>***** Iteration #$INDEX of $MAX  
*****</text>"  
echo "<iteration>"  
curl -s --data-binary @findByLocation.xml  
http://$1:$2/WestAddressSearch -H "SOAPAction: \"\""  
"Content-Type: text/xml"  
echo "<text> </text>"  
curl -s --data-binary @findByName.xml http://$1:$2/WestAddressSearch  
-H "SOAPAction: \"\""  
-H "Content-Type: text/xml"  
echo "</iteration>"  
  
(( COUNT -= 1 ))  
(( INDEX += 1 ))  
done  
echo "<text>Completed the iterations</text>"  
echo "</script>"
```

- \_\_\_ 8. The cURL commands invoke `findByLocation.xml` and `findByName.xml`. The `findByLocation` request is looking for addresses in Arizona (AZ), and `findByName` is looking for Ms Angela Reed.
- \_\_\_ 9. Notice the echoed text in the script to help make the output easier to read.
- \_\_\_ 10. Close the editor.
- \_\_\_ 11. Open a Terminal window.
- \_\_\_ 12. Change to the `<lab_files>/SLM` directory.
- \_\_\_ 13. Run the test script, pointing to your AddressSearchProxy, and tell it to send 10 pairs of cURL commands:
- ```
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 10
```
- \_\_\_ 14. Examine the results. You see 10 iterations of the responses.



## Information

An XML tool can help analyze the response that is written to a file, but since the response is actually a series of SOAP responses, the XML declaration `<?xml version="1.0" encoding="UTF-8"?>` comes up numerous times in the total response. An XML-aware tool sees the multiple occurrences as an error, and XML parsing fails.

## 14.2.Create a log target for SLM log messages

When many requests are sent to the web service, SLM-related messages can be hard to find in the system log. In this section, you create a log target that collects only SLM log messages.

- \_\_\_ 1. In the DataPower WebGUI, enter `log` in the navigation bar search field.
- \_\_\_ 2. Select **Log Target** in the choices. This results in a Configure Log Target page.
- \_\_\_ 3. Click **Add**.
- \_\_\_ 4. Set the following values on the **Main** tab:
  - \_\_\_ a. Name: `SLMtarget`
  - \_\_\_ b. Target Type: `File`
  - \_\_\_ c. Log Format: `XML`
  - \_\_\_ d. Identical Event Detection: `off`
  - \_\_\_ e. File Name: `logtemp:///SLMtarget.log`

The screenshot shows the 'Main' tab configuration for a log target named 'SLMtarget'. The 'Name' field is set to 'SLMtarget'. Under 'General Configuration', the 'Administrative State' is set to 'enabled'. The 'Comments' field contains 'log for SLM only'. The 'Target Type' is set to 'File'. The 'Log Format' is set to 'XML'. The 'Timestamp Format' is set to 'syslog'. Under 'Feedback Detection', the 'on' radio button is selected. Under 'Identical Event Detection', the 'off' radio button is selected. At the bottom, the 'File Name' is specified as 'logtemp:///SLMtarget.log'.

| Setting                   | Value   |
|---------------------------|---|
| Name                      | SLMtarget   |
| General Configuration     |   |
| Administrative State      | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments                  | log for SLM only  |
| Target Type               | File  |
| Log Format                | XML   |
| Timestamp Format          | syslog  |
| Feedback Detection        | <input type="radio"/> on <input checked="" type="radio"/> off           |
| Identical Event Detection | <input type="radio"/> on <input checked="" type="radio"/> off           |
| Destination Configuration |   |
| File Name                 | logtemp:///SLMtarget.log  |

- \_\_\_ 5. Set the following values on the **Event Subscriptions** tab:
  - \_\_\_ a. Event Category: `slm`

- \_\_\_ b. Minimum Event Priority: debug

| Event Category | Minimum Event Priority |
|----------------|------------------------|
| slm            | debug                  |

**Event Subscriptions**

**Name:** SLMtarget \*

**Apply** **Cancel**

- \_\_\_ 6. Click **Apply**.
- \_\_\_ 7. Click **Apply** again.
- \_\_\_ 8. Observe any log entries in the new log target.
- \_\_\_ a. In the WebGUI, click **Status > View Logs > System Logs**. The default system log displays.
- \_\_\_ b. In the Terminal window, resend the script command to send the cURL commands 10 times:
- ```
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 10
```
- \_\_\_ c. On the System Log page, click **Refresh Log**. The entries update.
- \_\_\_ d. For the **Target** field, change it from **default-log** to **SLMtarget**.



If a custom log target is defined with a Log Format of **XML**, the log is included in the Target list, and can be viewed.

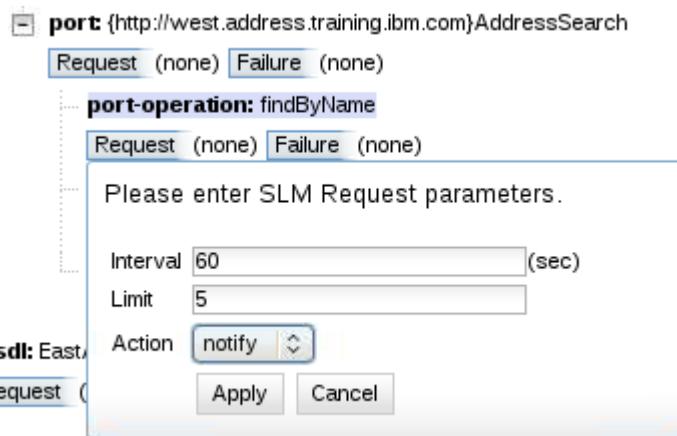
- \_\_\_ e. The SLMtarget log is still empty, since there is no SLM criteria yet. Switch the log back to **default-log**.

## 14.3.Add SLM criteria to the web service proxy

Although there is an **SLM action** in the default proxy-level request rule, there are no criteria that are set by default. You must add that if you want to manage the traffic.

In the **WestAddress** WSDL, you want to be notified when the `findByName` operation exceeds 5 requests per minute, and you want to reject `findByLocation` requests when the rate hits 5 requests per minute.

- \_\_\_ 1. In the DataPower WebGUI, open the **AddressSearchProxy**.
- \_\_\_ 2. Select the **SLM Policy** tab.
- \_\_\_ 3. Expand **WestAddressSearch.wsdl** until the port-operations are visible.
- \_\_\_ 4. Click **Request** under the `findByName` port-operation.
- \_\_\_ 5. In the dialog, enter an Interval of **60**, a Limit of **5**, and an Action of **notify**.



- \_\_\_ 6. Click **Apply**.
- \_\_\_ 7. Click **Request** under the `findByLocation` port-operation.
- \_\_\_ 8. In the dialog, enter an Interval of **60**, a Limit of **5**, and an Action of **throttle**.
- \_\_\_ 9. Click **Apply**.

10. Click **Apply** at the top of the page for the web service proxy.

The screenshot shows the Service Level Monitor (SLM) configuration interface. At the top, there are four buttons: 'Apply' (highlighted with a red box), 'Cancel', 'Delete', and 'Refresh'. Below the buttons, the title 'SLM' is displayed. A descriptive text states: 'Use this pane to define service level monitor (SLM) policies to comply with your implemented standards'. Under the heading 'Auto Generated SLM Statements', it says: 'Define the request or failure SLM policy.' The configuration tree is shown on the left:

- proxy: AddressSearchProxy
  - Request (none) Failure (none)
- wsdl: WestAddressSearch.wsdl
  - Request (none) Failure (none)
- service: {http://west.address.training.ibm.com}AddressSearchService
  - port: {http://west.address.training.ibm.com}AddressSearch
    - Request (none) Failure (none)
    - port-operation: findByName
      - Request Interval=60,Limit=5,Action=notify Failure none Graph
    - port-operation: retrieveAll
      - Request (none) Failure (none)
    - port-operation: findByLocation
      - Request Interval=60,Limit=5,Action=throttle Failure none Graph

11. Click **Save Config**.

## 14.4. Run the test script with SLM criteria in effect

The AddressSearchProxy now has SLM criteria that are specified for the `findByName` and `findByLocation` operations in the WestAddress WSDL. Run the test script to observe the results in the logs.

- \_\_\_ 1. In the WebGUI, open the system logs: **Status > View Logs > System Logs**.
- \_\_\_ 2. In the Terminal window, resend the script command to send the cURL commands 10 times:  

```
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 10
```
- \_\_\_ 3. Examine the response in the Terminal window. Notice that the response for the `findByLocation` request returns a SOAP fault that starts at about iteration 6:

```
<text>***** Iteration #6 of 10 *****</text>
<iteration>
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"><env:Body><env:Fault><faultcode>env:Client</faultcode><faultstring>Rejected by SLM Monitor (from client)</faultstring></env:Fault></env:Body></env:Envelope><text> </text>
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs d="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Header/><soapenv:Body><p726:findByNameResponse xmlns:p726="http://west.address .training.ibm.com"><findByNameReturn><details><city>Phoenix</city><state>AZ</sta te><street>4434 Elm Road</street><zipCode>85048</zipCode></details><name><firstName>Angela</firstName><lastName>Reed</lastName><title>Ms</title></name></findByNameReturn></p726:findByNameResponse></soapenv:Body></soapenv:Envelope></iteratio n>
```

This fault occurs because the `findByLocation` SLM criteria takes effect.

- \_\_\_ 4. On the System Log page in the WebGUI, click **Refresh Log**. The entries update.

5. Scroll down to find the red error entries for the SLM action (you might find it necessary to set **Show all**):

rror	52590241	error	172.16.80.115	0x02430001	wsgw (AddressSearchProxy): Message throttled
rror	52590241	request	172.16.80.115	0x80c00009	wsgw (AddressSearchProxy): request AddressSearchProxy_default_request-rule #1 slm: 'INPUT AddressSearchProxy failed: Rejected by SLM Monitor
rror	52590241	request	172.16.80.115	0x80c00010	wsgw (AddressSearchProxy): Execution of 'store:///dp/slmpolicy.xls' aborted: Rejected by SLM Monitor
warn	52590241	request	172.16.80.115	0x80e003d4	wsgw (AddressSearchProxy): SLM Throttle: Rejected by SLM Monitor
rror	52590241	request	172.16.80.115	0x01d30004	wsgw (AddressSearchProxy): Reject by SLM
ebug	52590241	request	172.16.80.115		slm-policy (AddressSearchProxy): statement 2 (Auto Generated) triggered reject action throttle
ebug	52590241	request	172.16.80.115		slm-policy (AddressSearchProxy): Identifier 2 resource type(wsdl-operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByLocation) using match type(exact-match) matches(true)
ebug	52590241	request	172.16.80.115		slm-policy (AddressSearchProxy): Identifier 1 resource type(wsdl-operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByLocation) using match type(exact-match) matches(false)

Also, notice the entries in the highlight box that indicate the evaluation of the SLM statements.



### Information

Another way to filter the log in this situation is to set the Filter field to **slm**. That limits the displayed log entries to just those entries that relate to the **slm** category.

6. Do a **Find** in the browser page (Ctrl+F) to find any entries for the **notify** action. There are none.

- \_\_\_ 7. For the **Target** field, change it from **default-log** to **SLMtarget**.

## System Log

Help

Refresh Log Target: SLMtarget Filter: (none) (none)

at: 20:40:56 on 2012-09-11

category level tid direction client msgid message Show last 50 100

1.2012

category	level	tid	direction	client	msgid	message
slm	warn	53011073	request	172.16.80.115	0x80e003d4	wsgw (AddressSearchProxy): SLM Throttle: Rejected by SLM Monitor
slm	debug	53011073	request	172.16.80.115		slm-policy (AddressSearchProxy): statement 2 (Auto Generated triggered reject action throttle)
slm	debug	53011073	request	172.16.80.115		slm-policy (AddressSearchProxy): Identifier 2 resource type(wsd operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByLocation) using match type(exact-match) matches(true)
slm	debug	53011073	request	172.16.80.115		slm-policy (AddressSearchProxy): Identifier 1 resource type(wsd operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByLocation) using match type(exact-match) matches(false)
slm	warn	53011041	request	172.16.80.115	0x80e003d4	wsgw (AddressSearchProxy): SLM Throttle: Rejected by SLM Monitor

This log target contains entries that are related to the slm category only.

- \_\_\_ 8. Notice that there are no entries that are related to the `findByName` requests, which the SLM criteria should generate. That situation is investigated in the next section.

## 14.5.Add an SLM action to a port-operation request rule

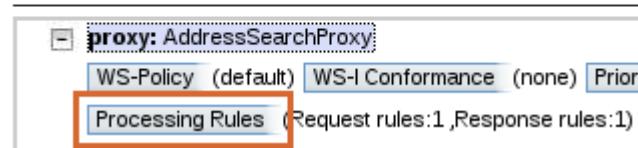
The `findByLocation` operation does not have its own request rule defined. Nor does it have a request rule that is defined at any higher level, except at the proxy level. For `findByLocation`, the SLM action in the default proxy level request rule is in effect.

However, the `findByName` operation already has its own request rule. It does not contain an SLM action. Therefore, when this operation is invoked, it uses the rule at the operation level that does not contain an SLM action. No SLM criteria are applied to any `findByName` requests.

- \_\_\_ 1. In the WebGUI, open the **AddressSearchProxy** page.
- \_\_\_ 2. Select the **Policy** tab.
- \_\_\_ 3. Select **Processing Rules** at the proxy level.

### WSDL Policy Tree Representation

Define the policies to apply in the tree.

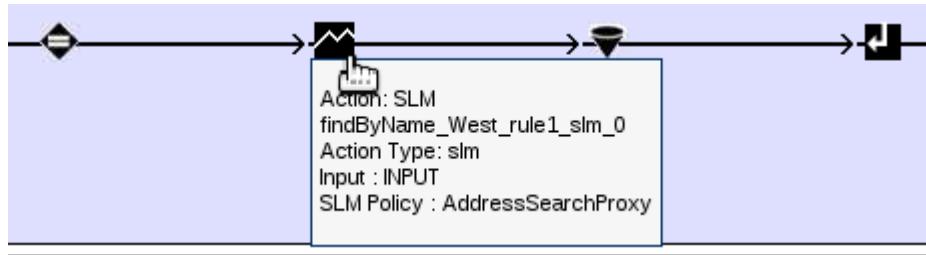


- \_\_\_ 4. Near the bottom of the page, the policy editor opens. Notice the **SLM action** in the default rule. This action is the SLM action that affects the `findByLocation` requests.
- \_\_\_ 5. Hover the mouse over the SLM action. Notice that the name of the SLM policy is **AddressSearchProxy**, which is the default name that DataPower assigns.
- \_\_\_ 6. Back up in the policy tree, expand the WestAddress WSDL branch.
- \_\_\_ 7. Select **Processing Rules** beneath the `findByName` operation.



- \_\_\_ 8. In the policy editor at the bottom of the page, the rule displays. Drag an **SLM** action after the Match action.

- \_\_\_ 9. Hover the mouse over the SLM action. Notice that the SLM policy defaults to **AddressSearchProxy**. This situation is what you want.

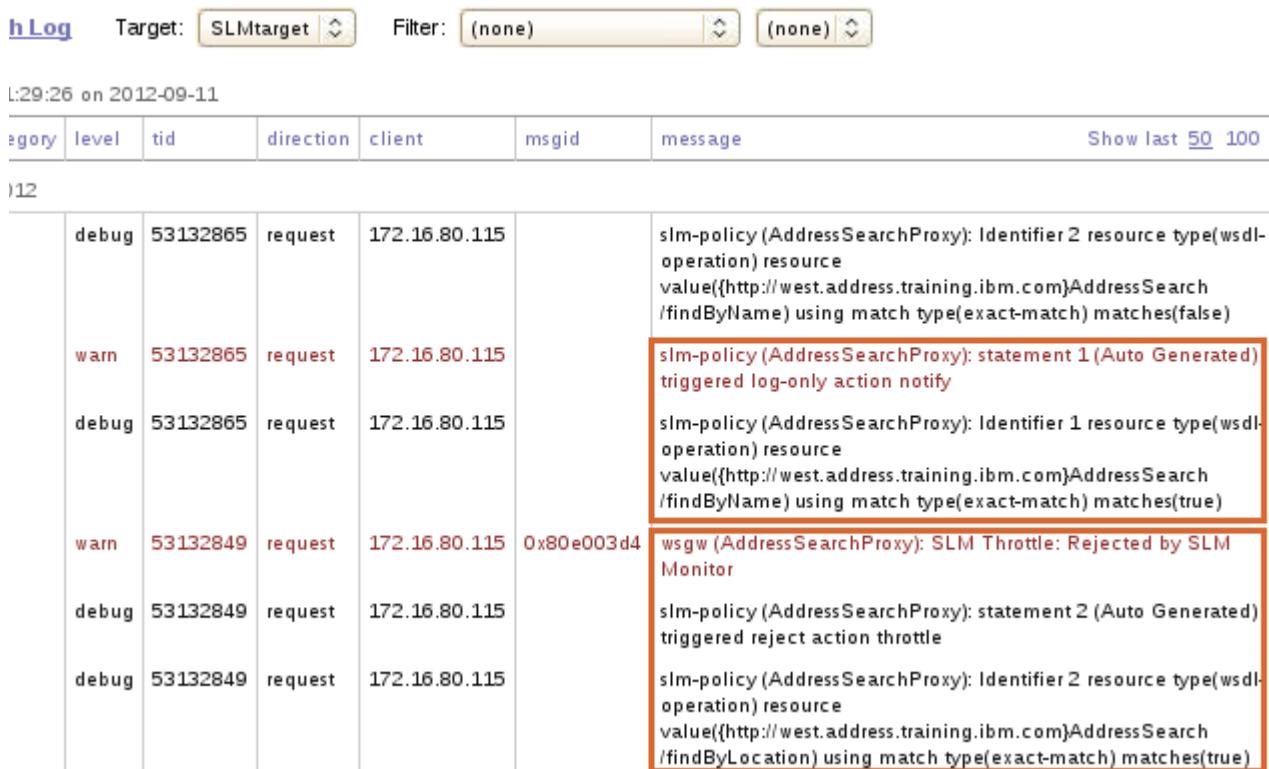


- \_\_\_ 10. Click **Apply** at the top of the page.

## 14.6.Run the test script with the operation-level SLM action

The `findByName` requests now are subject to the `AddressSearchProxy` SLM policy. Test it to verify the expected results.

- \_\_\_ 1. In the WebGUI, switch to the **System Log** page.
- \_\_\_ 2. Set the Target as **SLMtarget**.
- \_\_\_ 3. In the Terminal window, run the script command:  
`./driveSLM.sh <dp_public_ip> <wsp_prox_port> 10`
- \_\_\_ 4. The results should again show successful `findByName` requests, with some failing `findByLocation` requests.
- \_\_\_ 5. In the WebGUI **System Log** page, click **Refresh Log**.
- \_\_\_ 6. Although the `findByName` requests completed successfully, as seen in the Terminal window, the SLM action of **notify** is now also in effect.



The screenshot shows the System Log page with the following details:

- Log Target:** SLMtarget
- Filter:** (none)
- Time:** 12:29:26 on 2012-09-11
- Columns:** category, level, tid, direction, client, msgid, message
- Actions:** Show last 50 100

The log entries are as follows:

category	level	tid	direction	client	msgid	message
debug	53132865	request	172.16.80.115			slm-policy (AddressSearchProxy): Identifier 2 resource type(wsdl-operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByName) using match type(exact-match) matches(false)
warn	53132865	request	172.16.80.115			slm-policy (AddressSearchProxy): statement 1 (Auto Generated) triggered log-only action notify
debug	53132865	request	172.16.80.115			slm-policy (AddressSearchProxy): Identifier 1 resource type(wsdl-operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByName) using match type(exact-match) matches(true)
warn	53132849	request	172.16.80.115	0x80e0003d4		wsgw (AddressSearchProxy): SLM Throttle: Rejected by SLM Monitor
debug	53132849	request	172.16.80.115			slm-policy (AddressSearchProxy): statement 2 (Auto Generated) triggered reject action throttle
debug	53132849	request	172.16.80.115			slm-policy (AddressSearchProxy): Identifier 2 resource type(wsdl-operation) resource value({http://west.address.training.ibm.com}AddressSearch /findByLocation) using match type(exact-match) matches(true)

- \_\_\_ 7. Click **Save Config**.

## 14.7. View the SLM policy object

An SLM policy object can be created, edited, and deleted as a stand-alone object. In this section, you have an opportunity to examine this object.

- \_\_\_ 1. On the WebGUI, select **Objects > Monitoring > SLM Policy**.

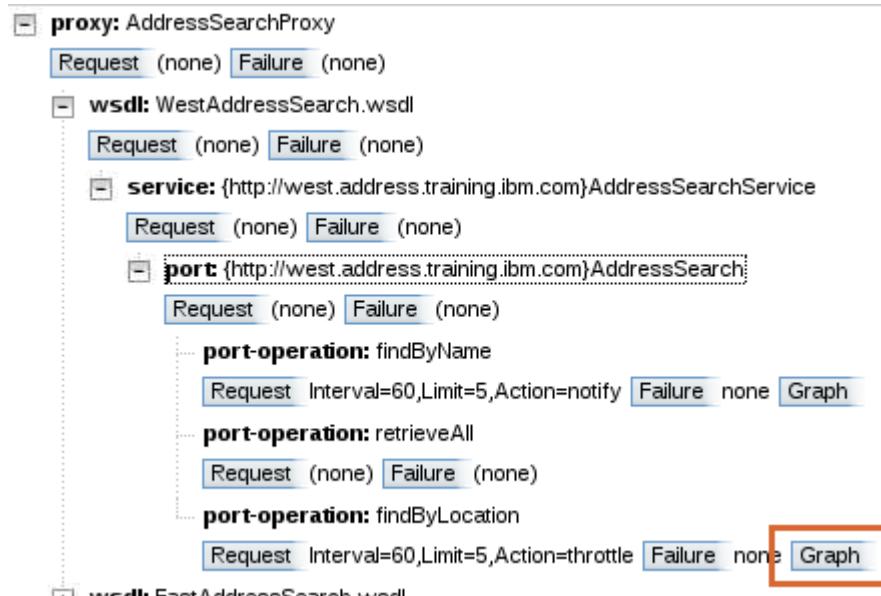
This selection displays the catalog list of the SLM policy objects that are defined in this domain. By default, a web service proxy creates an SLM policy object with the same name as the web service proxy. For a multi-protocol gateway, there is no default SLM policy object, so you must explicitly create one. This situation can occur during the configuration of an SLM action, or can occur on this page.

- \_\_\_ 2. Select **AddressSearchProxy**, the SLM policy object that the web service proxy creates.
- \_\_\_ 3. On the **Main** tab, you specify the evaluation method, and potentially identify an SLM peer group.
- \_\_\_ 4. The **Statement** tab controls the SLM statements within this SLM policy object. SLM statements can be created, edited, or deleted from this tab. This tab also lists any SLM statements that were auto-generated from the **SLM Policy** tab in the web service proxy. Clicking **Add** opens a dialog to create an SLM statement.
- \_\_\_ 5. When you are finished reviewing the SLM policy object, return to the catalog list page.

## 14.8.Examine the graph behavior (optional)

In this section, you examine the SLM graph tool that is part of the web service proxy configuration. This tool is a handy tool for simple SLM testing in the development environment.

- \_\_\_ 1. Open the **AddressSearchProxy** web service proxy in the WebGUI.
- \_\_\_ 2. Select the **SLM Policy** tab.
- \_\_\_ 3. Expand the **WestAddressSearch** WSDL down to the port-operation level.
- \_\_\_ 4. Click the **Graph** button for the **findByLocation** operation.

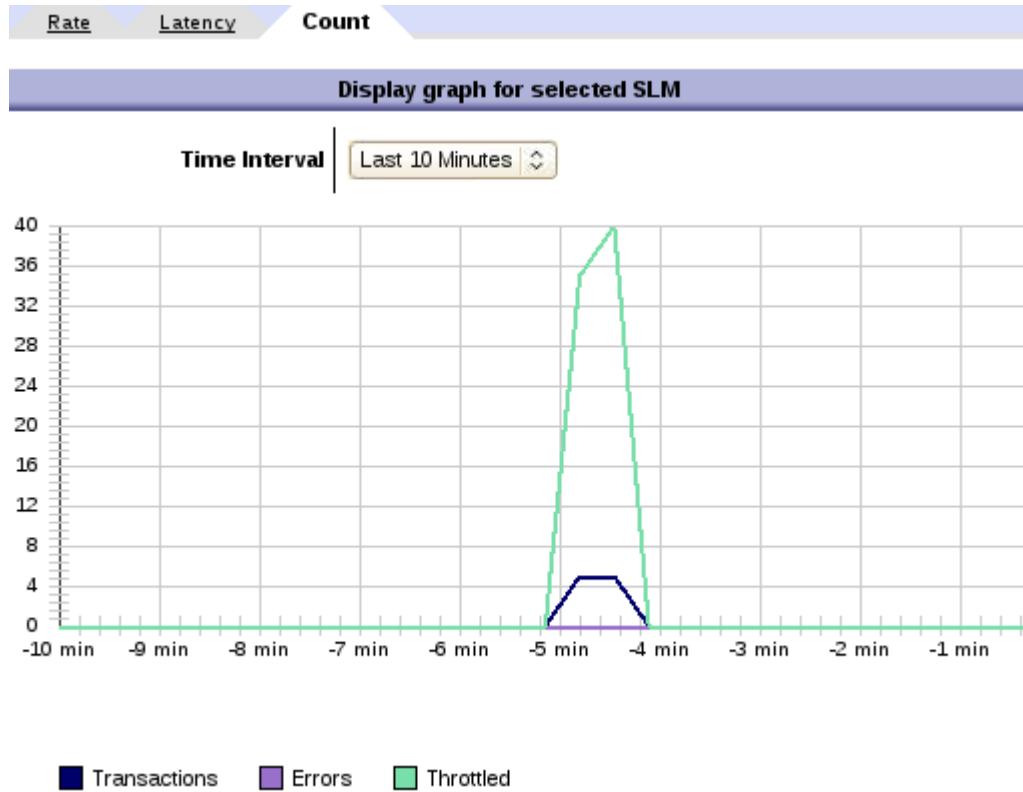
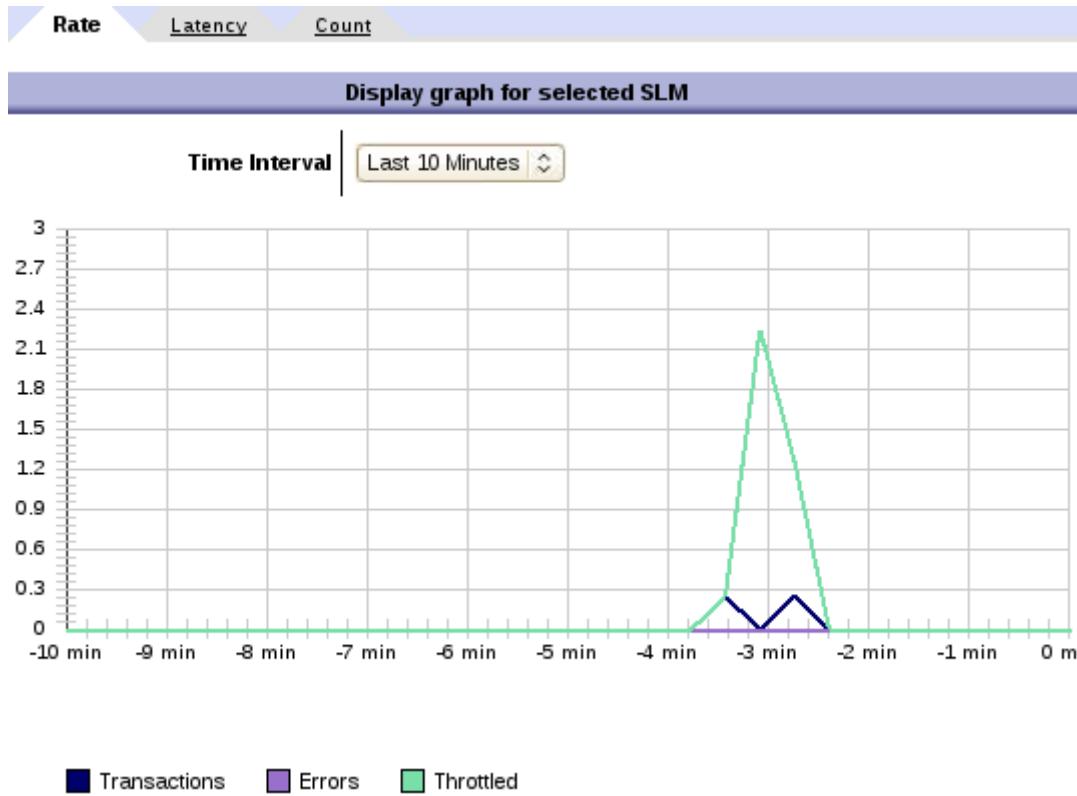


- \_\_\_ 5. In the Terminal window, run the test script numerous times in succession, varying the number of times the cURL commands are sent:
- ```

./driveSLM.sh <dp_public_ip> <wsp_prox_port> 10
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 30
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 15
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 5
./driveSLM.sh <dp_public_ip> <wsp_prox_port> 25

```
- \_\_\_ 6. Click **Refresh** in the graph window.

7. Select the different tabs to see the results. You can see the throttling occur.



- \_\_\_ 8. Do any further load testing that you want. When you are finished, close the graph window.
- \_\_\_ 9. Be sure that you save the configuration before you exit the WebGUI.

## **End of exercise**

## Exercise review and wrap-up

In this exercise, you modified the AddressSearchProxy web service proxy to enforce SLM criteria. You used auto-generated SLM statements to configure the SLM policy automatically, and you applied them at both the proxy level and at the operation level. A custom log target was created to capture the SLM-specific log messages.

# Exercise 15.Configuring a multi-protocol gateway service with WebSphere MQ

## What this exercise is about

This exercise shows you how to add support for WebSphere MQ to a multi-protocol gateway (MPGW) service. You add a WebSphere MQ front-side handler to the EastAddressSearch service that you created in an earlier exercise. You create another MPGW service to demonstrate one-way messaging to a back-end WebSphere MQ system. This MPGW service is used as a WebSphere MQ client, similar to the WebSphere MQ client RFHUtil, to get and put messages from queues. Finally, you learn about the transaction capabilities of the DataPower and WebSphere MQ integration.

## What you should be able to do

At the end of the exercise, you should be able to:

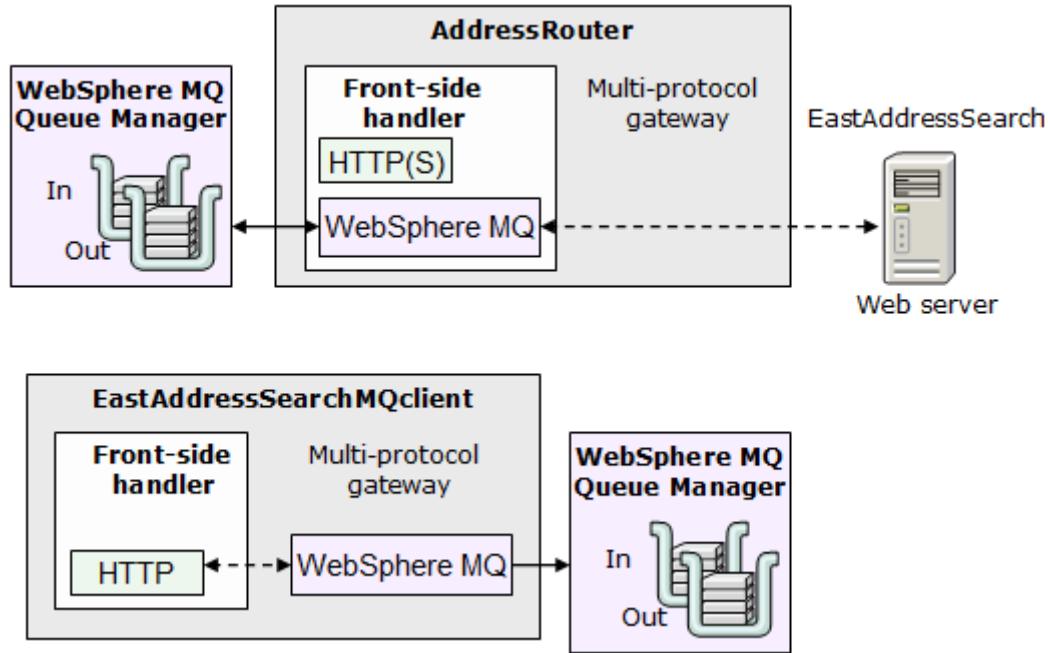
- Create a WebSphere MQ front-side handler (FSH) that gets messages from a queue and puts responses on a queue
- Send messages from a multi-protocol gateway service to a queue in WebSphere MQ in a fire-and-forget messaging pattern
- Configure transactionality between WebSphere DataPower and WebSphere MQ when errors occur during message processing

## Introduction

To support reliable, asynchronous messaging, enterprises use WebSphere MQ. A WebSphere MQ client can get and put messages onto queues in WebSphere MQ. DataPower features an enhanced version of the WebSphere MQ client implementation, which allows it to be configured through the DataPower management interface.

In this case study, the `EastAddressSearch` web service can be called over HTTP. WebSphere MQ can be used to support asynchronous messaging to the `EastAddressSearch` web service. The DataPower WebSphere MQ implementation can be used to bridge the gap between these two protocols. Requests for the `EastAddressSearch` web service can be put on a queue, which DataPower can retrieve, run a service policy, and send to the back-end `EastAddressSearch` web service over HTTP.

You work with two multi-protocol gateways (MPGW). The first MPGW, `EastAddressSearch`, is imported at the beginning of the exercise. The second MPGW, `EastAddressSearchMQclient`, is created and used to simulate a WebSphere MQ client. This MPGW is used to put requests onto a queue in WebSphere MQ, which the `EastAddressSearch` service then retrieves for processing. The `EastAddressSearchMQclient` service is also used to get responses from a queue to verify that the `EastAddressSearch` processed the message correctly.



## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Access to a WebSphere MQ system with a queue manager, three queues (`EASTADDRESSQUEUEIN<nn>`, `EASTADDRESSQUEUEOUT<nn>`, `EASTADDRESSQUEUEERROR<nn>`), and a channel (`EASTADDRESS.CHANNEL`)
- **cURL**, to send requests to the DataPower appliance
- The Address Search web services that runs on WebSphere Application Server
- Access to the `<lab_files>` directory.

# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 13: Configure a multi-protocol gateway service.”
- 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 the exercise instructions refer to the following values:
  - *<mpgw\_mq\_client\_port>*: multi-protocol gateway port for sending requests to WebSphere MQ

## 15.1. Obtain the WebSphere MQ configuration

The course image is already installed with WebSphere MQ and is configured with a queue manager, queue, channel, cluster, and listener port. Each student is assigned a set of queues, where they put and get messages. Typically, a DataPower integration developer works with the WebSphere MQ team to obtain the information to connect to WebSphere MQ. In this section, the information about your WebSphere MQ configuration is provided.

A WebSphere MQ server is configured with a queue manager `QM_base`. In this queue manager, several queues are defined. Each student has three queues: an input queue, an output queue, and an error queue. The name of the queue that you use depends on your student number. For example, if you are `student01`, you are using the queues `EASTADDRESSQUEUEIN01`, `EASTADDRESSQUEUEOUT01`, and `EASTADDRESSQUEUEERROR01`. All students use the same server channel, `EASTADDRESS.CHANNEL`, to put and get messages from your queues. Similarly, all students use the port `1414` installed by default as the listener port.

The information is summarized here.



### Information

In the next section, you play the role of the DataPower integration developer, who connects to WebSphere MQ with this information.

- WebSphere MQ host name: `<backend_server_ip>`
- WebSphere MQ port: `1414`
- WebSphere MQ Queue Manager: `QM_base`
- WebSphere MQ server channel: `EASTADDRESS.CHANNEL`
- WebSphere MQ user name: `mqm`
- Get queue: `EASTADDRESSQUEUEIN<nn>`
- Put queue: `EASTADDRESSQUEUEOUT<nn>`
- Error queue: `EASTADDRESSQUEUEERROR<nn>`

The variable `<nn>` represents your student number.

## 15.2.Create a WebSphere MQ front side handler (FSH)

In this section, you use the configuration information in the previous section to create a WebSphere MQ queue manager object. The WebSphere MQ queue manager object can be used both as a front side handler and for defining the back-end URL. It is created by referencing an existing WebSphere MQ queue manager object and defining the GET and PUT queues.

- \_\_\_ 1. In the DataPower WebGUI **Control Panel**, click **Multi-Protocol Gateway**.
- \_\_\_ 2. Click **EastAddressSearch**.
- \_\_\_ 3. Create a WebSphere MQ queue manager object.

You create a WebSphere MQ queue manager object inside the WebSphere MQ front side handler window.

- \_\_\_ a. In the **EastAddressSearch** configuration page, under “Front side settings,” click **+ (new)** and select **WebSphere MQ Front Side Handler**.
- \_\_\_ b. Enter a name of: `EastAddressMQFSH`
- \_\_\_ c. In the “WebSphere MQ Front Side Handler” window, click **+ (new)** to create a WebSphere MQ queue manager object.
- \_\_\_ d. Select **Create a WebSphere MQ Queue Manager**.
- \_\_\_ e. In the WebSphere MQ queue manager object, complete the following information that is based on the configuration in the previous section (notice that this information is case-sensitive):

**Main tab:**

- **Name:** `EastAddressQM`
- **Host name:** `<backend_server_ip>(1414)`
- **Queue manager name:** `QM_base`
- **Channel name:** `EASTADDRESS.CHANNEL`
- **User name:** `mqm`
- **Alternate user:** `off`

**Connections tab:**

- **Automatic retry:** `off`

The Host name, Queue manager name, and Channel name fields are required fields for sending and receiving information from a WebSphere MQ queue manager object.

**Note**

You do not specify the GET and PUT queues within this object. Although the queue manager, and the queues as well, are members of a cluster, you still point to the queue manager for access. The queue manager detects any requests and passes them to the load-balancing mechanism in the cluster.

The **User Name** field is also required when communicating with a WebSphere MQ queue manager object because it identifies a user with administrative privileges on the local operation system. When you install WebSphere MQ in Windows, a default user that is called **MUSR\_MQADMIN** in the **mqm** user group is created. For Linux, a default user of **mqm** is created. If you specify a user who does not have administrative privileges, you are unable to connect to the WebSphere MQ queue manager.

The **SSL Key Repository** and **SSL Cipher Specification** settings are used to support SSL communication with WebSphere MQ. In this example, you do not use SSL.

The **Automatic Retry** is a flag on the WebSphere MQ client. It is used to continuously try connecting based on the retry interval if the WebSphere MQ client connection is broken. For simplicity, automatic retry is turned it off.

**Alternate User**, when **on** (default), specifies that WebSphere MQ uses the name in the WebSphere MQ header **MQMD.AlternateUserId** to complete authentication.

- \_\_\_ f. Click **Apply**. The WebSphere MQ queue manager object window closes.
- \_\_\_ g. In the “WebSphere MQ Front Side Handler” window, verify that the **Queue Manager** field is populated with: `EastAddressQM`
- \_\_\_ h. Enter the **Get Queue** (`EASTADDRESSQUEUEIN<nn>`) and **Put Queue** (`EASTADDRESSQUEUEOUT<nn>`), where `<nn>` is your student number.

**Important**

Each student has a unique GET and PUT queue that is based on the student number (this example is for student 01).

|   |  |
|---|--|
| Name                                    | EastAddressMQFSH   |
| <b>General</b>                          |  |
| Administrative State                    | <input checked="" type="radio"/> enabled <input type="radio"/> disabled  |
| Comments                                |  |
| Queue Manager                           | EastAddressQM <input type="button" value="..."/> <input type="button" value="+"/> <input type="button" value="..."/> * |
| Get Queue                               | EASTADDRESSQUEUEIN01   |
| Put Queue                               | EASTADDRESSQUEUEOUT01  |
| The number of concurrent MQ connections | 1  |
| Get Message Options                     | 4097   |
| Polling Interval                        | 30   |

Be sure to specify your student number in the queue names.

- \_\_\_ i. For the **Admin State**, click **disabled**. This FSH is enabled when you are ready to get and put messages from the queues.

**Information**

The `EastAddressMQFSH` is purposely disabled so that it does not process messages on the queues. You enable it when a message is put on the `EASTADDRESSQUEUEIN<nn>` so that you can see the response from the back-end web service.

- \_\_\_ j. Click **Apply**. The “WebSphere MQ Front Side Handler” window closes.
- \_\_\_ k. Verify that the new handler is now in the front side protocol list.

\_\_ I. Click **Apply**.

The web page refreshes, and the object status of the `EastAddressMQFSH` is shown. When added to the gateway, the DataPower WebSphere MQ client tries to connect to the WebSphere MQ queue manager. If the connection fails, the object status is down.



**Important**

The object status of the `EastAddressMQFSH` is **down** because you disabled it. You can enable it to make sure that the DataPower WebSphere MQ client can connect to the WebSphere MQ queue manager. Make sure that you leave the state at **disabled** after you are finished testing.

## 15.3.Create a multi-protocol gateway that completes one-way messaging

In this section, you create a multi-protocol gateway service that sends and receives messages from queues on WebSphere MQ. This multi-protocol gateway service uses one-way messaging, or “fire and forget”: sending messages without waiting for a response. This service is used in the rest of the exercise to put and get messages from a queue in WebSphere MQ. If you are familiar with **RFHUtil**, this service is analogous to that tool.

- \_\_\_ 1. In the DataPower WebGUI **Control Panel**, click **Multi-Protocol Gateway**.
- \_\_\_ 2. Click **Add**.
- \_\_\_ 3. Enter `EastAddressSearchMQclient` as the name of the new gateway. Optionally, you can enter a description for the gateway.



### Information

This multi-protocol gateway contains a service policy that reads the incoming HTTP headers to build the back-end WebSphere MQ URL. The request rule of this policy contains a **Route** action that references a style sheet, which builds the back-end DataPower WebSphere MQ URL.

For example, to call this service to **put** a message onto the queue `EASTADDRESSQUEUEIN01`, a client enters the following cURL command:

```
curl -H "Operation: PUT" -H "Queue: EASTADDRESSQUEUEIN01" -H
"Content-Type: text/xml" -data-binary @[fileName]
http://<dp_public_ip>:<mpgw_mq_client_port>/uri
```

It generates a WebSphere MQ back-end URL of the form:

```
dpmq://EastAddressQM/?RequestQueue=EASTADDRESSQUEUEIN01
```

The DataPower WebSphere MQ queue manager named `EastAddressQM` is already defined.

To invoke this service to **get** a message from the queue `EASTADDRESSQUEUEOUT01`, a client enters the following cURL command:

```
curl -H "Operation: GET" -H "Queue: EASTADDRESSQUEUEOUT01" -H
"Content-Type: text/xml" -data-binary @[fileName]
http://<dp_public_ip>:<mpgw_mq_client_port>/uri
```

It generates a WebSphere MQ back-end URL of the form:

```
dpmq://EastAddressQM/?ReplyQueue=EASTADDRESSQUEUEOUT01
```

- \_\_\_ 4. In the **Multi-Protocol Gateway Policy** field, click **+** (new).
- \_\_\_ 5. Enter a Policy Name of: `EastAddressSearchMQclientPolicy`

- \_\_\_ 6. Click **New Rule**.
- \_\_\_ 7. Double-click the **Match** action that was created.
- \_\_\_ 8. Select an existing match all rule or create a matching rule that matches all URLs (use steps from a previous exercise as a guide).
- \_\_\_ 9. Add a **Route** action to the pipeline in the service policy editor.
- \_\_\_ 10. Double-click the **Route** action. Make sure that **Use Stylesheet to Select Destination** is selected as the **Selection Method**.
- \_\_\_ 11. Upload the XSL style sheet `MQRoute.xsl` from the `<lab_files>/MQ` directory.

**Input**

|              |        |          |
|--------------|--------|----------|
| <b>Input</b> | (auto) | (auto) ▾ |
|--------------|--------|----------|

**Options**

◆ **Route (Using Stylesheet or XPath Express)**

|                                |  |
|--------------------------------|--|
| <b>Selection Method</b>        | <input checked="" type="radio"/> Use Stylesheet to Select Destination<br><input type="radio"/> Use XPath to Select Destination<br><input type="radio"/> Use Variable to Select Destination |
| <b>Processing Control File</b> | local:///MQRoute.xsl *   |
|                                | local:/// ▾ MQRoute.xsl ▾ Upload... ▾  |
| <b>Asynchronous</b>            | <input type="radio"/> on <input checked="" type="radio"/> off  |

**Output**

|               |          |
|---------------|----------|
| <b>Output</b> | (auto) ▾ |
|---------------|----------|

- \_\_\_ 12. Click **Done**.
- \_\_\_ 13. Add a **Results** action after the route action.
- \_\_\_ 14. For **Rule Direction**, select **Client to Server**. This policy contains a request rule only.
- \_\_\_ 15. Click **Apply Policy**, and then **Close Window**.
- \_\_\_ 16. In the service window, change the **Type** from **static-backend** to **dynamic-backend**.

**Note**

Since the service policy contains a **Route** action that determines the back-end URL, you change the service type to **dynamic-backend**.

- \_\_\_ 17. Scroll down and select the **Response Type of Pass-Thru**.

This setting specifies that no response rule is run for the response message.

- \_\_\_ 18. Create an HTTP front side handler that is called `EastAddressSearchHTTPClient` and add it to the `EastAddressSearchMQclient` gateway.
- Under “Front side settings,” click **+** (new) and select **HTTP Front Side Handler**.
  - Enter the name `EastAddressSearchHTTPClient` and port number `<mpgw_mq_client_port>`.
  - Click **Apply**. The new handler is now in the Front Side Protocol list.
  - Click **Apply**.
- \_\_\_ 19. Test the MPGW WebSphere MQ client by using cURL to put a message on the `AddressQueueIn<nn>` queue.
- Open a terminal window and go to the `<lab_files>/MQ` directory.
  - Enter the command:

```
curl -H "Operation: PUT" -H "Queue: EASTADDRESSQUEUEIN<nn>"  
-H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_mq_client_port>
```



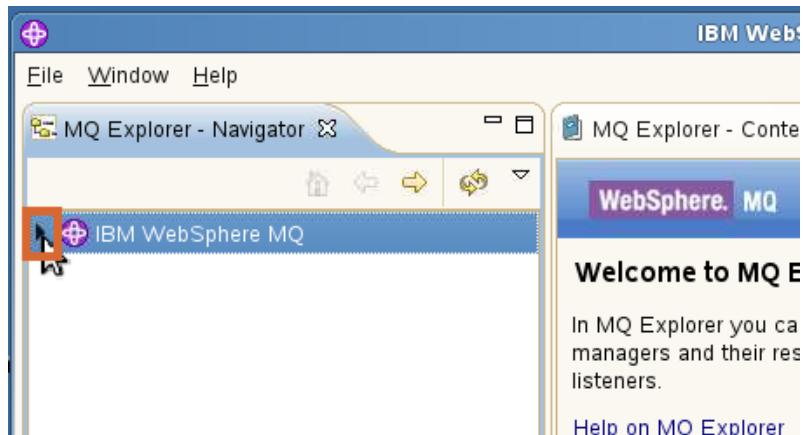
#### Note

This command puts the `findByLocation.xml` file onto the `EASTADDRESSQUEUEIN<nn>` without waiting for a reply. Therefore, the command finishes execution. Nothing is returned.

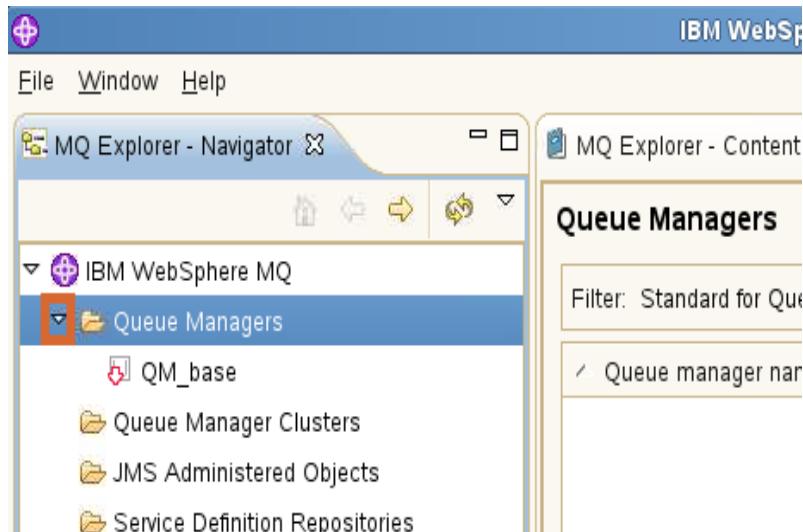
- Check the system logs to make sure that no errors were generated after running the command.
- \_\_\_ 20. Open WebSphere MQ Explorer to view the number of messages that exist on the different queues.
- Click the WebSphere MQ Explorer icon on the desktop.



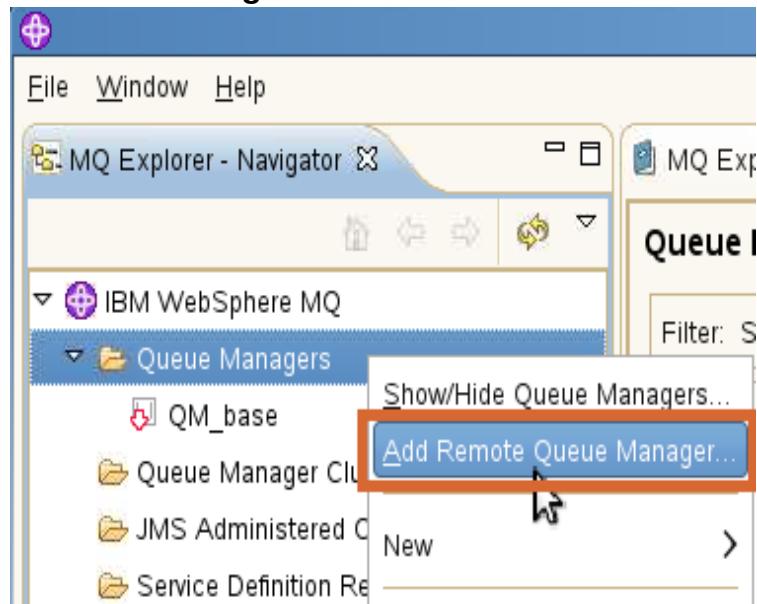
\_\_ b. Expand **IBM WebSphere MQ Explorer**.



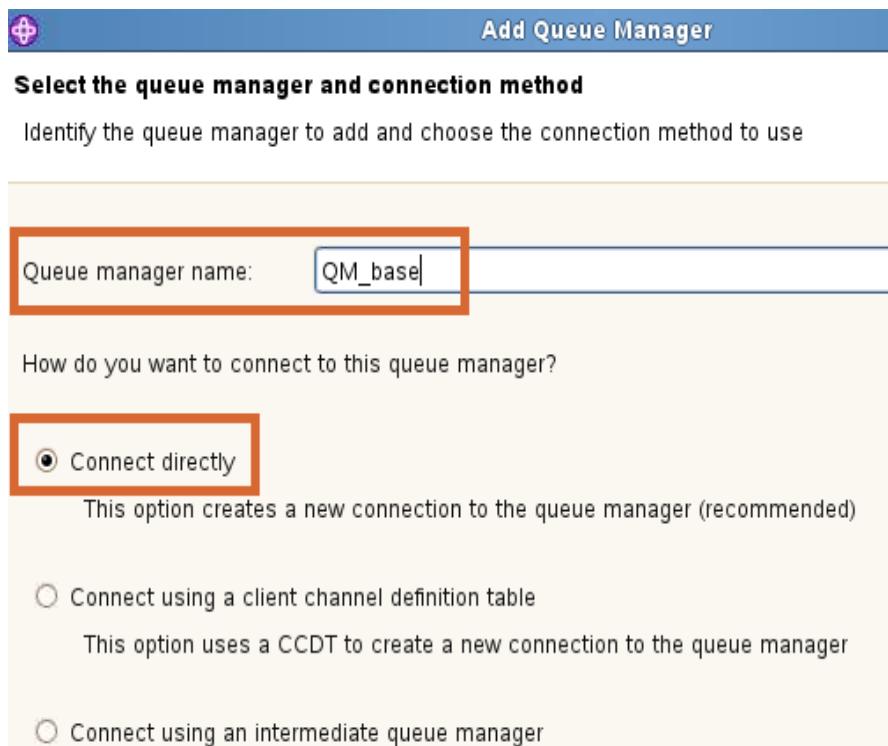
\_\_ c. Expand **Queue Managers**.



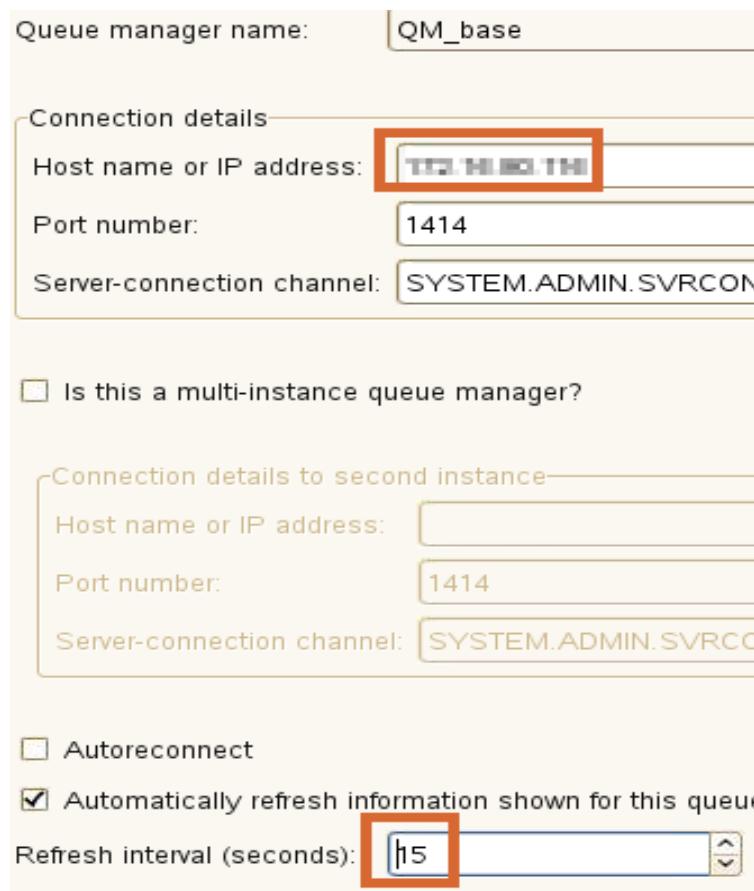
- \_\_\_ d. Right-click **Queue Managers** and select **Add Remote Queue Manager**.



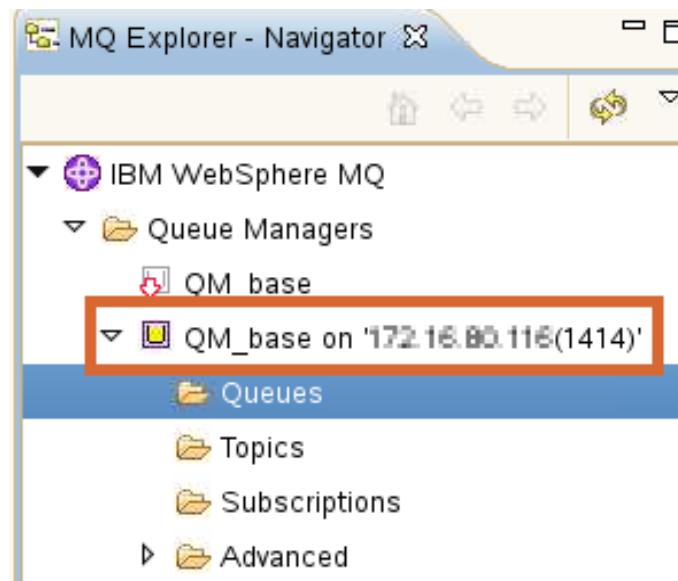
- \_\_\_ e. For the remote queue definition, enter the queue name QM\_base, and confirm the **Connect directly** is selected.



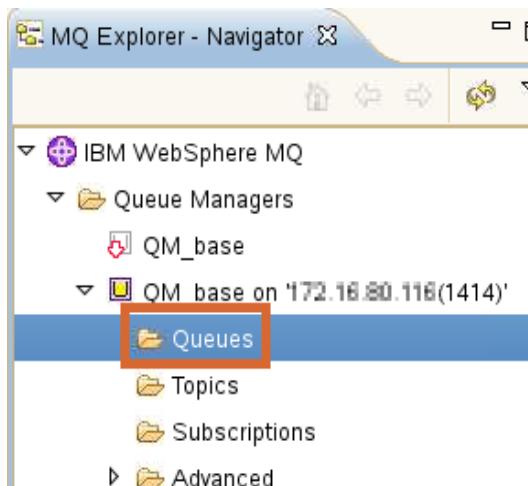
- \_\_ f. Enter the <backend\_server\_ip> address in the **Host name or IP address** section, and change the **Refresh interval (seconds)** to 15.



- \_\_ g. Click **Finish**.
- \_\_ h. The remote queue manager QM\_base is now shown as one of the queue managers.



- \_\_ i. Click the **Queues** folder that is located under the remote QM\_base to see all of the queue.



- \_\_ j. Scroll through the queue window that locates the correct items in queue. In this example, the queue for student01 is reviewed, EASTADDRESSIN01. By examining the queue depth, it confirms that a message was successfully put on the queue as there is a current queue depth of 1.

The screenshot shows the MQ Explorer - Content window with the title 'Queues'. A filter bar at the top says 'Filter: Standard for Queues'. Below is a table with the following data:

| Queue name                  | Queue type | Open input count | Open output count | Current queue depth |
|-----------------------------|------------|------------------|-------------------|---------------------|
| EASTADDRESSQUEUEERROR28     | Local      | 0                | 0                 | 0                   |
| EASTADDRESSQUEUEERROR29     | Local      | 0                | 0                 | 0                   |
| EASTADDRESSQUEUEERROR30     | Local      | 0                | 0                 | 0                   |
| <b>EASTADDRESSQUEUEIN01</b> | Local      | 0                | <b>1</b>          | <b>1</b>            |
| EASTADDRESSQUEUEIN02        | Local      | 0                | 0                 | 0                   |
| EASTADDRESSQUEUEIN03        | Local      | 0                | 0                 | 0                   |



### Information

If a message is not on the queue where you believe it should be, the 15-second refresh might not have occurred yet. To force an instant refresh, click the refresh icon that is located in the upper right corner of the WebSphere MQ Explorer content window. The refresh icon is two arrows that create a circle.

There are several ways to check on your queues:

- WebSphere MQ Explorer, if you have access to it.
- System log, to see whether any errors are generated during the request. You can also see info level message with the text: Issuing a request to URL: "dpmq://EastAddressQM/?..."
- The MQSC WebSphere MQ command. In a terminal window:
  - a. Enter `runmqsc QM_base` to start the MQSC command processor.
  - b. Enter `dis ql (EASTADDRESSQUEUExxxx) curdepth` to display the current depth of the indicated queue. You remain within the command processor after the reply, so you can enter more "dis ql" commands.
  - c. When you are finished with the command processor, enter `end` to terminate the command processor.

## 15.4. Use the WebSphere MQ FSH to call the EastAddressSearch web service

In this section, you use the `EastAddressSearch` WebSphere MQ FSH to call the `EastAddressSearch` web service over HTTP. The `EastAddressMQFSH` handler gets the request message from the `EASTADDRESSQUEUEIN<nn>` queue, which was sent in the last section by the MPGW WebSphere MQ client service. It runs the service policy, and forwards the request to the back-end web service over HTTP. If the back-end service returns a response, then the WebSphere MQ FSH places that response on the `EASTADDRESSQUEUEOUT<nn>` queue.

- \_\_\_ 1. In the `EastAddressSearch`, enable the **Admin State** of the `AddressSearchMQFSH` handler.



### Information

You disabled the **Admin State** so that this MPGW service would not automatically pick up the message from the `EASTADDRESSQUEUEIN<nn>`. You now enable it and verify its functionality.

- \_\_\_ a. Open the `EastAddressSearch` multi-protocol gateway configuration page.
- \_\_\_ b. Scroll down to front side settings and click **EastAddressMQFSH**. This action places the handler in the list.

| Front Side Protocol  |           |
|--|-----------|
| <code>AddressSearchMPG-HTTP (HTTP Front Side Handler)</code>         | X         |
| <code>AddressSearchMPG-HTTPS (HTTPS (SSL) Front Side Handler)</code> | X         |
| <code>EastAddressMQFSH (MQ Front Side Handler) [down]</code>         | X         |
| <code>EastAddressMQFSH (MQ Front Side Handler)</code>                | Add + ... |

- \_\_\_ c. Click [...] (edit).
- \_\_\_ d. Click **enabled** on the **Admin State** field, and then click **Apply** to start the WebSphere MQ FSH object.
- \_\_\_ e. Click **Apply** on the Configure Multi-Protocol Gateway window.



### Information

When enabled, the WebSphere MQ FSH of `EastAddressSearch` gets the message from the `EASTADDRESSQUEUEIN<nn>` queue, runs the service policy, and puts the response message from the back-end web service onto the `EASTADDRESSQUEUEOUT<nn>` queue.

- \_\_\_ f. Check the system logs to make sure that no errors occurred.

- \_\_\_ g. Check the **out** queue in WebSphere MQ Explorer to ensure that the message for your student number was moved from the **in** queue to the **out** queue.

| Queue name           | Queue type | Open input count | Open output count | Current queue depth |
|----------------------|------------|------------------|-------------------|---------------------|
| EASTADDRESSQUEUEIN30 | Local      | 0                | 0                 | 0                   |
| EASTADDRESSQUEUEOUT1 | Local      | 0                | 1                 | 1                   |



### Note

You can change the log level if you would like to see the debug-level messages that the service generates. If you have access to the WebSphere MQ Explorer on the host system, look for your message in the queue count.

In the next step, you use the other MPGW service (EastAddressSearchMQclient) to retrieve the response from the `EASTADDRESSQUEUEOUT<nn>` sent by this MPGW service (EastAddressSearch).

- \_\_\_ 2. Use the `EastAddressSearchMQclient` MPGW to retrieve the response message that the `EastAddressSearch` service put.

- \_\_\_ a. In a terminal window, go to the `<lab_files>/MQ` directory.  
\_\_\_ b. Enter the command:

```
curl -H "Operation: GET" -H "Queue: EASTADDRESSQUEUEOUT<nn>" -H "Content-Type: text/xml" --data-binary @findByLocation.xml http://<dp_public_ip>:<mpgw_mq_client_port>
```



### Information

Although the `findByLocation.xml` input file is not needed for the GET, the service is specified a request type of SOAP. If the file was not included on the HTTP request, a SOAP schema validation error would occur.

- \_\_\_ c. Verify in your terminal window that you get the correct response from the `EastAddressSearch` web service (a series of `<Address>` elements for the `<state>` of NC).

## 15.5.Configuring transaction capability on the DataPower WebSphere MQ queue manager

In the previous section, you tested a scenario where you successfully GET and PUT messages from queues in WebSphere MQ. In this section, you examine the support that the DataPower WebSphere MQ queue manager object provides when messages cannot be processed.

- \_\_\_ 1. Examine the East Address WebSphere MQ FSH transaction support.
  - \_\_\_ a. Open the `EastAddressSearch` configuration page.
  - \_\_\_ b. Scroll down to front side settings and select `EastAddressMQFSH`.
  - \_\_\_ c. Click [...] (edit) to open and view the configuration settings.
  - \_\_\_ d. In the Configure WebSphere MQ Front Side Handler page, open the `EastAddressQM` queue manager object.
  - \_\_\_ e. Scroll down to the **Units of Work** field and verify that it has the value 0.



### Note

The **Units Of Work** field can have two values: 0 or 1. The value 0 indicates that the WebSphere MQ FSH object does not participate in a transaction, which means that it silently discards failed messages. The value 1 indicates that the WebSphere MQ queue manager object does participate in a transaction (local), and errors that occur during processing of the message within the service cause a rollback.



### Information

A **transaction** either commits all of the operations that are completed or rolls all of them back if an error occurs.

- \_\_\_ 2. Change the `findByLocation.xml` file to force a schema validation error in the `EastAddressSearch` service policy.
  - \_\_\_ a. Open the `MQ/findByLocation.xml` file in any text editor and change the namespace value  
`xmlns:q0="http://east.address.training.ibm.com"` to  
`xmlns:q0="http://address.training.ibm.com"`  
 This change causes an error in the service policy when doing schema validation.

- \_\_\_ 3. Use the `EastAddressSearchMQclient` to put a message onto `EASTADDRESSQUEUEIN<nn>`.
- \_\_\_ a. In a terminal window, navigate to the `<lab_files>/MQ` directory.
- \_\_\_ b. Enter the PUT command as before:
- ```
curl -H "Operation: PUT" -H "Queue: EASTADDRESSQUEUEIN<nn>"  
-H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_mq_client_port>
```

**Note**

This command puts the `findByLocation.xml` message onto the `EASTADDRESSQUEUEIN<nn>`. The `EastAddressSearch` service retrieves the message from this queue and invokes the East Address Search web service. Since the `EastAddressSearch` service policy generates an error, a SOAP fault is generated. However, since the **Unit of Work** attribute is set to **0**, the WebSphere MQ queue manager object discards the message. Leave `findByLocation.xml` with the error.

- \_\_\_ 4. Use the `EastAddressSearchMQclient` to get the message from the `EASTADDRESSQUEUEOUT<nn>`.
- \_\_\_ a. Enter the GET command as before:
- ```
curl -H "Operation: GET" -H "Queue:  
EASTADDRESSQUEUEOUT<nn>" -H "Content-Type: text/xml"  
--data-binary @findByLocation.xml  
http://<dp_public_ip>:<mpgw_mq_client_port>
```
- The command retrieves the SOAP fault message. The original message sent to the back-end is lost.
- \_\_\_ 5. Change the `EastAddressSearchMQFSH` object to support transactions.
- \_\_\_ a. Scroll down to Front side settings and click `EastAddressMQFSH`.
- \_\_\_ b. In the Configure WebSphere MQ Front Side Handler page, open the `EastAddressQM` queue manager object.
- \_\_\_ c. Scroll down to the **Units of Work** field and change the value to: **1**
- \_\_\_ d. On the **Connection** tab screen, change **Automatic Retry** to **on**.
- \_\_\_ e. On the **Main** tab screen, change **Automatic Backout** to **on**. This option enables two more fields.

- \_\_\_ f. For **Backout Threshold**, specify 3 and for **Backout Queue Name** specify: EASTADDRESSQUEUEERROR<nn>

#### Units of Work and Backout

Units of Work

1

Automatic Backout

on  off

Backout Threshold

3

Backout Queue Name

EASTADDRESSQUEUEERROR01

- \_\_\_ g. Click **Apply**, **Apply** again, and close all windows. Click **Apply** in the main service configuration page.



#### Information

In this step, you configured support for transactions. If any operations done by the EastAddressSearch service fail, the WebSphere MQ FSH tries three times (Backout Threshold). If all of the attempts fail, then the message is put on the EASTADDRESSQUEUEERROR<nn> queue (Backout Queue Name).

- \_\_\_ 6. Use the `EastAddressSearchMQclient` to put a message onto EASTADDRESSQUEUEIN<nn>.

- \_\_\_ a. Enter the PUT command again:

```
curl -H "Operation: PUT" -H "Queue: EASTADDRESSQUEUEIN<nn>"  
-H "Content-Type: text/xml" --data-binary  
@findByLocation.xml  
http://<dp_public_ip>:<mpgw_mq_client_port>
```



#### Note

This command puts the original `findByLocation.xml` message onto EASTADDRESSQUEUEIN<nn>. A few moments later, the EastAddressSearch gets this message and invokes its service policy, which causes a schema validation error. Since the **Unit of Work** attribute is set at 1, the WebSphere MQ queue manager object does not lose the message and tries again. After three failed attempts, it puts the **original** message on the backout queue EASTADDRESSQUEUEERROR<nn>.

- \_\_\_ 7. Use the `EastAddressSearchMQclient` to get the message from the error queue, `EASTADDRESSQUEUEERROR<nn>`.

- \_\_\_ a. Enter a slightly different GET command:

```
curl -H "Operation: GET" -H "Queue:  
EASTADDRESSQUEUEERROR<nn>" -H "Content-Type: text/xml"  
--data-binary @findByLocation.xml  
http://<dp_public_ip>:<mpgw_mq_client_port>
```

The original SOAP request, `findByLocation.xml`, is displayed.

- \_\_\_ 8. Change the `findByLocation.xml` namespace URL back to its correct value of `xmlns:q0="http://east.address.training.ibm.com"`.
- \_\_\_ 9. Verify that the `EastAddressSearch` still functions by using the `EastAddressSearchMQclient` from the terminal window to get and put messages.

## **End of exercise**

## Exercise review and wrap-up

In this exercise, you created a WebSphere MQ FSH that gets messages from a queue and puts responses on another queue. You also created a multi-protocol gateway service to send and receive messages from a queue in WebSphere MQ in a fire-and-forget messaging pattern. Finally, you configured transaction support between DataPower and WebSphere MQ for when errors occur during message processing.



# Appendix A. Creating a firewall and HTTP proxy for a web application

## What this exercise is about

In this exercise, you create a web application firewall to secure the back-end East Address Search web service application. Clients connect to the web application firewall that is hosted on the DataPower appliance, which uses a AAA policy to authenticate users. You also configure an SSL proxy profile to securely access the back-end web application firewall.



### Note

This course provides an appendix unit and an appendix exercise on the web application firewall (WAF). Since the WAF is not commonly used, this material is not covered in class. It is provided for your self-study.

## What you should be able to do

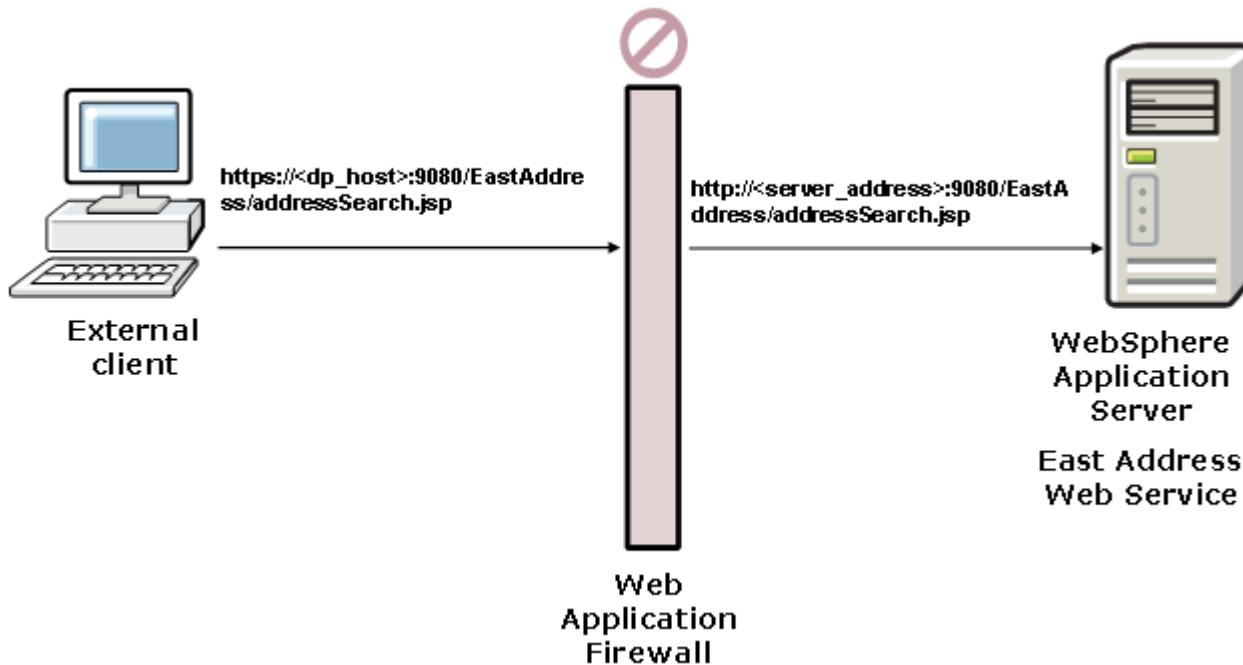
At the end of the exercise, you should be able to:

- Use the web application firewall wizard to create a web application firewall
- Implement a security policy on a web application firewall
- Create a reverse proxy to virtualize requests to web applications

## Introduction

In this exercise, you create a web application firewall on the DataPower appliance that secures access to the back-end web application.

The diagram shows the message exchange from the client to back-end web application.



The web application firewall can offload the tasks that are typically performed on an application server. In this exercise, you create a AAA policy that completes authentication by using HTTP basic authentication over SSL. The web application firewall can also protect against malicious attacks that are sent with URL-encoded HTTP parameters. A profile can be set up to check the validity of the parameters that are passed in with HTTP requests.

## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Completion of the previous exercises (see the Preface section in the exercise instructions for details)
- The web application front end to the **Address Search** web services that run on WebSphere Application Server

# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 10: 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:
  - *<dp\_public\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<dp\_internal\_ip>*: XMLthreat instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<dp\_public\_ip>*: instructor-assigned address for the IBM WebSphere DataPower SOA Appliance
  - *<backend\_server\_ip>*: instructor-assigned address for the enterprise application server that hosts the Address Search web services
  - *<waf\_http\_port>*: port number for the web application firewall
  - *<waf\_https\_port>*: port number for the web application firewall that uses SSL

## A.1. Examine existing East Address web application

In this section, you examine the web-based interface of the East Address web service.

- \_\_\_ 1. Examine the East Address web application.



### Note

A web interface for the West Address web application is not currently supported.

- \_\_\_ a. Open a web browser and enter the URL:

`http://<backend_server_ip>:9080/EastAddress/addressSearch.jsp`

A web page opens, allowing you to call the East Address search web service. This interface calls the back-end service over HTTP. It is not making SOAP over HTTP calls.

### Address Information Search

Search for address information by entering the following information

|                                       |   |
|---------------------------------------|---|
| Search type:                          | <input type="button" value="findByName"/> |
| Title                                 | <input type="text"/>                      |
| First Name                            | <input type="text"/>                      |
| Last Name                             | <input type="text"/>                      |
| <input type="button" value="Submit"/> |   |

- \_\_\_ b. Recall that the East Address web application supports three operations:

- `findByName`
- `findByLocation`
- `retrieveAll`

Use the **Search Type** field to select the operation to complete. For each operation, the parameters are shown dynamically based on the operation that is completed.

- \_\_\_ c. Select the **findByLocation** operation. The **city** and **state** fields are shown. Enter NC for the **State** and click **Submit**.

\_\_\_ d. Verify that five addresses are returned.

| Number | Title | First Name | Last Name | Details                                   |
|--------|-------|------------|-----------|---|
| 1      | Ms    | Lisa       | Williams  | 146 Miller Street, Charlotte, NC, 28205   |
| 2      | Mr    | Wayne      | Green     | 4715 Cherry Drive, Charlotte, NC, 28212   |
| 3      | Mr    | Phillip    | Hughes    | 4111 South Avenue, Charlotte, NC, 28213   |
| 4      | Ms    | Phyllis    | Walker    | 2280 Dogwood Avenue, Charlotte, NC, 28206 |
| 5      | Mrs   | Tina       | Cox       | 2649 First Lane, Charlotte, NC, 28206     |

\_\_\_ e. Close the web browser when you are done.

## A.2. Create a web application firewall

In this section, you create a web application firewall to securely access the web-based version of the Address web services applications.

- \_\_\_ 1. Create a web application firewall and configure the front-end and back-end ports.
  - \_\_\_ a. In the DataPower **Control Panel**, click **Web Application Firewall**.



- \_\_\_ b. In the Configure Web Application Firewall page, click **Add Wizard**. The wizard approach asks a series of questions to generate a web application firewall.
- \_\_\_ c. Enter the name for the web application firewall: **AddressWAF**
- \_\_\_ d. For **Remote Host Address**, enter `<backend_server_ip>` and use `<backend_server_ip>` for the **Remote IP Port**.

 Create a Web Application Firewall service

**Back End [Server] Information**

The information to connect to the remote web server.

|                                |  |
|--------------------------------|--|
| <b>Remote Host</b>             | <input type="text" value="backend_server_ip"/> * |
| <b>Remote Port</b>             | <input type="text" value="was_server_port"/> *   |
| <b>Do you want to use SSL?</b> | <input type="checkbox"/> *                       |

**Back** **Cancel** **Next**

The remote host address and IP port are the destination to which requests are forwarded after processing by the web application firewall.

- \_\_\_ e. Click **Next**.

- \_\_\_ f. For the **Front End (Client-Facing)** information, enter <dp\_public\_ip> for the IP address, and enter the port number for <waf\_http\_port>.



## Create a Web Application Firewall service

### Front End (Client-Facing) Information

The information for clients to connect to the appliance.

#### Source Addresses

| IP           | Port                 |
|--------------|----------------------|
| dp_public_ip | Select Alias<br>7954 |

- \_\_\_ g. Click **Add**.

The front-end information is the URI that clients enter in their web browser to call the web application firewall. The web application firewall can listen for requests on multiple ports.

#### Source Addresses

| IP           | Port         |
|--------------|--------------|
| dp_public_ip | 7954         |
| 0.0.0.0      | Select Alias |

- \_\_\_ h. Click **Next**.

- \_\_\_ 2. View existing AAA policies.

- \_\_\_ a. In the **Do you want to enable AAA** field, select **on** by selecting the check box. This action displays a list where you can select existing AAA policies. The AAA policies that are created from previous exercises exist here. Since an object-oriented approach is used for the configuration of the appliance, you do not create a AAA policy in this step. However, you create one later in the exercise. Turn off AAA by selecting **off** by clearing the check box, and click **Next**.

- \_\_\_ 3. A **Shared Secret Key** is not used; click **Next**.

- \_\_\_ 4. Create a name-value profile to validate name-value pairs.

This object is used to validate name-value pairs that are passed from HTML forms. This page allows you to create three types of name-value pair objects:

- Request HTTP header
- Request URLs

- Response HTTP header

The `addressSearch.jsp` page submits the request by using an HTML form element and the method POST. You create a URL name-value profile object.

- Turn on the **Enable URL Name Value profile**.
- When the screen refreshes, click the plus sign (+) beside the **Enable URL Name Value profile** field to create a name-value profile object.
- Enter `AddressNV` as the name of the object.
- Switch to the **Validation List** tab. Use this page to enter the name-value regular expression to validate incoming name-value pairs.
- Click **Add**.

If you look inside the HTML form, you notice that it contains a Form element with an **operation** control that contains the list of operations (`findByName`, `findByLocation`, and `retrieveAll`) with values of 0, 1, or 2.

Depending on the operation, more controls are shown on the page. Each control is named starting with **input** and ending with a number. The values for each of these input controls can be any alphanumeric character. You build a name-value profile object that validates these values.

Configure Name-Value Profile

Main Validation List

Name-Value Profile

Apply Cancel Help

| Name    | Value Constraint | Failure Policy | Map Value | Check XSS | XSS (Cross Site Scripting) Protection Patterns File |
|---------|------------------|----------------|-----------|-----------|---|
| (empty) |                  |                |           |           | <input type="button" value="Add"/>                  |

- Enter the name expression: `operation`

- \_\_\_ g. Click **Value Helper** to open the “Value Helper” page. In the **Numeric** category, select **Digits Only**. This action generates an expression. Change the expression value from `^[0-9]+$` to: `^[0-2]+$`  
The latter expression allows the values 0, 1, or 2. This expression is generated by using Perl-compatible regular expressions.
- \_\_\_ h. Click **Use Expression**.

**Validation List**

| Name Expression  | Value Constraint  | Failure Policy | Map Value                                   | Check XSS |
|--|---|----------------|---|-----------|
| (empty)  |   |                |   |           |
| Name Expression  | operation   | *              |   |           |
| Value Constraint   | <code>^[0-2]+\$</code>  |                | <input type="button" value="Value Helper"/> | *         |
| Failure Policy   | Error   | *              |   |           |
| Check XSS  | <input type="radio"/> on <input checked="" type="radio"/> off |                |   |           |
| <input type="button" value="Apply"/> <input type="button" value="Cancel"/> |   |                |   |           |

- \_\_\_ i. Click **Apply**.
- \_\_\_ j. Click **Add** again to create a name-value expression with the following information:
- **Name Expression:** `^input[0-9]$`
  - **Value Constraint:** `^[a-zA-Z]+$`

**Validation List**

| Name Expression  | Value Constraint  | Failure Policy | Map Value                                   | Check XSS   |
|--|---|----------------|---|---|
| operation  | <code>^[0-2]+\$</code>  | Error          |   | off  |
| Name Expression  | <code>^input[0-9]\$</code>                                    | *              |   |   |
| Value Constraint   | <code>^[a-zA-Z]+\$</code>                                     |                | <input type="button" value="Value Helper"/> | *   |
| Failure Policy   | Error   | *              |   |   |
| Check XSS  | <input type="radio"/> on <input checked="" type="radio"/> off |                |   |   |
| <input type="button" value="Apply"/> <input type="button" value="Cancel"/> |   |                |   |   |

- \_\_\_ k. Click **Apply**.

- \_\_ I. Verify that you have the following name-value expressions:

| Validation List |                  |                |           |           |   |
|-----------------|------------------|----------------|-----------|-----------|---|
| Name Expression | Value Constraint | Failure Policy | Map Value | Check XSS | XSS (Cross Site Scripting) Protection Patterns File |
| operation       | ^[0-2]+\$        | Error          |           | off       | store:///XSS-Patterns.xml                           |
| ^input[0-9]\$   | ^[a-zA-Z]+\$     | Error          |           | off       | store:///XSS-Patterns.xml                           |
|                 |                  |                |           |           | <input type="button" value="Add"/>                  |

The failure policy for both entries is set to error, which specifies that if validation fails, the request is not forwarded to the back-end web application. If an error policy is defined, it can specify the conditions when errors occur in the request processing:

<RemoteEndpointHostname>x.x.x.x</RemoteEndpointHostname>

- \_\_ 5. Click **Apply** to save the Name Value Profile object.
- \_\_ 6. Click **Next**.
- \_\_ 7. Query string handling:

The query string is the section of the URL after the question mark (?); for example, the query string part of the URL

`http://dphost:port/services/Address?operation=1`

is `operation=1`. The **Allow Query String** field allows the web application firewall to allow or disallow query strings. If query strings are allowed, the web application firewall can specify a name-value profile to inspect the query strings that are passed with the request.

- \_\_ a. Leave the default values on the query string page. The web application firewall allows query strings, but it does not validate the values that are passed.

**Request Information - Query strings**

---

The query string profile validates query strings in requests.

---

**Allow Query String**

---

**QueryString Name-Value Profile**

---

- \_\_ b. Click **Next**.
- \_\_ 8. Cookie handling:

The web application firewall can accept or deny cookies. If allowed, the cookies can be signed or encrypted by the DataPower appliance. In addition, a name-value profile can be used to validate cookie values.

- \_\_\_ a. Leave the default values on the cookies page. The web application firewall allows cookies, but it does not validate the values that are passed.

**Request Information - Cookies**

The cookie profile provides management for cookies in requests.

**Allow Cookies**

**Sign/Encrypt Cookies**

**Sign or Encrypt All Cookies**

**Cookie Content Name-Value Profile**

- \_\_\_ 9. Click **Next**.
- \_\_\_ 10. The **Response Header** does not require verification now; click **Next**.

- \_\_\_ 11. Verify that the web application firewall is created correctly.  
\_\_\_ a. Verify that your web application firewall service contains the following values:

 Create a Web Application Firewall service

**Confirm Your Changes and Commit**

| Web Application Firewall Name:     | AddressWAF   |       |      |     |              |     |       |
|------------------------------------|--|-------|------|-----|--------------|-----|-------|
| Summary:                           |  |       |      |     |              |     |       |
| Remote Host:                       | WSserver   |       |      |     |              |     |       |
| Remote Port:                       | 9080   |       |      |     |              |     |       |
| Source Addresses:                  | <table><thead><tr><th>IP</th><th>Port</th><th>SSL</th></tr></thead><tbody><tr><td>dp_public_ip</td><td>704</td><td>(off)</td></tr></tbody></table> | IP    | Port | SSL | dp_public_ip | 704 | (off) |
| IP                                 | Port   | SSL   |      |     |              |     |       |
| dp_public_ip                       | 704  | (off) |      |     |              |     |       |
| AAA Policy:                        |  |       |      |     |              |     |       |
| URL Name-Value Profile:            | AddresNV   |       |      |     |              |     |       |
| Allow Query String:                | allow  |       |      |     |              |     |       |
| QueryString Name-Value Profile:    |  |       |      |     |              |     |       |
| Allow Cookies:                     | allow  |       |      |     |              |     |       |
| Sign/Encrypt Cookies:              | none   |       |      |     |              |     |       |
| Cookie Content Name-Value Profile: |  |       |      |     |              |     |       |

**Back**   **Cancel**   **Commit**

- \_\_\_ b. Click **Commit**.  
\_\_\_ c. Click **View Web Application Firewall** to view the configuration.

## A.3. Examine the objects that the web application firewall generated

In this section, you examine the objects that the web application firewall wizard generated.

- \_\_\_ 1. Examine the web application firewall objects.
  - \_\_\_ a. Verify that the web application firewall generated a security policy called AddressWAF\_security\_policy. There is also a reference to an existing XML manager called **default**. The front side and back side settings are populated with the values that are entered from the wizard. These values can be modified if the values are entered incorrectly.

Web Application Firewall status: [up]

### General Configuration

Configuration settings the service needs to process transactions.

| <b>Name</b>  | <b>XML Manager</b>   |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
|--|--|------------------------|--------|-----|--------|--------------|------------------|-------|--------|--------------------------------------|---|------------------------|-----|
| <input type="text" value="AddressWAF"/> *  | <input type="text" value="default"/> <input type="button" value="+"/> <input type="button" value="..."/> *   |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| <b>Summary</b>   | <b>SSL Proxy Profile</b>   |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| <input type="text"/>   | <input type="text" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>  |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| <b>Back side settings</b>  | <b>Front side settings</b>   |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| The host and port for the server. For the host, specify a host name or host alias. | The IP address and port on the appliance on which the service listens and whether this address accepts HTTP or HTTPS requests. Click Add after defining these settings.  |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| <b>Remote Host</b>   | <b>Source Addresses</b>  |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| <input type="text" value="WSserver"/> *  | <table border="1"> <thead> <tr> <th>IP</th> <th>Port</th> <th>SSL</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>dp_public_ip</td> <td>7<del>4</del>4</td> <td>(off)</td> <td>Remove </td> </tr> <tr> <td><input type="text" value="0.0.0.0"/></td> <td><input type="button" value="Select Alias"/></td> <td><input type="button"/></td> <td>Add </td> </tr> </tbody> </table> | IP                     | Port   | SSL | Action | dp_public_ip | 7 <del>4</del> 4 | (off) | Remove | <input type="text" value="0.0.0.0"/> | <input type="button" value="Select Alias"/> | <input type="button"/> | Add |
| IP   | Port   | SSL                    | Action |     |        |              |                  |       |        |                                      |   |                        |     |
| dp_public_ip   | 7 <del>4</del> 4   | (off)                  | Remove |     |        |              |                  |       |        |                                      |   |                        |     |
| <input type="text" value="0.0.0.0"/>   | <input type="button" value="Select Alias"/>  | <input type="button"/> | Add    |     |        |              |                  |       |        |                                      |   |                        |     |
| <b>Remote Port</b>   |  |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |
| <input type="text" value="9080"/> *  |  |                        |        |     |        |              |                  |       |        |                                      |   |                        |     |

- \_\_\_ 2. Verify the security policy.

- \_\_\_ a. Open the application security policy object to view the generated object by clicking **Edit** ([...]) next to it.
- \_\_\_ b. In the web page that opens, verify that you see four tabs: **General**, **Request Maps**, **Response Maps**, and **Error map**.

The request maps contain a list of security profiles that run or a request message when matched by a matching rule. The response map is similar, except that it applies to the response message.

- \_\_\_ c. Click the **Request Maps** tab and open the AddressWAF\_request request profile by clicking the **Edit** (the [...]).
- \_\_\_ d. The request map profile contains the default configuration, with the addition of the name-value profile that the wizard creates. Click the **Name Value** tab to view the generated object.



- \_\_\_ e. In the **URL-Encoded Body Name-Value Profile** field, open **AddressNV**.
- \_\_\_ f. In the web page that opens, click the **Validation List** tab to view the name-value expressions created.
- \_\_\_ g. Close all windows when you are done.

The response map object that is generated contains the default security policy configuration.

- \_\_\_ 3. Test the web application firewall AAA policy.
  - \_\_\_ a. Close any web browser windows to terminate previously opened connections.
  - \_\_\_ b. Open a web browser and enter the web application firewall URL:  
`http://<dp_public_ip>:<waf_http_port>/EastAddress/addressSearch.jsp`
  - \_\_\_ c. Select the **findByLocation** operation. The **city** and **state** text boxes are shown dynamically. Enter **NC** for the **State** and click **Submit**.

\_\_\_ d. Verify that the following addresses are returned:

| Number | Title | First Name | Last Name | Details                                   |
|--------|-------|------------|-----------|---|
| 1      | Ms    | Lisa       | Williams  | 146 Miller Street, Charlotte, NC, 28205   |
| 2      | Mr    | Wayne      | Green     | 4715 Cherry Drive, Charlotte, NC, 28212   |
| 3      | Mr    | Phillip    | Hughes    | 4111 South Avenue, Charlotte, NC, 28213   |
| 4      | Ms    | Phyllis    | Walker    | 2280 Dogwood Avenue, Charlotte, NC, 28206 |
| 5      | Mrs   | Tina       | Cox       | 2649 First Lane, Charlotte, NC, 28206     |

\_\_\_ e. Close the web browser when you are done.

## A.4. Configure the web application security policy

- \_\_\_ 1. Create a session policy to restrict the set of allowable web pages.
- \_\_\_ a. Open the **AddressWAF\_security\_policy** web application security policy.
- \_\_\_ b. Click the **Request Maps** tab.
- \_\_\_ c. Open the **AddressWAF\_request** request profile.
- \_\_\_ d. Click the **Profile** tab.
- \_\_\_ e. Scroll down and create a **Session Policy**. The **Session Management policy** web page allows you to specify a matching rule that contains valid URIs.
- \_\_\_ f. Enter `AddressWAF_SMP` in the name field for the name of the Session Management Policy.
- \_\_\_ g. Scroll down the page, and click the ellipse (...) next to the Start Page to create a Start Page.

Main

Session Management Policy

**Name** AddressWAF\_SMP \*

Administrative State  enabled  disabled

Comments

Auto Renew  on  off

Session Lifetime 3600 seconds

Address Independent Sessions  on  off

**Start Pages** (none) + ...

- \_\_\_ h. Enter `AddressWAF_SP` in the name field.

The **Start Pages** field allows you to select a matching rule for specifying a valid set of URIs.

- \_\_\_ i. Create a matching rule that is called `AddressWAF` that matches the URI `/EastAddress/*` by clicking `+` (new).

Main Matching Rule

Matching Rule

Apply Cancel Help

Name AddressWAF\_SP \*

**Matching Rule**

| Matching Type | HTTP Header Tag | HTTP Value Match | URL Match      | Error Code | XPath Expression | HTTP Method |     |
|---------------|-----------------|------------------|----------------|------------|------------------|-------------|-----|
| URL           |                 |                  | /EastAddress/* |            |                  | default     | Add |

- \_\_\_ j. Click **Apply** to save the **Matching Rule**.

- \_\_\_ k. Click **Apply** to save the **Start Page**.

#### Session Management Policy

Apply Cancel

Name AddressWAF\_SMP \*

Administrative State  enabled  disabled

Comments

Auto Renew  on  off

Session Lifetime 3600 seconds

Address Independent Sessions  on  off

Start Pages AddressWAF\_SP \*

- \_\_\_ l. Click **Apply** to save the **AddressWAF\_SMP** Session Management Policy.
  - \_\_\_ m. Click **Apply** to save the **AddressWAF\_request** Web Request Profile.
  - \_\_\_ n. Click **Commit** to save the **Configure an Application Security Policy**.
  - \_\_\_ o. Click **Apply** to save the **Web Application Firewall**.
- \_\_\_ 2. Test the web application firewall session management policy.
- \_\_\_ a. Close any web browser windows to terminate previously opened connections.
  - \_\_\_ b. Open a web browser and enter an invalid URL such as  
`http://<dp_public_ip>:<waf_http_port>/SomeAddress/  
addressSearch.jsp`
  - \_\_\_ c. Verify that you get an error in the web browser with the following text:

You cannot start here. Perhaps your login credential is expired.

You might want to clear the cookies in your browser.
- \_\_\_ d. Now enter the valid URL:  
`http://<dp_public_ip>:<waf_http_port>/EastAddress/  
addressSearch.jsp`
- The Address search web page is shown. The session management policy that you created enforces a set of valid URLs through the match rule.
- \_\_\_ 3. Create a AAA policy to validate user credentials by using HTTP basic authentication.



### Information

The AAA policy is an object that is used to authenticate, authorize, and audit user credentials. The web application security firewall security policy might refuse AAA policies that are created in previous services.

In this part, you create a AAA policy object by using the credentials in an HTTP basic authentication header and authenticate it by using the AAA.

- \_\_\_ a. Open the web application firewall security policy, **AddressWAF\_security\_policy**.
- \_\_\_ b. Click the **Request Maps** tab.
- \_\_\_ c. Open the **AddressWAF\_request** request profile.
- \_\_\_ d. Click the **Profile** tab.
- \_\_\_ e. Scroll down to create a **AAA Policy**.
- \_\_\_ f. Enter the name: `AddressWAF_AAA`

- \_\_\_ g. Click the **Extract Identity** tab and select the **HTTP Authentication Header** box. Leave the **HTTP Basic Authentication Realm** value at **login**.

The screenshot shows the 'AAA Policy' configuration screen. At the top, there are 'Apply' and 'Cancel' buttons. Below them, the 'Name' field is set to 'AddressWAF\_AAA'. Under the 'Methods' section, the 'HTTP Authentication Header' checkbox is checked. A list of other methods is provided, each with an unchecked checkbox. At the bottom, the 'HTTP Basic Authentication Realm' field contains the value 'login'.

| Name                            | AddressWAF_AAA *  |
|---------------------------------|---|
| Methods                         | <input checked="" type="checkbox"/> HTTP Authentication Header<br><input type="checkbox"/> Password-carrying UsernameToken Element from WS-Security Header<br><input type="checkbox"/> Derived-key UsernameToken Element from WS-Security Header<br><input type="checkbox"/> BinarySecurityToken Element from WS-Security Header<br><input type="checkbox"/> WS-SecureConversation Identifier<br><input type="checkbox"/> WS-Trust Base or Supporting Token<br><input type="checkbox"/> Kerberos AP-REQ from WS-Security Header<br><input type="checkbox"/> Kerberos AP-REQ from SPNEGO Token<br><input type="checkbox"/> Subject DN of the SSL Certificate from the Connection Peer<br><input type="checkbox"/> Name from SAML Attribute Assertion<br><input type="checkbox"/> Name from SAML Authentication Assertion<br><input type="checkbox"/> SAML Artifact<br><input type="checkbox"/> Client IP Address<br><input type="checkbox"/> Subject DN from Certificate in the Message's signature<br><input type="checkbox"/> Token Extracted from the Message<br><input type="checkbox"/> Token Extracted as Cookie Value<br><input type="checkbox"/> LTPA Token<br><input type="checkbox"/> Processing Metadata<br><input type="checkbox"/> Custom Template<br><input type="checkbox"/> HTML Forms-based Authentication<br><input type="checkbox"/> OAuth<br>* |
| HTTP Basic Authentication Realm | login   |

- \_\_\_ h. Click the **Authenticate** tab. For the method, select **Use DataPower AAA Info File**.  
 \_\_\_ i. In the **AAA Info File URL**, click **+** (new).  
 \_\_\_ j. Enter `AddressUser` as the **Default Credential Name**.  
 \_\_\_ k. Click **Next**.  
 \_\_\_ l. Click **Add** to enter Credentials for Authenticated Users.

\_\_\_ m. In the Edit AAA Info XML File page, enter the following information;

- **Username:** AddressAdmin
- **Password:** web1sphere
- **Credential Name:** AddressUser

| Edit AAA Info XML File  |  |
|---|--|
| Add a new Credential (Authentication List Entry)                            |  |
| <b>Host Name or IP Address</b>  | <input type="text"/>                                       |
| <b>IP network</b>   | <input type="text"/>                                       |
| <b>Username</b>   | <input type="text" value="AddressAdmin"/>                  |
| <b>Password</b>   | <input type="password"/> *****<br><input type="password"/> |
| <b>Distinguished Name</b>   | <input type="text"/>                                       |
| <b>Custom Token</b>   | <input type="text"/>                                       |
| <b>Credential Name</b>  | <input type="text" value="AddressUser"/> *                 |
| <input type="button" value="Submit"/> <input type="button" value="Cancel"/> |  |

- \_\_\_ n. Click **Submit**.
- \_\_\_ o. Click **Next**.
- \_\_\_ p. For **Map Credentials**, do not add any entries. Click **Next**.
- \_\_\_ q. Click **Next** for **Map Resources**.
- \_\_\_ r. Click **Next** for **Authorized Access to Resources**.
- \_\_\_ s. Enter the file name `AddressInfo.xml` and make sure that the `local:` directory is selected. Optionally, enter a description in the summary field.

| Edit AAA Info XML File  |   |
|---|---|
| <b>Name of new file</b>   | <input type="text" value="local:///"/> <input type="button"/>                   |
|   | <input type="text" value="AddressInfo.xml"/> *                                  |
| <b>file summary</b>   | <input type="text" value="AAA file to validate credentials for Address Users"/> |
| <input type="button" value="Cancel"/> <input type="button" value="Back"/> <input type="button" value="Next"/> |   |

- \_\_\_ t. Click **Next**, click **Commit**, and then click **Done**.

You are back at the “Configure AAA policy” web page.

- \_\_\_ u. Ensure that `local:///AddressInfo.xml` is selected as the AAA information file URL. Set it if it is not selected.
- \_\_\_ v. Click the **Extract Resource** tab to select how to extract a resource from the message. In this simple example, you use the client URL. Check the **URL Sent by Client** box.

### AAA Policy

**Apply**   **Cancel**

|                             |  |
|-----------------------------|--|
| <b>Resource Information</b> | <input type="checkbox"/> URL Sent to Back End<br><input checked="" type="checkbox"/> URL Sent by Client<br><input type="checkbox"/> URI of Toplevel Element in the Message<br><input type="checkbox"/> Local Name of Request Element<br><input type="checkbox"/> HTTP Operation (GET/POST)<br><input type="checkbox"/> XPath Expression<br><input type="checkbox"/> Processing Metadata<br>* |
|-----------------------------|--|

- \_\_\_ w. Click **Apply**.
- \_\_\_ x. Verify that the AAA policy is created and selected in the profile web page. Click **Apply** again.
- \_\_\_ y. Click **Commit**.
- \_\_\_ z. Click **Done**.
- \_\_\_ aa. Click **Apply** to save the **Web Application Firewall**.
- \_\_\_ 4. Modify the generated `AddressInfo.xml` file to remove access to unauthenticated identities.
- \_\_\_ a. From the DataPower **Control Panel**, click **File Management**.

- \_\_\_ b. Expand local: and click **Edit** beside AddressInfo.xml. A new web page opens with the contents of the file.

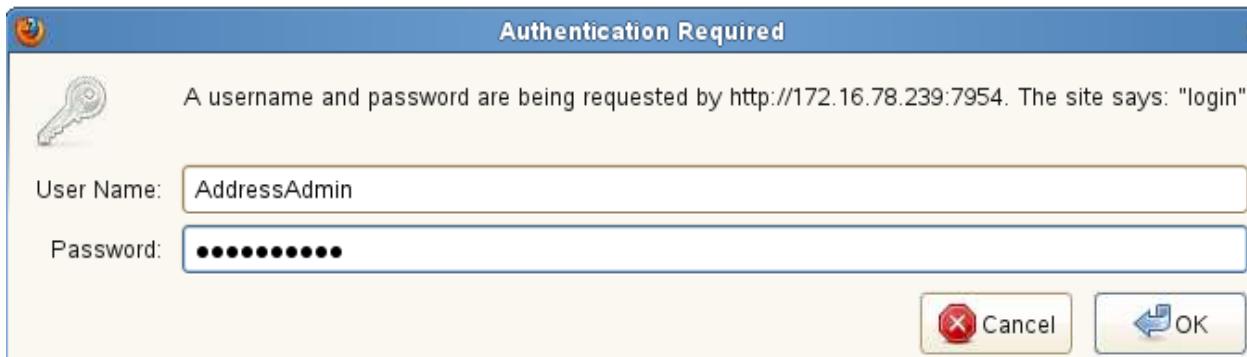
File name: local:/AddressInfo.xml Mode: Edit

```
<aaa:AAAInfo xmlns:dpfunc="http://www.datapower.com/extensions/functions" xmlns:aaa="http://www.datapower.com/AAAInfo">
<aaa:FormatVersion>1</aaa:FormatVersion><aaa:Filename>local:
///AddressInfo.xml</aaa:Filename><aaa:Summary>AAA file to validate
credentials for Address users</aaa:Summary><!-- Determine
credential from output of the extract-identity phase.
--><aaa:Authenticate><aaa:Username>AddressAdmin</aaa:Username>
<aaa:Password>websphere</aaa:Password>
<aaa:OutputCredential>AddressUser</aaa:OutputCredential>
</aaa:Authenticate><!-- Specify credential (if any) to use when
there is no authenticated identity. --><aaa:Authenticate>
<aaa:Any/><aaa:OutputCredential>AddressUser</aaa:OutputCredential>
</aaa:Authenticate><!-- Map credentials to different credentials.
--><!-- Determine resource from output of the extract-resource
phase. --><!-- Authorize access to resource for credentials.
--></aaa:AAAInfo>
```

Submit Cancel

- \_\_\_ c. Click **Edit**.
- \_\_\_ d. Find the following text and delete it (all of the text is on a single line):
- ```
<aaa:Authenticate>
<aaa:Any/>
<aaa:OutputCredential>AddressUser</aaa:OutputCredential>
</aaa:Authenticate>
```
- \_\_\_ e. Click **Submit** and close.
- \_\_\_ 5. Test the web application firewall AAA policy.
- \_\_\_ a. Close any open web browser windows to terminate previously opened connections.
- \_\_\_ b. Open a web browser window and enter the web application firewall URL:  
http://<dp\_public\_ip>:<waf\_http\_port>/EastAddress/  
addressSearch.jsp
- The web browser opens a window that asks for you for a user name and password.

- \_\_\_ c. Enter the user ID (AddressAdmin) and password (websphere) for the AAA policy that you created (these attributes are case-sensitive).



The Address search results page is shown.

- \_\_\_ d. Close the web browser and enter the web application firewall URL `http://<dp_public_ip>:<waf_http_port>/EastAddress/addressSearch.jsp` again.
- \_\_\_ e. This time, enter an invalid password. After three attempts, the web application firewall fails and displays a web page with the following text:

The Web Application Firewall has denied your transaction due to a violation of policy.

You might want to clear the cookies in your browser.

- \_\_\_ 6. Secure communications to the web application firewall over SSL.

 **Hint**

The SSL proxy profile from the previous exercise can be reused for the web application firewall. If you did not complete that exercise, run the following steps to create an SSL proxy profile.

- \_\_\_ a. Switch back to the DataPower **Control Panel**.
- \_\_\_ b. Open the AddressWAF web application firewall.
- \_\_\_ c. Create an SSL proxy profile.
- Click + (new) in the **SSL Proxy Profile** field.
  - Enter `AddressWAF_PP` as the name.
  - Create a reverse (server) crypto profile.
  - Enter the name: `AddressWAF_CP`

**Note**

In this SSL proxy profile, you create an identification credential object because the web application firewall is the SSL server. It uses a private and public key pair in SSL communication. The validation credential object is used as the presented certificate. If the web application firewall completed mutual authentication, then it would need a validation credential object as well.

- Create an **identification credential** object that is called `AddressWAF_IC` that references the Alice certificate and a private key object that references the Alice private key.

Main

Crypto Identification Credentials

**Apply** **Cancel**

<b>Name</b>	<input type="text" value="AddressWAF_IC"/> *
Administrative State	
<input checked="" type="radio"/> enabled <input type="radio"/> disabled	
<b>Crypto Key</b>	<input type="text" value="AliceKey"/> <input type="button" value="..."/> <input type="button" value="+"/> * <input type="button" value="Add"/>
<b>Certificate</b>	<input type="text" value="AliceCert"/> <input type="button" value="..."/> <input type="button" value="+"/> * <input type="button" value="Add"/>
<b>Intermediate CA Certificate</b>	<input type="text" value="(empty)"/> <input type="button" value="..."/> <input type="button" value="Add"/> + ...

- Click **Apply**.

- Back in the crypto profile, clear the **Disable SSL version 2** check box and verify that the crypto profile contains the following information:

Crypto Profile

---

<b>Name</b>	<input type="text" value="AddressWAF_CP"/> *
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Identification Credentials	<input type="text" value="AddressWAF_IC"/> <input type="button" value="+"/> <input type="button" value="..."/>
Validation Credentials	<input type="text" value="(none)"/> <input type="button" value="+"/> <input type="button" value="..."/>
Ciphers	<input type="text" value="HIGH: MEDIUM::aNULL::eNULL:@STRENGTH"/>
Options	<input checked="" type="checkbox"/> Enable default settings <input type="checkbox"/> <b>Disable SSL version 2</b> <span style="border: 2px solid red; padding: 2px;"> </span> <input type="checkbox"/> Disable SSL version 3 <input type="checkbox"/> Disable TLS version 1 <input type="checkbox"/> Permit insecure SSL renegotiation to a legacy SSL client

- Click **Apply**, click **Apply** to close the SSL proxy profile window, and click **Apply** in the web application firewall configuration page.

7. Create an HTTP listener for the web application firewall.
  - a. Under Front side settings, add a listener with the port <waf\_https\_port> and make sure that you select **SSL**.

#### Front side settings

The IP address and port on the appliance on which the service listens and whether this address accepts HTTP or HTTPS requests. Click Add after defining these settings.

#### Source Addresses

IP	Port	SSL	Action
dp_public_ip	7954	(off)	Remove 
<input type="text" value="dp_public_ip"/>	<input type="button" value="Select Alias"/>	<input type="text" value="7955"/>	<input checked="" type="checkbox"/> Add 

- \_\_\_ b. Remove the existing HTTP listener from the web application firewall so clients cannot communicate through HTTP.
  - \_\_\_ c. Click **Apply**.
- \_\_\_ 8. Use the Eclipse web browser to test the web application firewall SSL proxy profile.
- \_\_\_ a. Close any web browser windows to terminate previously opened connections.
  - \_\_\_ b. Open Eclipse.
  - \_\_\_ c. Select the Web perspective. Select **Window > Open Perspective > Web** from the menu bar and click **OK**.
  - \_\_\_ d. Open a web browser by clicking the globe icon at the top of the workspace.
  - \_\_\_ e. Enter the web application firewall URL:  
`https://<dp_public_ip>:<waf_https_port>/EastAddress/addressSearch.jsp`
  - \_\_\_ f. Click **View Certificate** and explore the certificate details.
  - \_\_\_ g. Click **OK** to close the certificate details; then click **Yes** to proceed to the web page.
  - \_\_\_ h. Enter the user name and password as before. After login, you are able to view the Address Search page.



#### Important

Clients cannot use `http` to connect to the AddressWAF web application firewall. They must use `https`. This mistake is common. If the logic fails, ensure that you are using `https`, **not** `http`.



#### Note

You can also use the browser to test SSL. You are presented with a certificate exception and accept the exception to process the SSL request.

`https://<dp_public_ip>:<waf_https_port>/EastAddress/addressSearch.jsp`

- \_\_\_ 9. Close and open the web browser, and enter the old web application firewall URL, `http://<dp_public_ip>:<waf_http_port>/EastAddress/addressSearch.jsp` again. You get an error that states the page cannot be displayed since the HTTP listener was removed.
- \_\_\_ 10. Close Eclipse.

## **End of exercise**

## Exercise review and wrap-up

In this exercise, you created a web application firewall to secure the back-end East Address Search web application. Clients connect to the back-end web application by connecting through the web application firewall with HTTP basic authentication over SSL. The web application firewall forwards to the East Address web service.

# Appendix B. Configuring WebSphere JMS

## What this exercise is about

This exercise shows you how DataPower can send and receive messages to and from the WebSphere Application Server default messaging engine. You create a multi-protocol gateway service that receives a request from cURL and sends a message to a WebSphere Application Server messaging engine to invoke the East Address Search web service over JMS.



### Note

This exercise reinforces the lecture in Unit 19, but since some students might not be interested in JMS, it is not included in the agenda.

## What you should be able to do

At the end of the exercise, you should be able to:

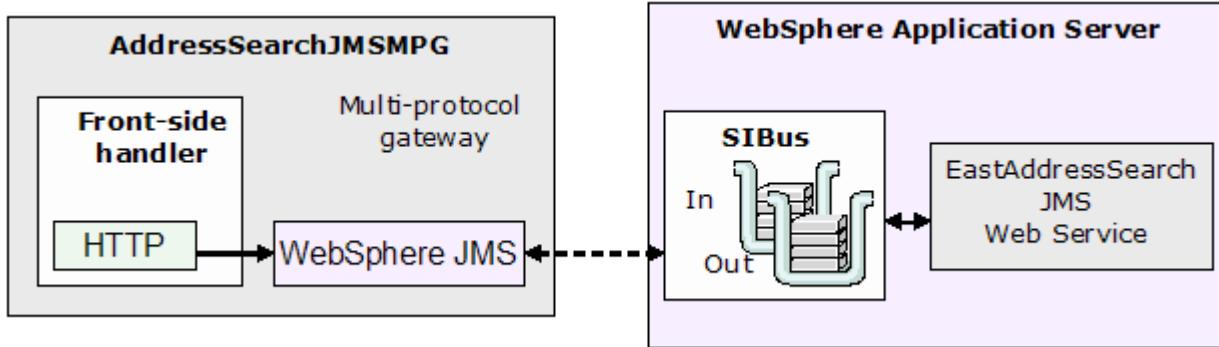
- Identify the fields in the service integration bus configuration on WebSphere Application Server V7.0 that are needed to configure the WebSphere DataPower JMS object
- Create a multi-protocol gateway service that invokes the East Address Search web service over the JMS transport

## Introduction

Java Messaging Service (JMS) is a Java API for accessing messaging middleware. It allows any Java client to access a messaging engine by using a standard set of interfaces. In a way that is similar to a WebSphere MQ client, JMS clients can access messaging middleware in a synchronous or asynchronous style. WebSphere Application Server V7.0 is a JMS provider that enables JMS clients to access JMS queues. DataPower does not use the JMS API to interact with the JMS provider since no JVM exists on the appliance. DataPower uses the IBM JFAP (JetStream Formats and Protocols) to communicate with the service integration bus and the JMS queues.

In this exercise, you create a multi-protocol gateway service with an HTTP front side handler. The service policy has a match all action and a **Result** action. After the service policy finishes execution, the

message is sent to a WebSphere Application Server destination on the service integration bus called EastAddressBus. WebSphere Application Server V6.0, V6.1, and V7 support JMS messaging through the service integration bus.



## Requirements

To complete this exercise, you need:

- Access to the **DataPower** appliance
- Access to a back-end WebSphere Application V7.0 server that is configured with a service integration bus called EastAddressBus
- **cURL**, to send requests to the DataPower appliance
- The East Address Search web service that is running on WebSphere Application Server by using the JMS transport
- Access to the `<lab_files>` directory

# Exercise instructions

## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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 the exercise instructions refer to the following values:
  - <mpgw\_jms\_client\_port>: multi-protocol gateway port for sending requests to WebSphere JMS.

## B.1. Using the WebSphere JMS transport

The East Address enterprise application that is deployed on WebSphere Application Server also contains an EJB implementation that can be invoked over JMS.

The WebSphere Application Server administrative console can be used to view the deployment configuration of this JMS web service.

- \_\_\_ 1. Log in to the WebSphere Application Server administrative console.
  - \_\_\_ a. Open a web browser window and enter the following URL:  
`http://<backend_server_ip>:<server_address_admin_p>/ibm/console/`
  - \_\_\_ b. Enter any user name to log in to the WebSphere Application Server administrative console.
- \_\_\_ 2. Examine the East Address EJB web service.
  - \_\_\_ a. In the navigation bar, select **Applications > Application Types > WebSphere enterprise applications**.
  - \_\_\_ b. Click the **EastAddress** enterprise application.
  - \_\_\_ c. Under Modules, click **Manage Modules** to view the EJB modules inside the enterprise application.
  - \_\_\_ d. Verify that the **EastAddress** modules are listed:

The EastAddressEJB project contains the implementation for the East Address EJB web service. The same web service that you invoke over HTTP is also implemented as an EJB. The EastAddressJMSRouter project contains code for receiving JMS requests and sending them to the East Address EJB web service. To enable support for receiving JMS messages, you must create a service integration bus (SIBus). The bus references existing JMS queues where it gets and puts messages.

- \_\_\_ 3. Examine the SIBus configuration
  - \_\_\_ a. In the navigation bar, select **Service Integration > Buses**.
  - \_\_\_ b. Select **EastAddressBus** to open the EastAddressBus configuration.



### Information

The DataPower JMS client sends and receives messages on a service integration bus. The EastAddressBus that you are viewing in this step is the name to use when you configure the DataPower WebSphere JMS object.

- \_\_\_ c. In the EastAddressBus configuration, select the **Destinations** link.
- \_\_\_ d. Verify that you see the following items listed:
  - EastAddressQueueReg

- EastAddressQueueResp

The `EastAddressQueueReq` and the `EastAddressQueueResp` are the destinations that are used to send and receive JMS messages. From the perspective of a JMS client, such as DataPower, it interacts with the SIBus by using these destinations.



### Information

When you configure the DataPower WebSphere JMS object, you use the `EastAddressQueueReq` as the request queue and the `EastAddressQueueResp` as the reply queue.

The JMS resources such as the JMS queue, connection factory, and activation specification also must be configured. However, a JMS client does not interact directly with these objects. It communicates only with the SIBus and its configured destinations. The SIBus invokes the JMS resources.

- 4. Obtain the service integration bus endpoint.
  - a. In the navigation bar, select **Servers > Server Types > WebSphere application servers**.
  - b. Select **server1** to open the server configuration.
  - c. Under **Communications**, expand **Ports**.
  - d. Look in the list of port numbers for the **SIB\_ENDPOINT\_ADDRESS** value. It has a value of **7276** (default).



### Information

The information from the SIBus is summarized here. You need this information when you configure the DataPower WebSphere JMS object.

- WebSphere JMS Host: `<backend_server_ip>`
- WebSphere JMS port: `<SIB_ENDPOINT_ADDRESS>` (the default value is 7276)
- WebSphere JMS Messaging Bus: `EastAddressBus`
- Request Queue: `EastAddressQueueReq`
- Reply Queue: `EastAddressQueueResp`

The SIBus configuration on WebSphere Application Server V7.0 is already implemented for this exercise.

## B.2. Create a multi-protocol gateway to invoke a web service over JMS

In this section, you create a multi-protocol gateway service that receives requests from an HTTP client and uses the JMS transport to send that message to the East Address search web service.

- \_\_\_ 1. In the DataPower WebGUI, click **Control Panel > Multi-Protocol Gateway**.
- \_\_\_ 2. Click **Add** to create the service.
- \_\_\_ 3. Configure the `EastAddressSearchJMSMPGW` service.
  - \_\_\_ a. Enter the name: `EastAddressSearchJMSMPGW`
  - \_\_\_ b. Verify that the service **type** is **static-back-end** and the **default** XML Manager is being used.
- \_\_\_ 4. Create a service policy called `EastAddressSearchJMSMPGWPolicy`.
  - \_\_\_ a. Click **+** (new) in the **Multi-Protocol Gateway Policy** field.
  - \_\_\_ b. Enter the name: `EastAddressSearchJMSMPGWPolicy`
  - \_\_\_ c. Click **New Rule** to create a rule.
  - \_\_\_ d. Double-click the **Match** action.
  - \_\_\_ e. Select an existing match all rule or create a matching rule that matches all URLs.
  - \_\_\_ f. Add a **Results** action after the **Match** action.
  - \_\_\_ g. Double-click the **Results** action. Verify that the INPUT context is moving to the Output context, and click **Done**.
  - \_\_\_ h. For rule direction, make sure that **Both Directions** is selected.
  - \_\_\_ i. Click **Apply Policy** and close the window.
- \_\_\_ 5. Create an HTTP front side handler that is called `EastAddressSearchHTTPClient2` and add it to the `EastAddressSearchJMSMPGW` gateway.
  - \_\_\_ a. Under **Front side settings**, select **New** (the **[+]** button) and select **HTTP Front Side Handler**.
  - \_\_\_ b. Enter the name `EastAddressSearchHTTPClient2` and port number `<mpgw_jms_client_port>`.
  - \_\_\_ c. Click **Apply**.
  - \_\_\_ d. Verify that this new handler is added as a front side protocol.
- \_\_\_ 6. Build the back-end WebSphere JMS back-end URL.
  - \_\_\_ a. Under Back side settings, click **WebSphereJMSHelper**.
  - \_\_\_ b. In the WebSphere JMS URL Builder web page, click **+** (new) to create a server object.

- \_\_\_ c. Click the **WebSphere JMS Endpoint** tab.
- \_\_\_ d. Click **Add** to create an endpoint.
- \_\_\_ e. Enter the following information:
  - **WebSphere JMS Host:** <backend\_server\_ip>
  - **WebSphere JMS Port:** <SIB\_ENDPOINT\_ADDRESS> (default is 7276)
  - **WebSphere JMS Transport Protocol:** TCP
- \_\_\_ f. Click **Apply**.
- \_\_\_ g. Click the **Main** tab to return to the first configuration page.
- \_\_\_ h. Enter the name: EastAddressSearchJMS
- \_\_\_ i. Scroll down to the **WebSphere JMS Messaging Bus** field and enter:  
EastAddressBus



### Information

The name of the service integration bus on WebSphere Application Server that receives JMS messages is `EastAddressBus`.

- \_\_\_ j. Leave the remaining fields on this page at their default value.
- \_\_\_ k. Click **Apply**.
- \_\_\_ l. Verify that the “URL Builder” window is in focus. Make sure that the `EastAddressSearchJMS` server object is selected
- \_\_\_ m. Enter the following information:
  - **Request Queue:** `EastAddressQueueReq`
  - **Reply Queue:** `EastAddressQueueResp`

### Build a WebSphere JMS URL

#### Server:

`EastAddressSearchJMS`

#### Request Queue:

`EastAddressQueueReq`

#### Reply Queue:

`EastAddressQueueResp`

- \_\_\_ n. Click **Build URL**. It generates the following URL:

```
dpwasjms://EastAddressSearchJMS/
?RequestQueue=EastAddressQueueReq&ReplyQueue=EastAddressQueueResp
```

- \_\_\_ o. Click **Apply** to save the `EastAddressSearchJMSMPGW` configuration up to this point.
- \_\_\_ 7. Continue with the multi-protocol gateway configuration. On the **Main** tab, turn off URI propagation to the back-end URL.
  - \_\_\_ a. When you use the internal DataPower **dpwasjms** protocol, you cannot pass the input URI to the back end. Set **Propagate URI** to **off**.
- \_\_\_ 8. Insert a WebSphere JMS header to identify the target service.

The IBM WebSphere web services runtime on WebSphere Application Server requires a custom JMS header to identify the web service that is invoked. In this step, you insert a custom JMS header called **targetService** with the value of `EastAddressJMS`. (The referenced service `EastAddressJMS` is defined in the application `webservices.xml`.)

  - \_\_\_ a. Click the **Headers** tab.
  - \_\_\_ b. Click **Add** under **Header Injection Parameters**.
  - \_\_\_ c. Create a header with the following values:
    - **Direction:** Back
    - **Header Name:** targetService
    - **Header Value:** EastAddressJMS
  - \_\_\_ d. Click **Submit**.
  - \_\_\_ e. Click **Apply**.
  - \_\_\_ f. Click **Save Config**.
- \_\_\_ 9. Use cURL to send an East Address Search web service request to the `EastAddressSearchJMSMPGW` service.
  - \_\_\_ a. Open a command prompt and navigate to the `<lab_files>/JMS` directory.
  - \_\_\_ b. Enter the command:

```
curl -H "Content-Type: text/xml" --data-binary @findByLocation.xml
http://<dp_appliance_ip>:<jmpgw_jms_client_port>/

```
  - \_\_\_ c. Verify that you obtain the correct response (`<Address>` elements with `<state>` of NC).

## End of exercise

## Exercise review and wrap-up

In the first part of the exercise, you examined the service integration bus configuration on WebSphere Application Server V7.0 to identify the relevant components that a JMS client requires. In the second part of the exercise, you created a multi-protocol gateway service that used the DataPower WebSphere JMS object to send requests to the East Address Search web service that is running over JMS.



# Appendix C. 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 appliance into a `.zip` file. You can import a sample solution into your domain.
- Reference documentation for defining the LDAP definitions.

## **Part 1: Dependencies**

Certain exercises are dependent on previous exercises, and on other resources, such as the need for the back-end application server to support service calls.

In all cases, the initial dependency is on **Exercise 1: Exercises setup**, where exercise setup, variable definition, and port numbering are done.

**Table 1: Dependencies**

Exercise	Depends on exercise	Uses curl	Uses Eclipse	Needs Application Server
1: Exercises setup		Yes	Yes	Yes*
2: Creating XSL transformations	1		Yes	
3: Creating a simple XML firewall	1	Yes		
4: Creating an advanced multi-protocol gateway	1	Yes		Yes
5: Adding error handling to a service policy	1,4	Yes		
6: Creating cryptographic objects	1		Yes*	
7: Using SSL to secure connections	1,4,6	Yes		
8: Protecting against XML threats	1,4	Yes		Yes
9: Configuring a web service proxy	1		Yes	Yes
10: Web service encryption and digital signatures	1,9	Yes		Yes
11: Web service authentication and authorization	1,9	Yes	Yes	Yes
12: Creating message count monitors for a AAA policy	1,9,11	Yes		Yes
13: Creating a AAA policy by using LDAP	1, 9,11	Yes	Yes*	Yes
14: Implementing an SLM monitor in a web service proxy	1, 9	Yes		Yes
15: Configuring a multi-protocol gateway service with WebSphere MQ	1, 4	Yes		Yes
App A: Creating a firewall and HTTP proxy for a web application	1	Yes		Yes

**Table 1: Dependencies**

<b>Exercise</b>	<b>Depends on exercise</b>	<b>Uses cURL</b>	<b>Uses Eclipse</b>	<b>Needs Application Server</b>
App B: Configuring WebSphere JMS	1	Yes		Yes

If the class is using the standard images and setup, the East and West Address enterprise applications and the LDAP service are running on a separate back-end server image. Only cURL and Eclipse are on the student image.

In addition, Exercise 12 requires IBM Tivoli Directory Server.

Exercise 1 needs access to WebSphere Application Server only if it is run on the student's machine, rather than on a separate server machine.

Exercise 6 requires Eclipse, but the real requirement is for the Java `keytool.exe` in the `eclipse` subdirectory.

## 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: Exercises setup	
2: Creating XSL transformations	dev_Ex02_XSLTFirewall.zip
3: Creating a simple XML firewall	dev_Ex03_SimpleXMLfirewall.zip
4: Creating an advanced multi-protocol gateway	dev_Ex04_AdvMPG.zip
5: Adding error handling to a service policy	dev_Ex05_Errors.zip
6: Creating cryptographic objects	dev_Ex06_CryptoTools.zip
7: Using SSL to secure connections	dev_Ex07_SSL.zip
8: Protecting against XML threats	dev_Ex08_XMLthreat.zip
9: Configuring a web service proxy	dev_Ex09_WSP.zip
10: Web service encryption and digital signatures	dev_Ex10_WSP_security.zip
11: Web service authentication and authorization	dev_Ex11_AAA.zip
12. Creating message count monitors for a AAA policy	dev_Ex12_Counter.zip
13: Creating a AAA policy by using LDAP	dev_Ex13_LDAP.zip
14: Implementing an SLM monitor in a web service proxy	dev_Ex14_SLM.zip
15: Configuring a multi-protocol gateway service with WebSphere MQ	dev_Ex15_MQ.zip
App A: Creating a firewall and HTTP proxy for a web application	dev_ExA_WAF.zip
App B: Configuring WebSphere JMS	dev_ExB_JMS.zip

- \_\_\_ a. The .zip file names begin with the naming convention ExNN, where NN represents the two-digit exercise number.
- \_\_\_ b. To import a solution to begin a new exercise, import the solution for the previous exercise. For example, if you are ready to start Exercise 10, you would import <lab\_files>/solutions/dev\_Ex10\_WSP\_security.zip.
- \_\_\_ 2. Import the .zip solution file into your application domain.
- \_\_\_ a. From the Control Panel, in the vertical navigation bar, select **Administration > Configuration > Import Configuration**.

- \_\_\_ b. Make sure that **From** has **ZIP Bundle** selected and **Where** has **File** selected.
  - \_\_\_ c. Click **Browse** and navigate to your respective .zip solution file.
  - \_\_\_ d. Click **Next**.
  - \_\_\_ e. In the next page, leave the files selected. Scroll down and click **Import**.
  - \_\_\_ f. Make sure that the import is successful. Click **Done**.
- \_\_\_ 3. Be sure to update the port numbers and application server location to your local values. Since 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. Upon import, you move those files into the `cert:` directory.



**IBM**  
®