

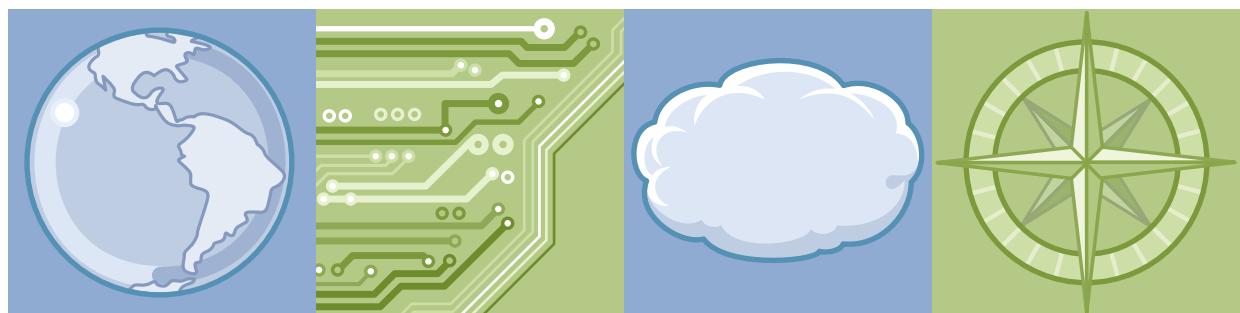


IBM Training

Student Exercises

Accelerate, Secure and Integrate with IBM DataPower V6

Course code WE601 / ZE601 ERC 1.1



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.

June 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

Trademarks	vii
Exercises description	xi
Exercise 1. Exercise setup	1-1
1.1. Define variables and ports	1-3
1.2. Identify the student image IP address	1-5
1.3. Upload the crypto key and certificate files	1-6
1.4. Upload the WSDL files to the appliance	1-8
1.5. Launch Eclipse	1-9
1.6. Import the East and West Address WSDL files into Eclipse	1-10
Exercise 2. Creating a simple XML firewall	2-1
2.1. Validation: XML firewall creation	2-5
2.2. Validation: XML firewall examination and configuration	2-10
2.3. Validation: XML firewall testing	2-18
2.4. Transformation: XML firewall creation	2-21
2.5. Transformation: XML firewall configuration	2-22
2.6. Transformation: XML firewall testing	2-27
2.7. Integration: Edit the MyBasicFirewall policy	2-31
2.8. Integration: XML firewall testing	2-32
Exercise 3. Creating an advanced multi-protocol gateway	3-1
3.1. Create a basic MPGW to validate SOAP messages	3-5
3.2. Create a multi-protocol gateway for WestAddressSearch	3-27
3.3. Configure an HTTP front side protocol handler	3-32
3.4. Create an MPGW for content based routing	3-34
3.5. Perform end-to-end content-based routing scenario testing	3-41
Exercise 4. Adding error handling to a service policy	4-1
4.1. Adding error processing	4-3
Exercise 5. Creating cryptographic objects and configuring SSL	5-1
5.1. Generate a certificate-key pair on the DataPower appliance	5-4
5.2. Creating cryptographic objects	5-8
5.3. Verify web service behavior	5-12
5.4. Add an HTTPS handler to the EastAddressSearch service	5-13
5.5. Test the EastAddressSearch HTTPS access	5-15
5.6. Create the SSL client support in the AddressRouter service	5-16
5.7. Test the SSL connection from the AddressRouter to the EastAddressSearch	5-19
Exercise 6. Configuring a web service proxy	6-1
6.1. Create a web service proxy for EastAddressSearch web service	6-5
6.2. Add a WSDL to the web service proxy for the WestAddressSearch web service	6-9
6.3. Verify the generated components	6-10
6.4. Test the EastAddressSearch web service	6-15
6.5. Test the WestAddressSearch web service	6-20

6.6. Add an operation-level rule to West Address Search web service proxy	6-22
Exercise 7. Implementing an SLM monitor in a web service proxy	7-1
7.1. Test the existing AddressSearchProxy by using the test script	7-4
7.2. Create a log target for SLM log messages	7-7
7.3. Add SLM criteria to the web service proxy	7-9
7.4. Run the test script with SLM criteria in effect	7-11
7.5. Add an SLM action to a port-operation request rule	7-14
7.6. Run the test script with the operation-level SLM action	7-16
7.7. View the SLM policy object	7-17
7.8. Examine the graph behavior (optional)	7-18
Exercise 8. Web service encryption and digital signatures	8-1
8.1. Create cryptographic objects	8-4
8.2. Examine the East Address Search web service proxy	8-6
8.3. Create a multi-protocol gateway to encrypt messages	8-7
8.4. Create a rule to sign messages	8-14
8.5. Configure message decryption on the web service proxy	8-16
8.6. Configure field-level message decryption on the web service proxy	8-19
8.7. Configure message verification on the web service proxy	8-23
8.8. Send a signed and encrypted message to the web service proxy	8-26
Exercise 9. Web service authentication and authorization	9-1
9.1. Test the East Address Search web service	9-4
9.2. Configure a AAA policy action for a web service operation	9-7
9.3. Test the access control policy for a web service operation	9-13
9.4. Test the access control policy with a web services security user name token	9-16
9.5. Configure the access control policy to handle SAML assertions	9-18
9.6. Test the access control policy with a SAML assertion	9-22
Exercise 10. Define a three-legged OAuth scenario that uses DataPower services	10-1
10.1. The three-legged architecture for this exercise	10-4
10.2. Prepare the security objects	10-5
10.3. Define the OAuth client	10-7
10.4. Define the OAuth client group	10-10
10.5. Define the AAA policy for the web token service	10-11
10.6. Define the web token service	10-12
10.7. Define the AAA policy for an enforcement point	10-13
10.8. Define the enforcement point and resource server	10-14
10.9. Define the back-end service	10-16
10.10. Test the solution	10-18
Exercise 11. Using a DataPower pattern	11-1
11.1. Import a pattern into your application domain	11-4
11.2. Deploy a pattern	11-6
11.3. Test the generated service	11-11
Exercise 12. Configuring a multi-protocol gateway service with WebSphere MQ	12-1
12.1. Obtain the WebSphere MQ configuration	12-4
12.2. Verify MQ install and Availability	12-5
12.3. Create a WebSphere MQ front side handler (FSH)	12-10

12.4. Create a multi-protocol gateway that completes one-way messaging	12-14
12.5. Use the WebSphere MQ FSH to call the EastAddressSearch web service	12-19
12.6. Configuring transaction capability on the DataPower WebSphere MQ queue manager	12-21
Exercise 13. Implementing REST services using DataPower	13-1
13.1. Compare the REST interface to the SOAP interface of the back-end web service	13-4
13.2. The high-level design of the REST service	13-9
13.3. Create the request side of the RESTful multi-protocol gateway service	13-10
13.4. Create the findByName response side of the RESTful multi-protocol gateway service	13-19
13.5. Create the findByLocation response side of the RESTful multi-protocol gateway service	13-23
13.6. Create the retrieveAll response side of the RESTful multi-protocol gateway service ..	13-27
13.7. Catch incorrect requests to the RESTful multi-protocol gateway service	13-29
Appendix A. Exercise solutions	A-1

Trademarks

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

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

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

Approach®	BladeCenter®	DataPower®
DB™	DB2 Connect™	DB2®
developerWorks®	Express®	IMS™
Notes®	RACF®	Rational®
RDN®	Redbooks®	Tivoli®
WebSphere®	z/OS®	400®

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.

Lenovo and ThinkPad are trademarks or registered trademarks of Lenovo in the United States, other countries, or both.

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 a simple XML firewall:** Creates an XML firewall that does validation and transformation actions.
3. **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.
4. **Adding error handling to a service policy:** Uses either the On Error action or the error rule to prepare for problems.
5. **Creating cryptographic objects and configuring SSL:** 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. These cryptographic objects that are used to secure communications to and from the DataPower appliance.
6. **Configuring a web service proxy:** Virtualizes both the East and West Address search web services.
7. **Implement an SLM monitor in a web service proxy:** Adds an SLM monitor to the web service proxy.
8. **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.
9. **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).
10. **Define a three-legged OAuth scenario that uses DataPower services:** Demonstrates the DataPower support of OAuth 2.0.
11. **Using a DataPower pattern:** Creates a DataPower service from a supplied pattern. The pattern references a AAA policy that uses LDAP.

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.

12. **Implementing REST services using DataPower:** Uses a multi-protocol gateway to convert a client REST request into a SOAP message to the web service proxy.

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 extra challenge.

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>*: /home/localuser/dplabs/dev
 - *<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.

- 1. Complete the following table with the values supplied by the instructor.

Table 1-1: Variable and port assignments

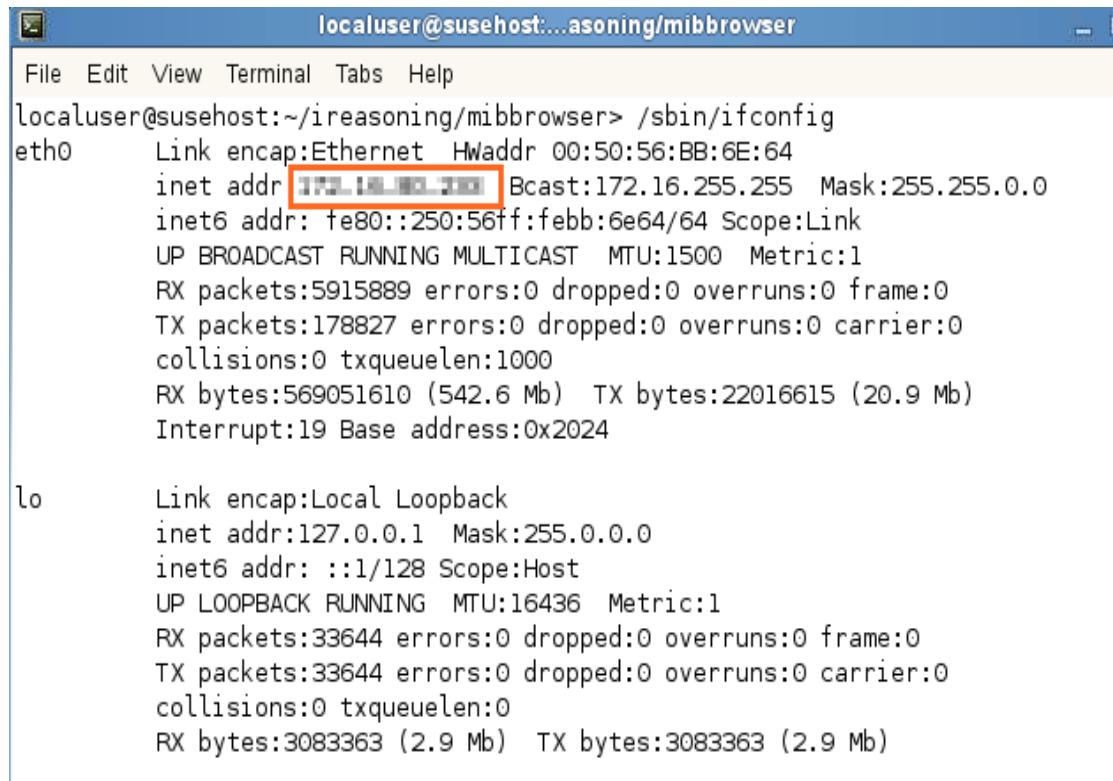
Object	Value (default)
Lab files location	
<lab_files>	/home/localuser/dplabs/dev
Location of student lab files for WE601	
Student information	
<nn>	
<studentnn>	
<studentnn_domain>	
<studentnn_password>	student<nn>
<studentnn_updated_password>	
<studentnn_image_ip>	
WSserver<nn>	
Logins that are not DataPower	
<linux_user>:	localuser
<linux_user_password>	websphere
<linux_root_user>	root
<linux_root_password>	
DataPower information	
<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	
Server information	

<backend_server_ip> IP address for the services that run in WebSphere Application Server (AddressSearch web services, LDAP, registry)	
<studentnn_image_ip>	127.0.0.1
<ldap_password>	websphere
<ldap_server_port> Port number for the LDAP administration console	9080
<ldap_server_root_dir>	/opt/ibm/ldap/V6.3/bin
<ldap_user_name>	cn=root
<http_server_port>	80
<http_server_root_dir>	/opt/IBM/HTTPServer/
<logger_app_port>	1112
<was_server_port> Port number for the services that run in WebSphere Application Server (AddressSearch web services, LDAP, registry)	9080

1.2. Identify the student image IP address

On Linux, you can discover the IP address by running the `ifconfig` command. From within a terminal window, run the `ifconfig` command that includes the full path, as follows:

```
> /sbin/ifconfig
```



```
localuser@susehost:~/ireasoning/mibbrowser> /sbin/ifconfig
eth0      Link encap:Ethernet HWaddr 00:50:56:BB:6E:64
          inet addr 172.16.255.200 Bcast:172.16.255.255 Mask:255.255.0.0
                      inet6 addr: fe80::250:56ff:febb:6e64/64 Scope:Link
                        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                        RX packets:5915889 errors:0 dropped:0 overruns:0 frame:0
                        TX packets:178827 errors:0 dropped:0 overruns:0 carrier:0
                        collisions:0 txqueuelen:1000
                        RX bytes:569051610 (542.6 Mb) TX bytes:22016615 (20.9 Mb)
                        Interrupt:19 Base address:0x2024

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

When the IP address of the local student image is obtained, update the information in table 1.1 for the variable `<studentnn_image_ip>`.

1.3. Upload the crypto key and certificate files

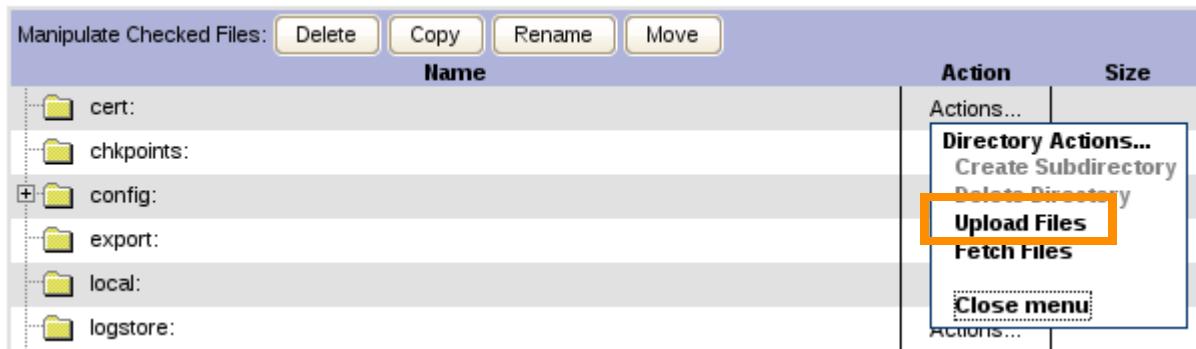
In this section, you upload a crypto private key and certificate that are used in the exercises in this course.

- ___ 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>).



- ___ 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” pane.



- ___ 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” pane.
- ___ 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.

1.4. 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” pane.
 - 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” pane.
 - 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.

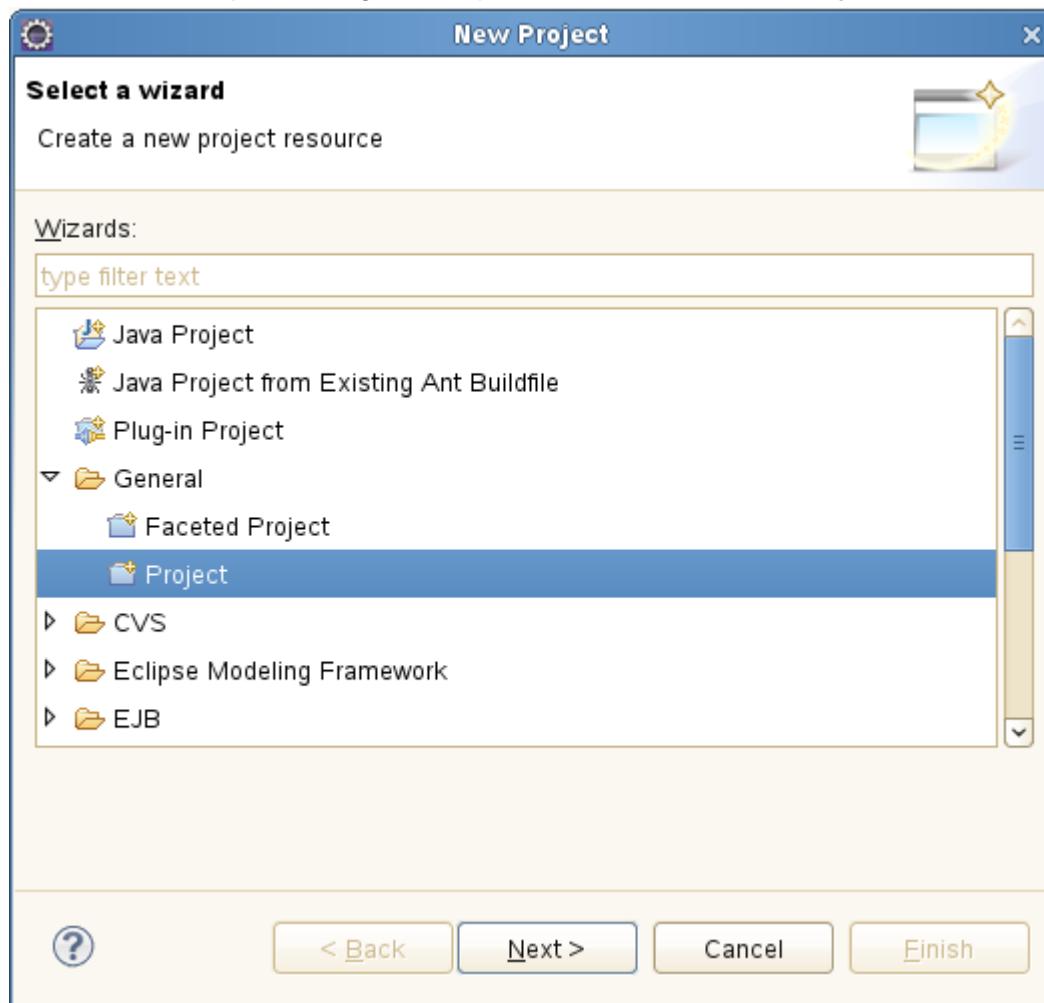
1.5. Launch Eclipse

- ___ 1. Check the desktop for a **Link to Eclipse** icon, or verify that Eclipse is installed at /home/localuser/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 64-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 start 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.

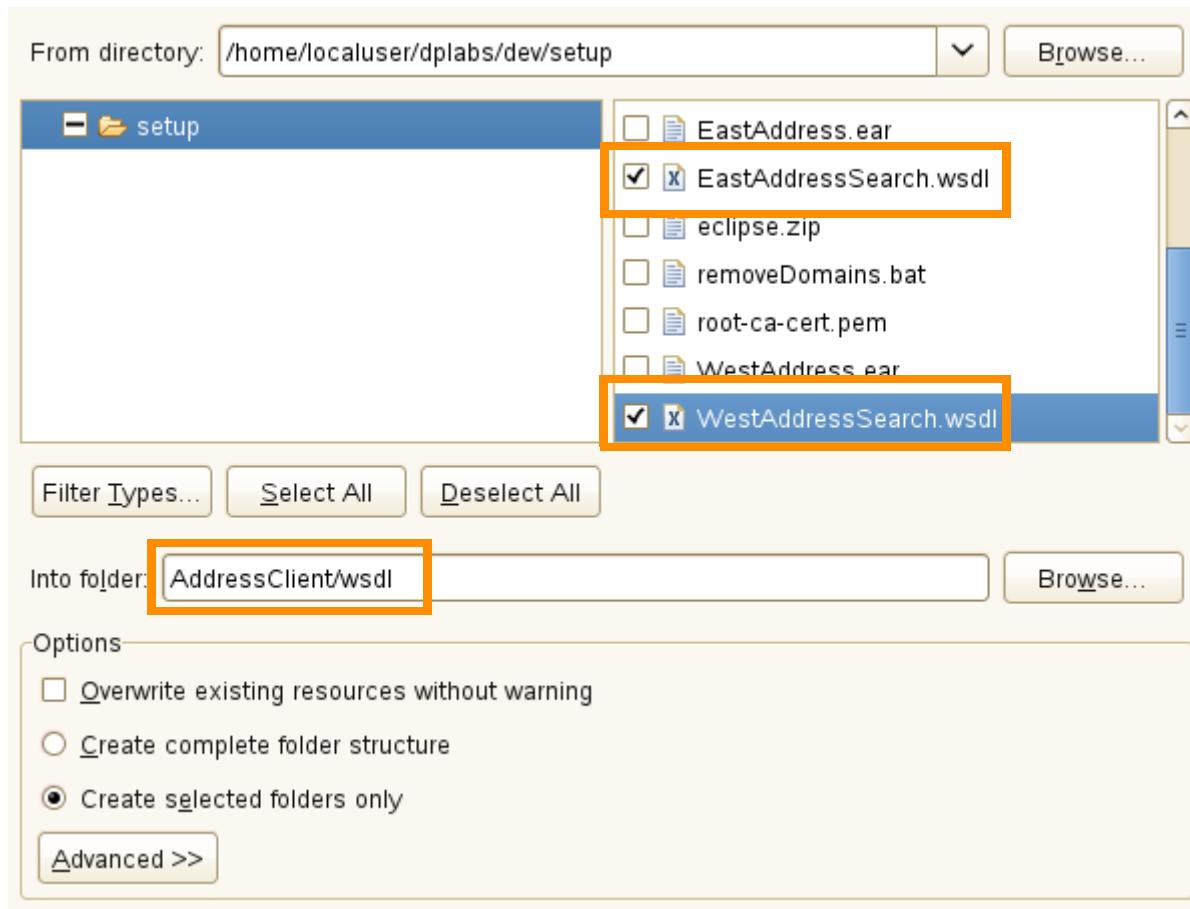
1.6. Import the East and West Address WSDL files into Eclipse

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 box, 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. Click **Window > Open Perspective > Resource**.
 - b. Click **OK**.
- 3. Create a simple project called **AddressClient**.
 - a. Click **File > New > Project**.
 - b. In the “New Project” dialog box, 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. Click **File > Import**.
 - ___ c. In the Import dialog box, 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. Click **Properties** from the menu.
 - ___ d. On the Properties dialog box, 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.

Ensure that the enterprise applications are running WebSphere Application Server should be running on your local machine. WebSphere Application Server is used to provide the web services support for the exercises. In this step, you ensure that the EastAddressSearch and WestAddressSearch enterprise applications are running on the image.

- ___ 1. Open Firefox and confirm that the EastAddress and WestAddress web services are running on WebSphere Application Server.
 - ___ a. Open Firefox.
 - ___ b. Enter the following URL in Firefox and confirm the West Address web service is running in WebSphere Application Server:
http://localhost:9080/WestAddress/services/AddressSearch
 - ___ c. Enter the following URL in Firefox and confirm the East Address web service is running in WebSphere Application Server:
http://localhost:9080/EastAddress/services/AddressSearch
 - ___ d. If the web services (EastAddress and WestAddress) are not responding, contact your instructor.
- ___ 2. Use the Firefox bookmarks to start the same web services. If the URLs are incorrect, update them.

End of exercise

Exercise review and wrap-up

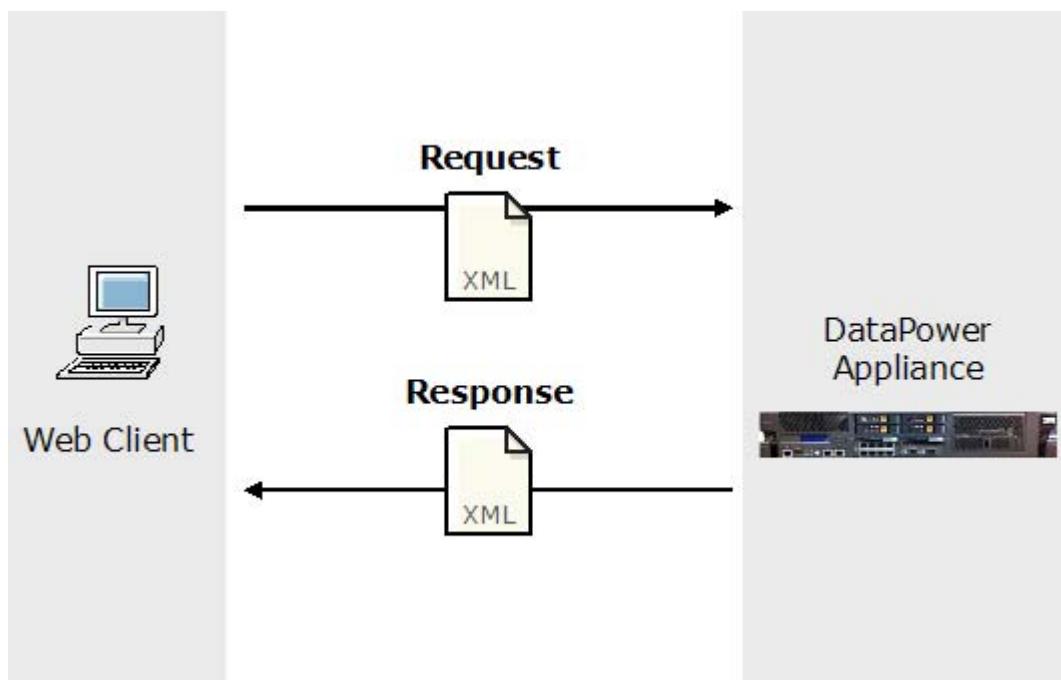
You determined all of the port number and variable assignments that are used in subsequent exercises. You also uploaded the Alice private key and certificate for later crypto work, and the WSDL files for later web services work. Last, you imported WSDLs into Eclipse for later testing.

Exercise 2. Creating 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

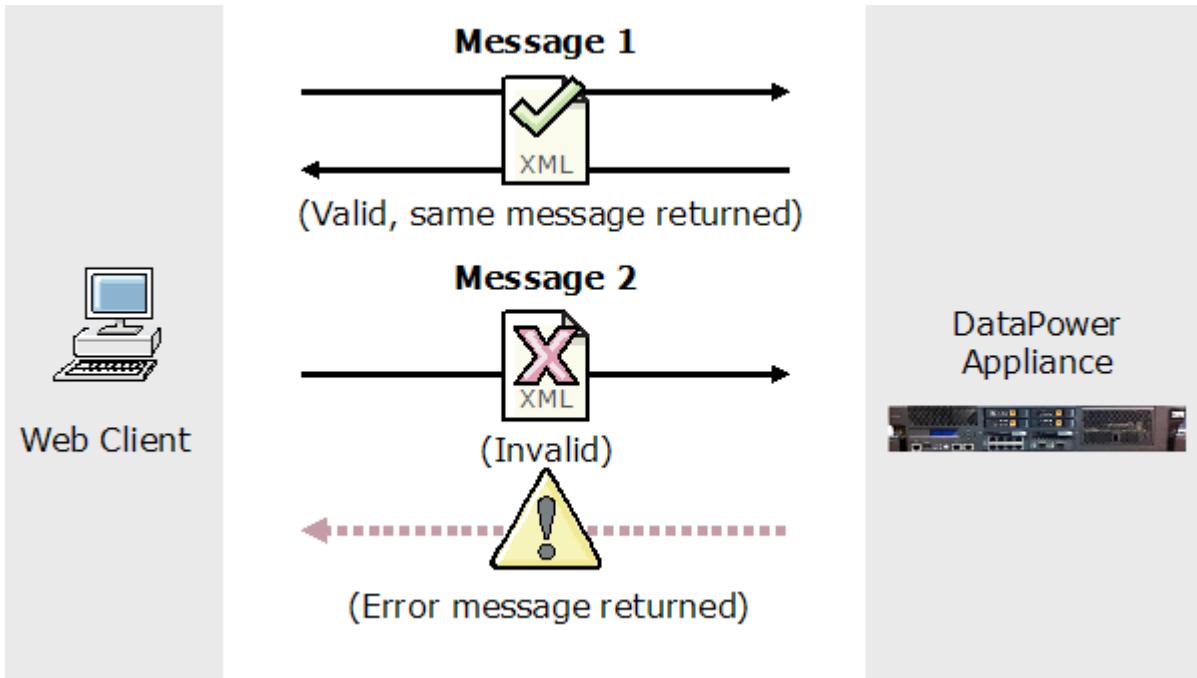
Exercise instructions

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 unless directed.
- The references in exercise instructions refer to the following values:
 - <lab_files>: /home/localuser/dplabs/dev
 - <studentnn_image_ip>: IP address of the image.
 - <dp_admin_password>: DataPower secondary administrator password.
 - <dp_internal_ip>: IP address of the appliance’s management interfaces.
 - <dp_public_ip>: IP address of the public services on the appliance.
 - <dp_WebGUI_port>: WebGUI port number of the DataPower appliance. The default port is 9090.
 - <studentnn>: Student developer user account.
 - <studentnn_domain>: Student application domain.
 - <xmfw_validation_port>: Port on which the BasicValidationXML service listens (6nn0).
 - <xmfw_transform_port>: Port on which the BasicTransformationXML service listens (6nn1).

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

2.1. Validation: XML firewall creation

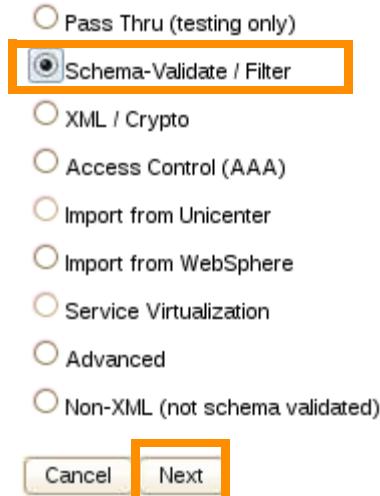
- 1. Use a web browser to log on to the DataPower WebGUI.
- a. Use your assigned student user account and domain to log in to the WebGUI (https://<dp_internal_ip>:<dp_WebGUI_port>). Be sure that you are **not** in the default domain.
- b. When you are logged in, take some time to familiarize yourself 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 usage 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 the name: MyBasicFirewall

A screenshot of a configuration dialog for a new firewall. It has fields for 'Firewall Name' containing 'MyBasicFirewall' and 'Attempt to use streaming mode' with the 'off' radio button selected. At the bottom are 'Cancel' and 'Next' buttons.

Firewall Name
MyBasicFirewall *

Attempt to use streaming mode.
 on off *

Cancel Next

- ___ 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 configuration dialog for the firewall type. A dropdown menu shows 'loopback-proxy' selected. At the bottom are 'Back', 'Cancel', and 'Next' buttons.

Firewall Type
loopback-proxy *

Back Cancel Next

- ___ 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
 *

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 **Validate Document via Schema URL**.



Note

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

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

Validation Information

Select the validation method.

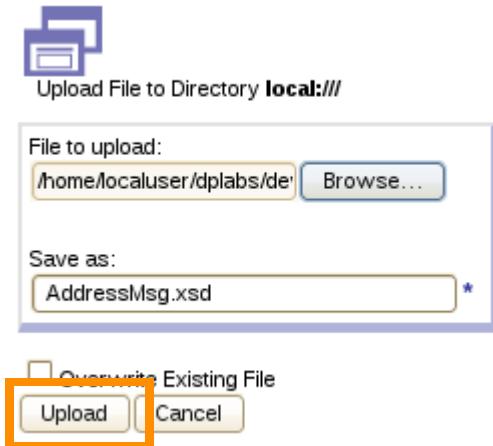
Schema Validation Method

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

Schema URL

*

- __ j. On the File Management dialog box, click **File** and click **Browse**.
- __ k. From the file selection dialog box, select <lab_files>/simpleFW/AddressMsg.xsd, which is the intended schema.
- __ l. The File Management dialog box updates the file name in the **File to upload** field. Click **Upload**.



An upload confirmation opens.

- __ m. 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.

- __ n. Click **Next**. You get a “Confirm Your Changes and Commit” page.

___ o. 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**

___ p. Click **Done**. You are returned to the Control Panel.

2.2. 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 pane is the general editor pane 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 corner of the wizard 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 → Diamond → Match → Route → ORIGIN SERVER

Create Reusable Rule

Order	Rule Name	Direction	Actions
1	MyBasicFirewall_request	Client to Server	Match, Route, delete rule

[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.



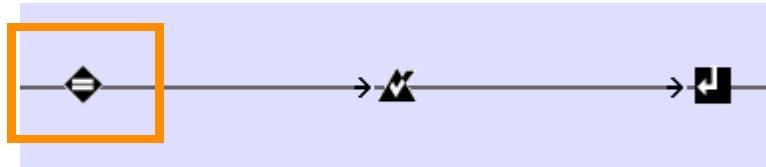
Each rule has a Match action at its beginning to determine whether this rule applies. Therefore, multiple rules can be specified for a single policy, and the Match action for each rule determines its applicability. Rules also have directionality. The single rule here applies only as the request enters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

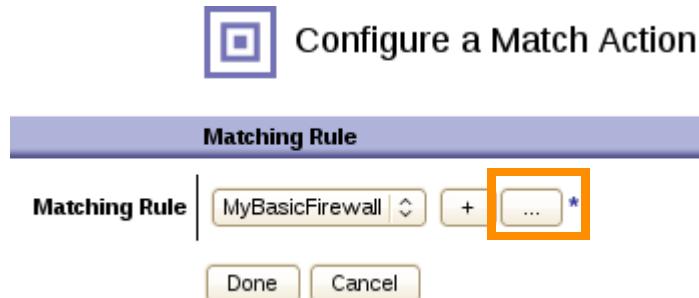
**Information**

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.

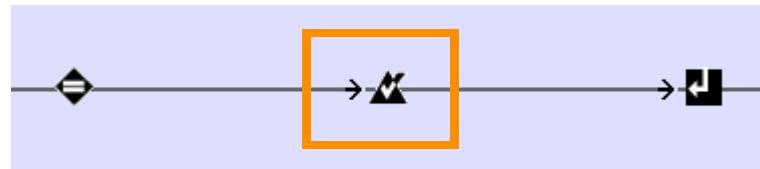
Matching Rule						
Matching Type	HTTP Header Tag	HTTP Value Match	URL Match	Error Code	XPath Expression	HTTP Method
URL			*			default

**Note**

Notice that the wizard used the same name, **MyBasicFirewall**, for multiple resources in your service: service name, policy name, matching rule name. This name 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. In 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. The validation does not change the document and hence the **output** field is empty.

Configure Validate Action

[Help](#)

Basic [Advanced](#)

Input

Input *

Options

Validate

Schema Validation Method

- Validate Document via Attribute Rewrite Rule
- Validate Document via JSON Schema URL
- Validate Document via Schema Attribute
- Validate Document via Schema URL
- Validate Document via WSDL URL
- Validate Document with Encrypted Sections *

Schema URL

SOAP Validation

Asynchronous

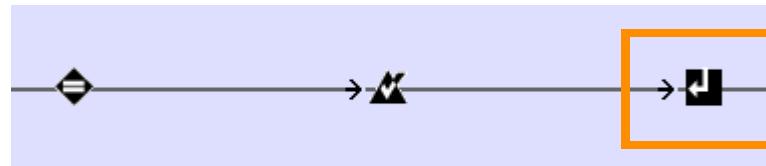
on off

Output

Output

- ___ 10. Click **Cancel** to close the window.

- ___ 11. In 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 (the output document from the rule processing).

Configure Results Action [Help](#)

Basic [Advanced](#)

Input

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

Options

Results

Destination	<input type="text" value="cert:///"/> <input type="button" value="cert:///"/>
	<input type="text" value="(none)"/> <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"/>
Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off
Number of Retries	<input type="text" value="0"/>
Retry Interval	<input type="text" value="1000"/> msec
Method	<input type="button" value="POST"/> *

Output

Output	<input type="text"/> <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. In 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 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. In the firewall editor window, click **View Status**.



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

name	config-state	op-state	admin-state	detail	logs
XML Firewall Service					
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.

2.3. 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 a text editor and review it before you run this test case.
 - b. On your Linux desktop, open a command-line terminal by clicking **Computer > Gnome Terminal**.
 - c. Go to the `<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>
```

Where,

`-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 that is 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**

2.4. 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, no wizards target transformation. So, for this section, you use the firewall service editor. Click **Add Advanced**.
- d. The editor page that is similar to the one you examined in the previous section is displayed. For this new firewall service, specify the following values:
 - **Firewall Name:** MyTransformFirewall
 - **Firewall Type:** Loopback (can cause the page to reload)
 - **Local IP address:** <dp_public_ip>
 - **Device Port:** <xmfw_transform_port>
 - **Request Type:** XML



Note

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

2.5. Transformation: XML firewall configuration

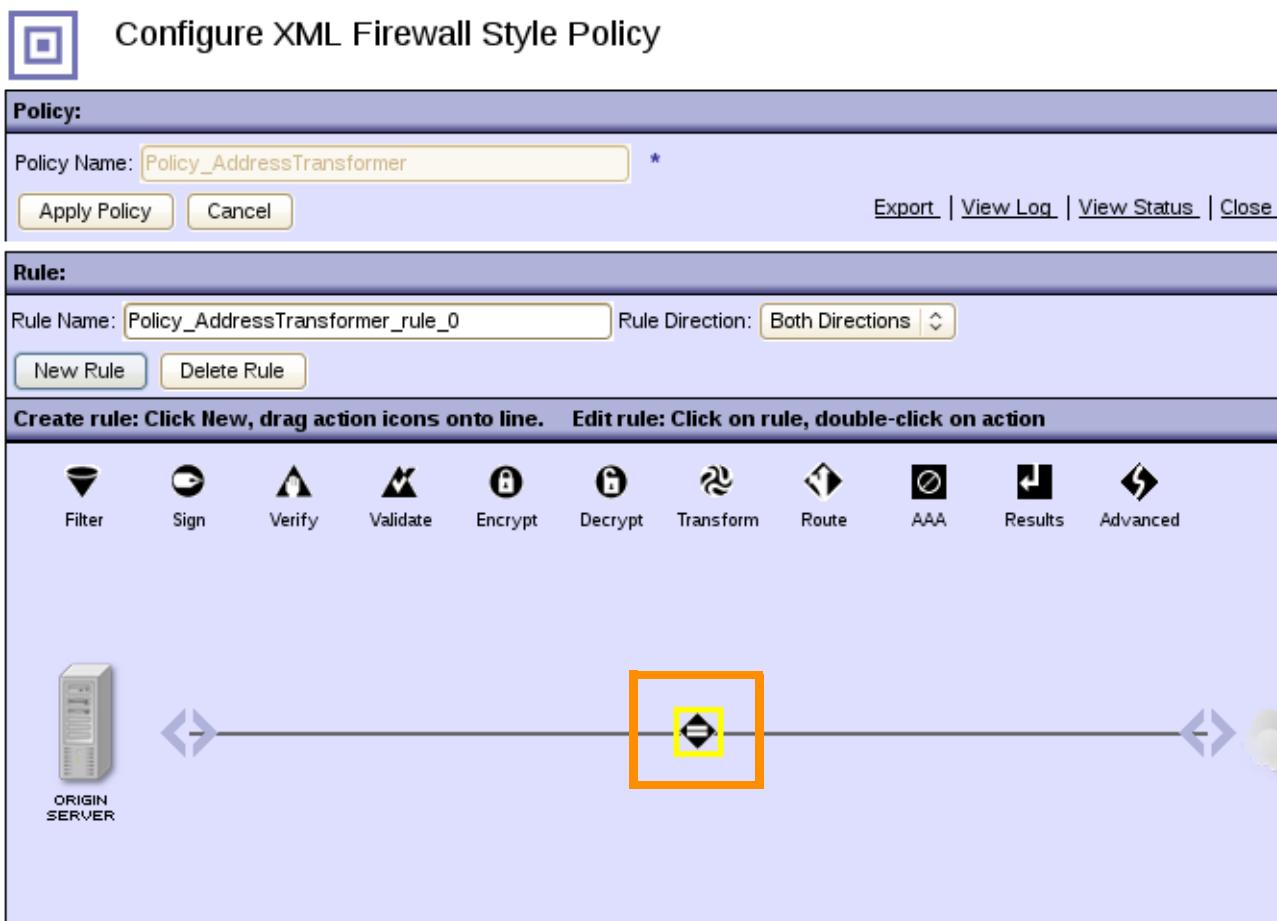
In this step, you create the firewall policy (document processing policy) to do XSL transformation on an incoming message.

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



- __ b. 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.



- ___ 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.

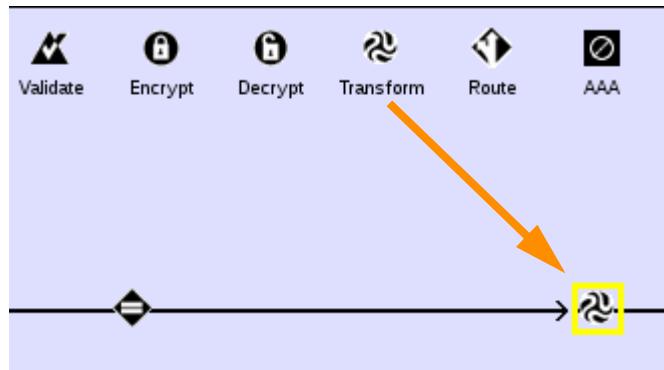


Note

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.

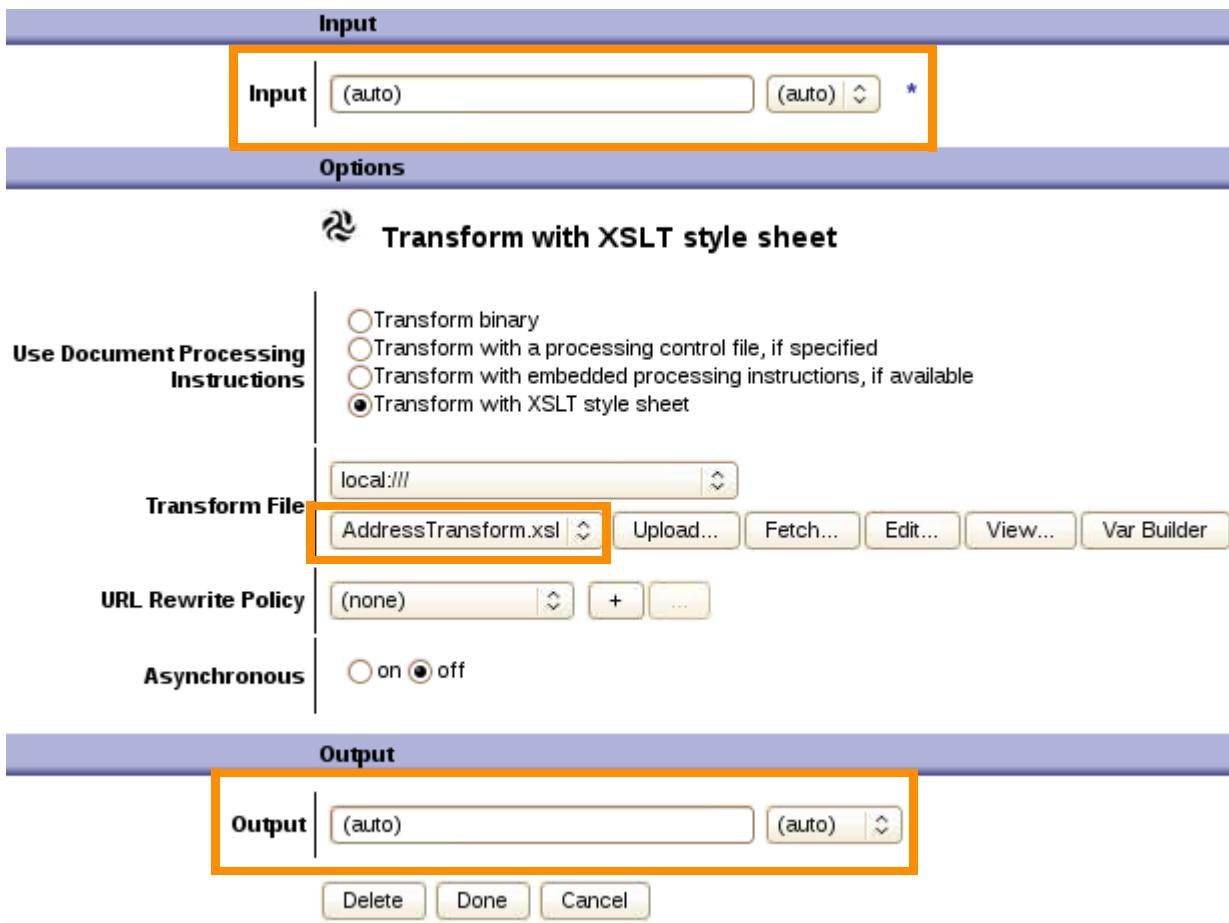
2. Add a **Transform** action to the processing steps of the **AddressTransformerPolicy** policy.

- 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 **Transform with XSLT style sheet**. 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. Click **Browse** to search and select the `AddressTransform.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 step ensures 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 step ensures 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. In 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.

2.6. 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></table></body></html>
```

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

- c. Undo your changes from `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 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 that order. This approach involves running **section 1** and **section 2** sequentially. However, instead of each of these actions that have 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**

2.7. 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**. The Configure XML Firewall page for the MyBasicFirewall service is displayed.
- ___ 3. Click **Edit** (the [...] button) to modify the MyBasicFirewall Firewall Policy. The MyBasicFirewall policy editor opens.
- ___ 4. Drag on the **Results** action to the **Delete** (trash can) icon.
- ___ 5. Drag the **Transform** icon to the right of the **Validate** icon on the configuration path.
- ___ 6. Double-click the **Transform** icon to open the “Configure Transform Action” window.
- ___ 7. Click **Transform with XSLT style sheet**.
- ___ 8. 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`.
- ___ 9. Verify that the **Input** field is set to **auto**. In this case, the action picks up the document from the INPUT context.
- ___ 10. Verify that the **output** field is also set to **auto**. The transformed document is moved to the output context.
- ___ 11. Click **Done**. The rule configuration path window reopens.
- ___ 12. Click **Apply Policy** and then after the window refreshes, click the **Close Window** link. You are now back on the Configure XML firewall page.
- ___ 13. Click **Apply** to save the changes.
- ___ 14. Click **Save Config** to save the XML firewall service properties to the startup configuration.

2.8. 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 clicking **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 click **gedit**.
 - ___ 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 3. 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:

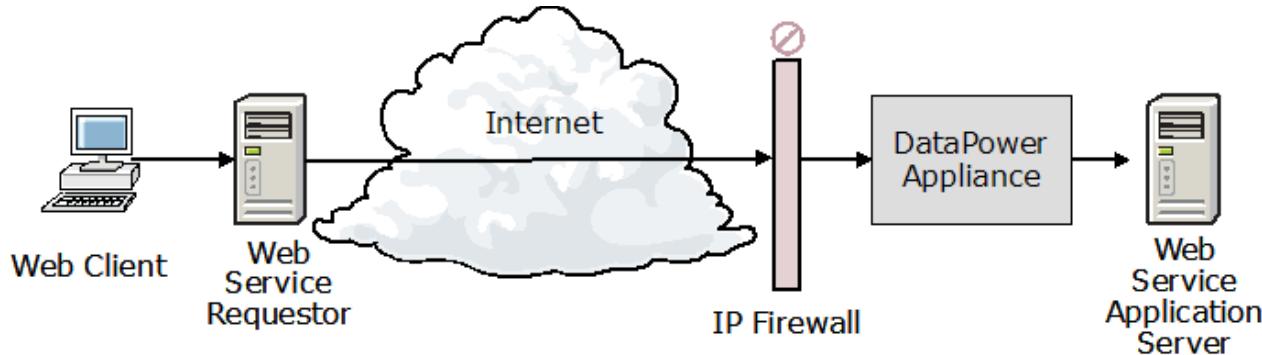
- Create an MPGW from a WSDL definition
- Configure a document processing policy with more actions
- Configure content-based routing by using a Route action
- Test the MPGW policy by using the command-line tool cURL
- Perform basic debugging by using the system log and multistep probe

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.



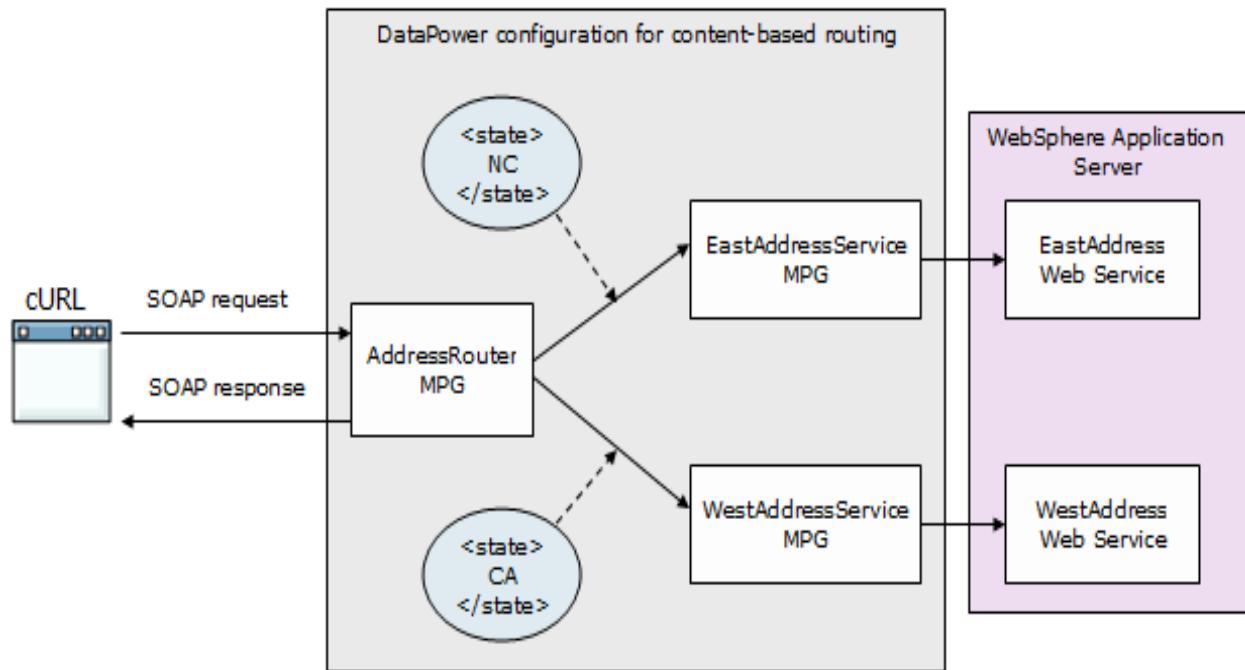
Requirements

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 is placed in front of 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

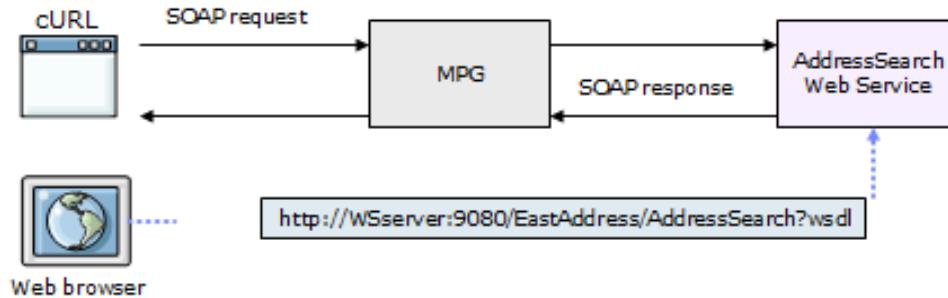
Preface

- 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. The default port is 9090.
- *<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). The default port is 9080.
- *<mpgw_east_port>*: instructor-assigned port number for the MPGW that handles the East Address web service: 6nn7.
- *<mpgw_west_port>*: instructor-assigned port number for the MPGW for the West Address web service: 6nn8.
- *<mpgw_content_based_routing_port>*: instructor-assigned port number for the MPGW that provides the route function: 6nn9.
- *<localhost_address>*: IP address that points to the local system itself (usually 127.0.0.1)

3.1. 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.



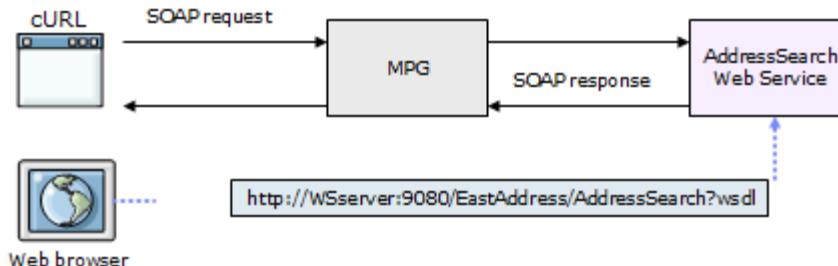
This lab implements the scenario that is described in four essential steps:

- 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 create and configure the MPGW.
- a. Using any text editor such as gedit, open and review the `EastAddressSearch.wsdl` file that is 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](#)

General Configuration

Multi-Protocol Gateway Name: *

Summary:

Type: dynamic-backend static-backend

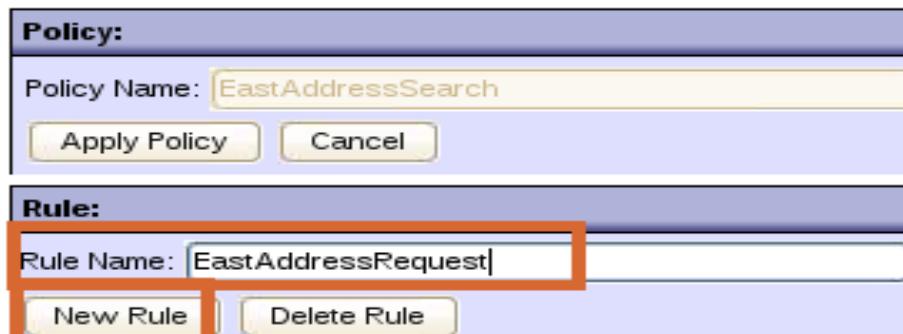
XML Manager: [...](#) * [+](#)

Multi-Protocol Gateway Policy: [+](#) [...](#) *

URL Rewrite Policy: [+](#) [...](#)

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

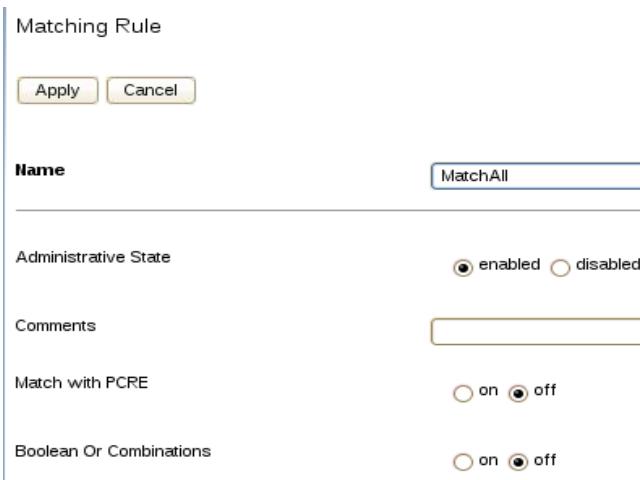
- ___ 7. Configure a message processing rule to forward any client request message to the back-end service.
- ___ a. In the processing rule editor for EastAddressSearch, click **New Rule** and rename the rule to: EastAddressRequest



- ___ 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



- ___ 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 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.



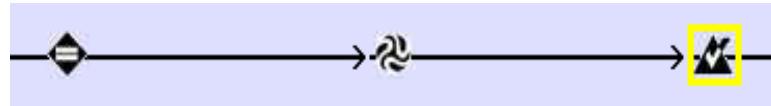
- __ b. Double-click the **Transform** action to open the **Transform** action configuration window.
- __ c. Verify that **Transform with XSLT style sheet** is selected.
- __ d. Click **Upload** to upload a style sheet that is called `<lab_files>\AdvMPGW\Address-EastRenameNamespace.xsl` that copies the 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 following values
 - **Use Document Processing Instructions:** Transform with XSLT style sheet
 - **Transform File:** Address-EastRenameNamespace.xsl
 - **URL Rewrite Policy:** (none)
 - **Asynchronous:** off

- **Output:** (auto)

Transform with XSLT style sheet

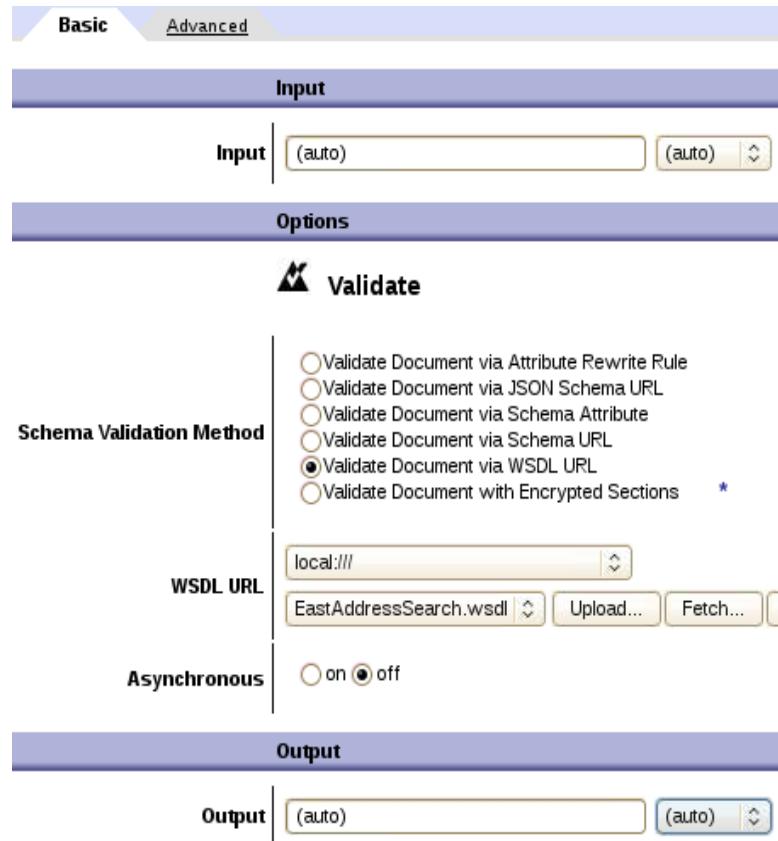
Use Document Processing Instructions	<input type="radio"/> Transform binary <input type="radio"/> Transform with a processing control file, if specified <input type="radio"/> Transform with embedded processing instructions, if available <input checked="" type="radio"/> Transform with XSLT style sheet
Transform File	local:/// Address-EastRenameNamespace.xsl <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"/>
URL Rewrite Policy	(none) <input type="button" value="+"/> <input type="button" value="..."/>
Asynchronous	<input type="radio"/> on <input checked="" type="radio"/> off
Output	
Output	(auto) <input type="button" value="Delete"/> <input type="button" value="Done"/> <input type="button" value="Cancel"/>

- ___ 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.



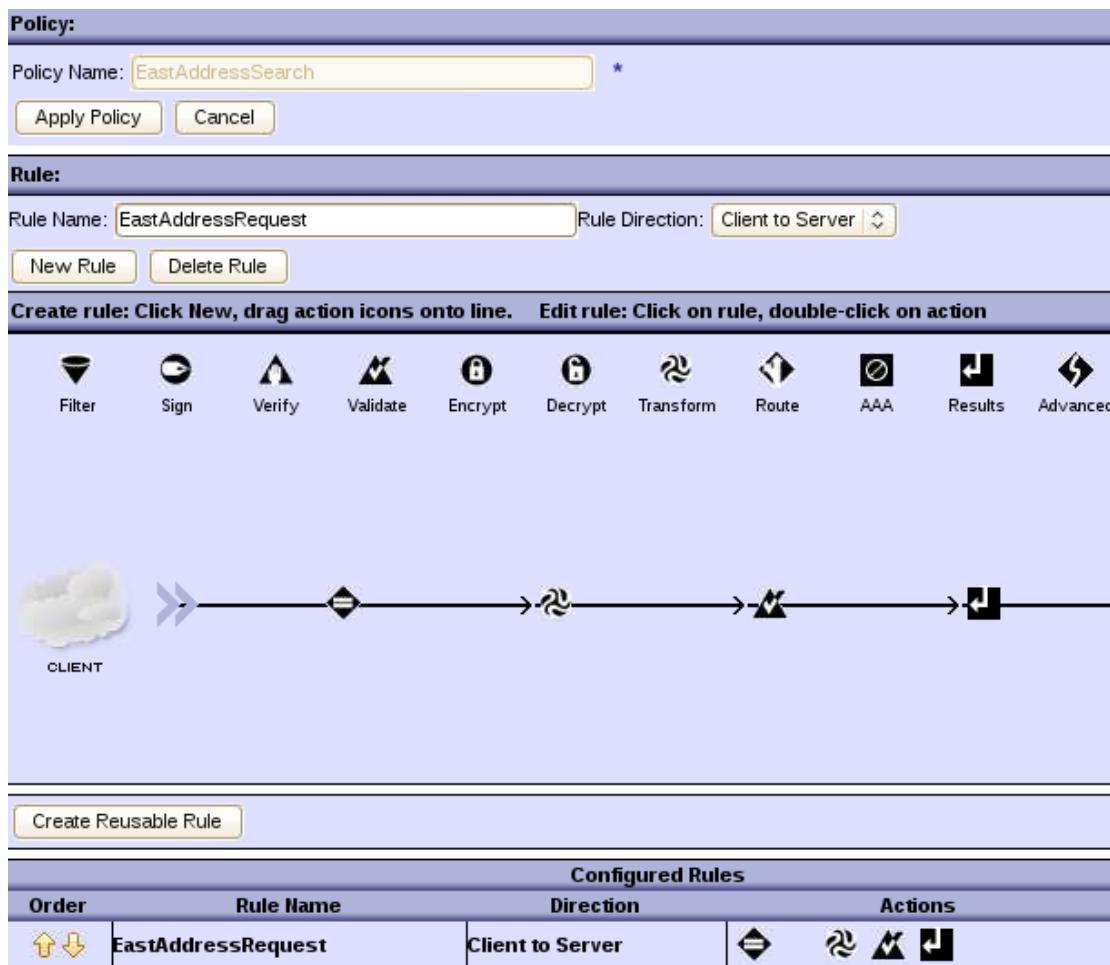
- ___ b. Double-click the **Validate** action to open the **Validate** action configuration window.
- ___ c. Verify that **Validate Document via a WSDL URL** is selected.
- ___ d. For the Transform File, select `EastAddressSearch.wsdl` in the local: directory. You uploaded this WSDL in Exercise 1.
- ___ 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)**. If required, set the parameters 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.



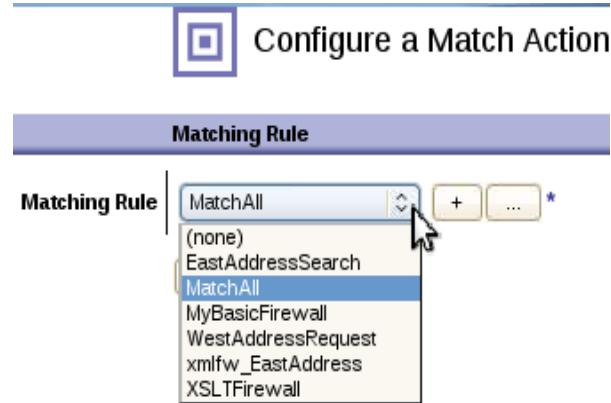
- ___ 13. Create a second document processing rule named EastAddressResponse. The response rule returns the message from the back side server through the DataPower appliance back to the client.
- ___ a. In the processing rule editor for AddressSearchMPGWPolicy, click **New Rule** and rename the rule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



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



- __ c. Double-click the **Match** action icon to configure it.
__ d. From the **Matching Rule** list, select **MatchAll**.



- __ 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.

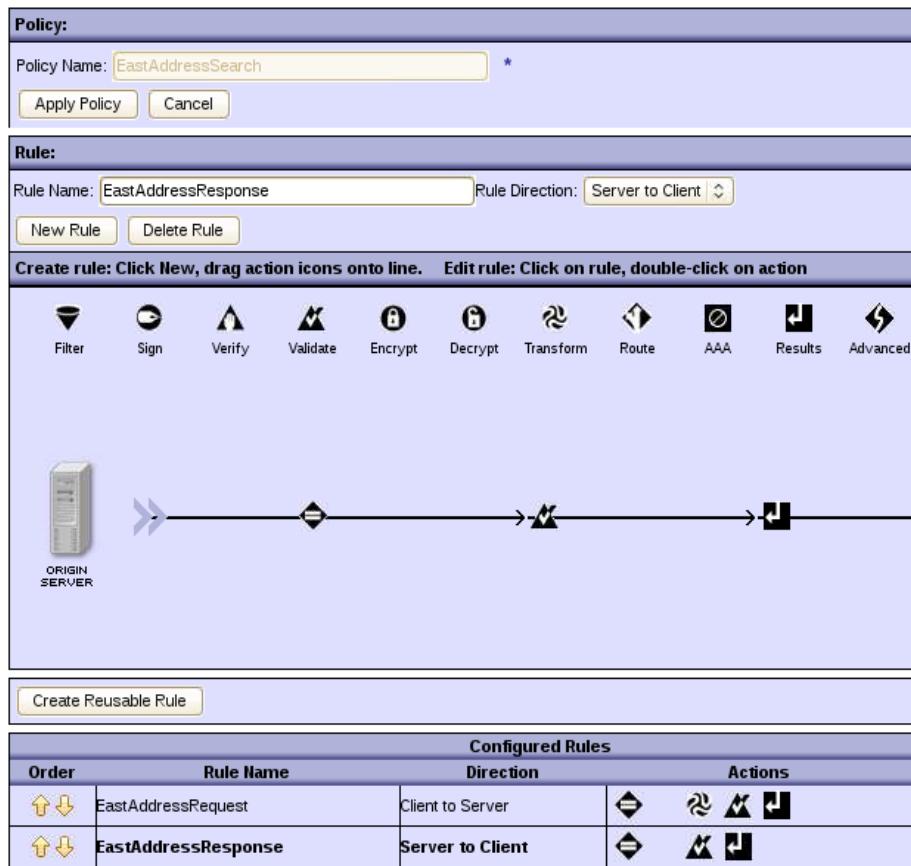


- __ b. Double-click the **Validate** action to open the **Validate** action configuration window.
__ c. Verify that **Validate Document via a WSDL URL** is selected.
__ d. From **WSDL URL**, select `EastAddressSearch.wsdl`.

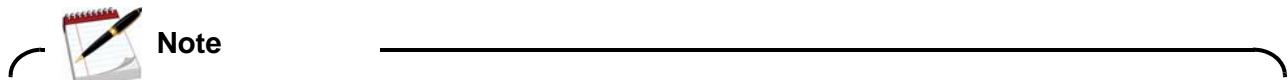
Schema Validation Method	<input type="radio"/> Validate Document via Attribute Rewrite Rule <input type="radio"/> Validate Document via JSON Schema URL <input type="radio"/> Validate Document via Schema Attribute <input type="radio"/> Validate Document via Schema URL <input checked="" type="radio"/> Validate Document via WSDL URL <input type="radio"/> Validate Document with Encrypted Sections
WSDL URL	local:/// EastAddressSearch.wsdl <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/>
Asynchronous	(none) EastAddressSearch.wsdl

- __ 15. Click **Done**.

- ___ 16. Add a **Results** action to the `EastAddressResponse` document processing rule to forward the results to the back-end service.
- In the processing rule editor for `EastAddressResponse`, 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.
 - Confirm that the new document processing rule is in the list of configured rules.



- ___ 17. Click **Close Window** to close the `EastAddressSearch` document policy editor.



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

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

- ___ b. Set the back-end URL to:
`http://<backend_server_ip>: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

Backend URL
<input type="text" value="http://9.135.135.135:9080/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. Two common binary data encoding schemes are used in the industry:

- **Multipurpose Internet Mail Extensions (MIME)** 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 run time, support this standard.
- **Direct Internet Message Encapsulation (DIME)** 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 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.

Response Type
<input type="radio"/> JSON
<input type="radio"/> Non-XML
<input type="radio"/> Pass through
<input checked="" type="radio"/> SOAP
<input type="radio"/> XML
Back attachment processing format
<input checked="" type="radio"/> Dynamic
<input type="radio"/> MIME
<input type="radio"/> DIME
<input type="radio"/> Detect
Back Side Timeout
<input type="text" value="120"/>
Stream Output to Back
<input checked="" type="radio"/> Buffer Messages
<input type="radio"/> Stream Messages
HTTP Version to Server
<input type="radio"/> HTTP 1.0
<input checked="" type="radio"/> HTTP 1.1
Propagate URI
<input type="radio"/> on <input checked="" type="radio"/> off
Compression
<input type="radio"/> on <input checked="" type="radio"/> off

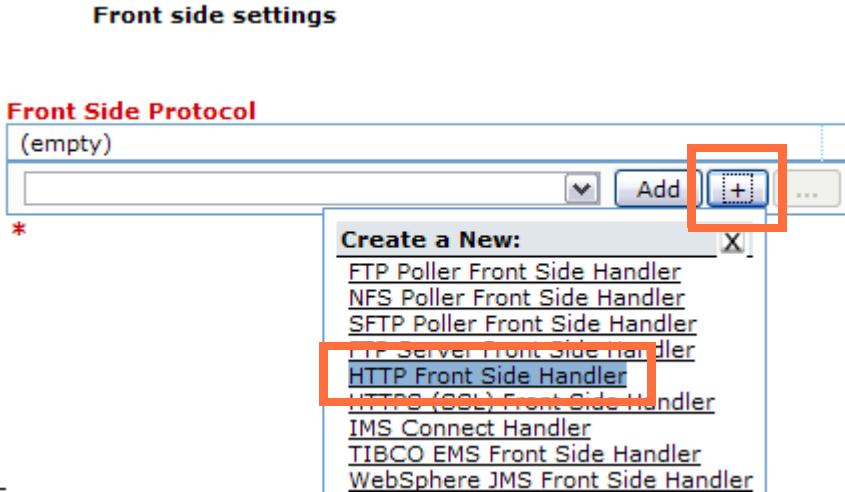
Section 3: Configure an HTTP front side protocol handler

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

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

**Note**

The local IP address value of <dp_public_ip> indicates that the front side protocol handler can receive messages on the Ethernet interface on the appliance that is defined for access from the public. You can specify a **Host Alias** with this name so a specific IP address is not indicated in the handler, which makes the service more portable across deployed appliances.

- ___ 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](#)

[Apply](#) [Cancel](#) [Delete](#)

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**.

[Apply](#) [Cancel](#) [Delete](#)

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**.

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

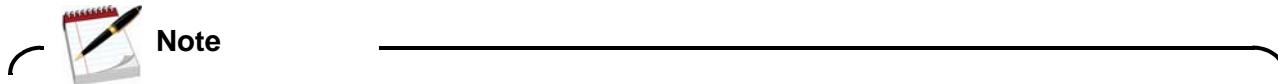
- ___ e. The Multi-Protocol Gateway `EastAddressSearch` is now ready for testing.

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.`

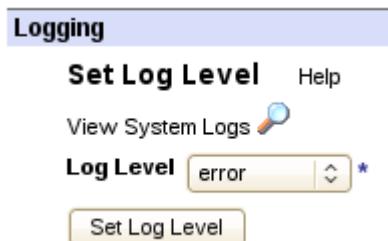
Change the value of `<state>` to NC. You send this message to the East Address MPGW again in the next section.

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 the log window 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/AddressSearch
```

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 pane, 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 that are 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-end 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	xmImgr(default): Attempting TCP connect to WSserver
12:27:03	multistep	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	multistep	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	xmImgr(default): wsdl Compilation Request: Found in cache (local:///EastAddressSearch.wsdl)
12:27:03	ws-proxy	debug	2264719		 0x80a002aa	xmImgr(default): wsdl Compilation Request: Checking cache for URL local:///EastAddressSearch.wsdl
12:27:03	multistep	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	multistep	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	xmImgr(default): xslt Compilation Request: Found in cache (local:///Address-EastRenameNamespace.xsl)
12:27:03	xslt	debug	2264719		 0x80a002aa	xmImgr(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		0x80e000b4	mpgw (EastAddressSearch): rule (EastAddressResponse): selected via match 'MatchAll' from processing policy 'EastAddressSearch'
12:37:31	mpgw	debug	2266271			0x81000171	Matching (MatchAll): Match: Received URL [/EastAddress/services/AddressSearch] matches rule '*'
12:37:31	mpgw	debug	2266271			0x80e0012a	mpgw (EastAddressSearch): Selecting Backside Processing Rule Based on URL:/EastAddress/services/AddressSearch
12:37:31	mpgw	info	2266271			0x80e0015b	mpgw (EastAddressSearch): HTTP response code 500 for 'http://WSserver:9080/EastAddress/services/AddressSearch'
12:37:31	mpgw	info	2266271			0x80e0012d	mpgw (EastAddressSearch): Using Backside Server: http://WSserver:9080/EastAddress/services/AddressSearch
12:37:31	memory-report	debug	2266271			0x80e0068e	mpgw (EastAddressSearch): Request Finished: memory used 187568
12:37:31	mpgw	debug	2266271			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			0x80e00536	mpgw (EastAddressSearch): HTTP Header-Retention:Compression Policy: Off, URL: /EastAddress/services/AddressSearch
12:37:31	mpgw	debug	2266271			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			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			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			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			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. Because it is a successful message, you see both a request and a response entry.

Refresh	Flush	Disable Probe	Export Capture	View Log	Send Message	Close
view trans#	type	inbound-url		outbound-url		rule
<input checked="" type="checkbox"/>	1238352 request	http:// 9.12.15.15:6887 /EastAddress /services/AddressSearch		http:// 9.12.15.15:9080 /EastAddress /services/AddressSearch		EastAddressSearch_r
	1238352 response	http:// 9.12.15.15:6887 /EastAddress /services/AddressSearch		http:// 9.12.15.15:9080 /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. The rule ran successfully, hence, 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.



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.
- Again, edit `findByLocation.xml` to make the XML invalid (for example, change an element name), so that the schema validation fails.
 - Run the cURL command. You get an Internal Error response.
 - In the Transaction List, click **Refresh**.
 - A new entry is now shown. It is a single message and is displayed in red to indicate an error.

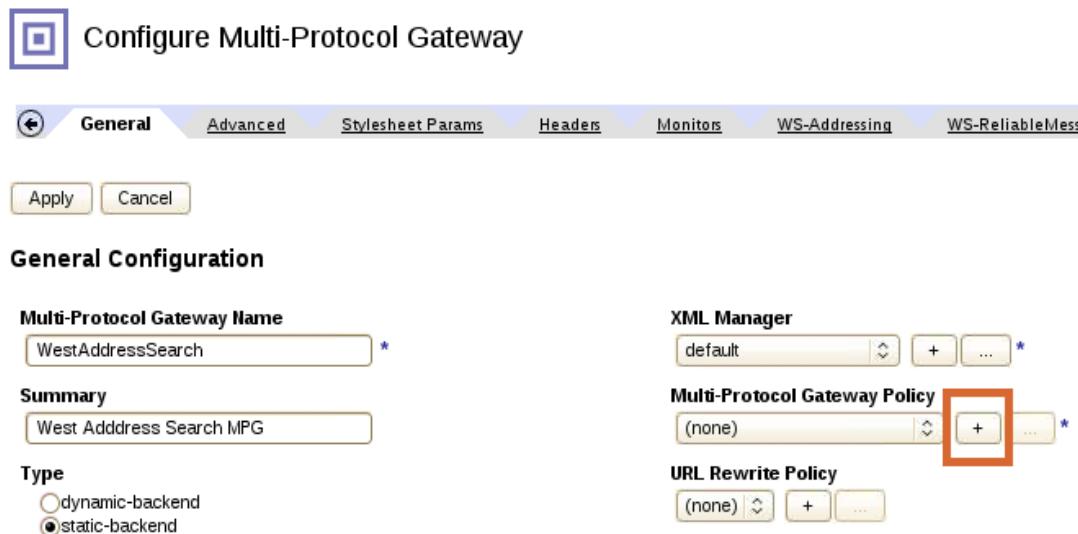
view	trans#	type	inbound-url	outbound-url	rule
[+]	1238352	request	http://9.10.10.17:6887/EastAddress/services/AddressSearch	http://9.10.10.15:9080/EastAddress/services/AddressSearch	EastAddressSearch
[+]	1241104	request	http://9.10.10.17:6887/EastAddress/services/AddressSearch	http://9.10.10.15:9080/EastAddress/services/AddressSearch	EastAddressSearch

- Click the magnifying glass next to the new request.
 - In the probe rule window, notice that the list of actions is shorter. It displays the actions that were processed.
 - Click the magnifying glass, which is the result of the validation.
 - 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.
 - 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.
 - 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.

3.2. Create a multi-protocol gateway for WestAddressSearch

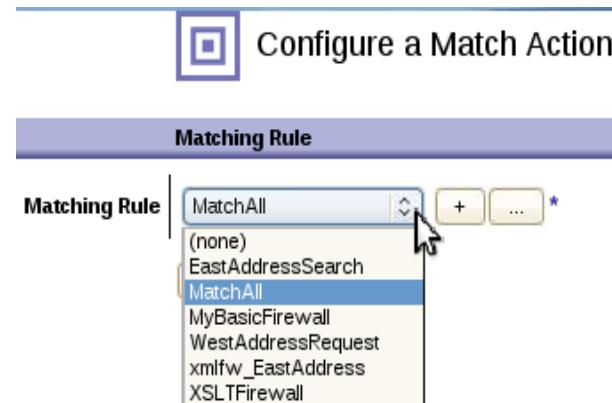
In this step, you create an MPGW based on a WSDL for the West Address Search web service. The steps are the same as the previous section, except that you use the `WestAddressSearch.wsdl` file.

- 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. From **Matching Rule**, select **MatchAll**.



- __ 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.

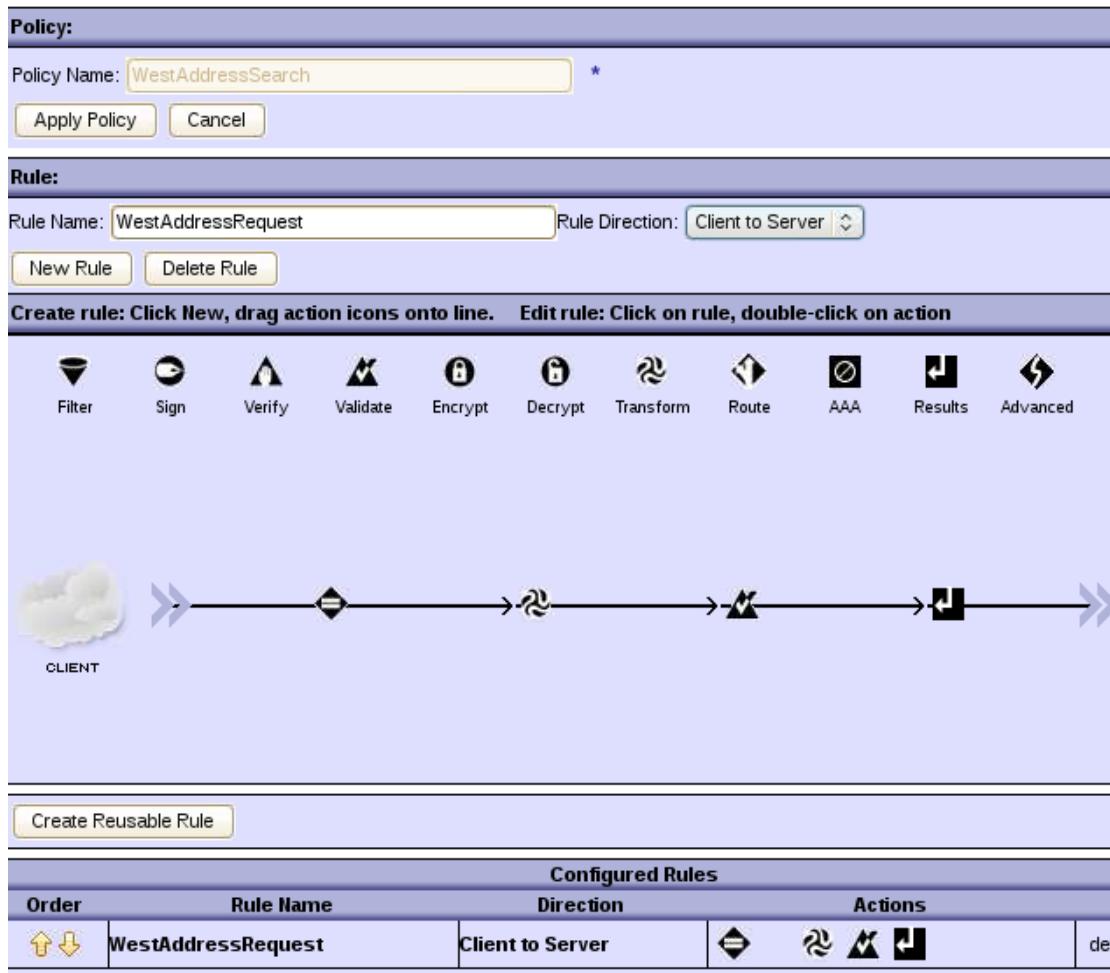


- __ b. Double-click the **Transform** action to open the **Transform** action configuration window.
- __ c. Verify that **Transform with XSLT style sheet** is selected.
- __ d. Click **Upload** to upload a style sheet that is called `<lab_files>/AdvMPGW/Address-WestRenameNamespace.xsl` that copies the 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.

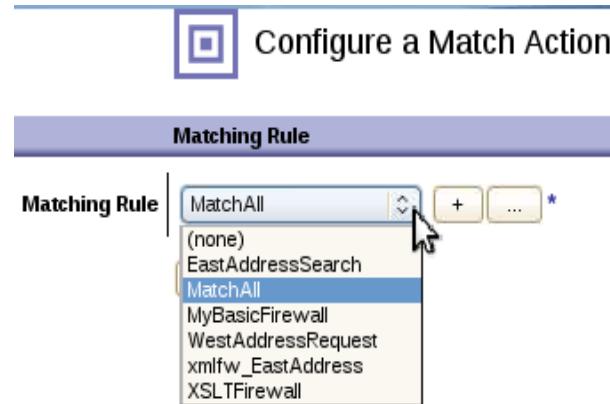


- __ b. Double-click the **Validate** action to open the **Validate** action configuration window.
- __ c. Verify that **Validate Document via a WSDL URL** is selected.
- __ d. For the Transform File, select `WestAddressSearch.wsdl` in the local: directory. You uploaded this WSDL in Exercise 1.

- ___ e. Click **Continue**.
 - ___ f. After uploading the wsdl, make sure that the Validate configuration window contains the required 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.

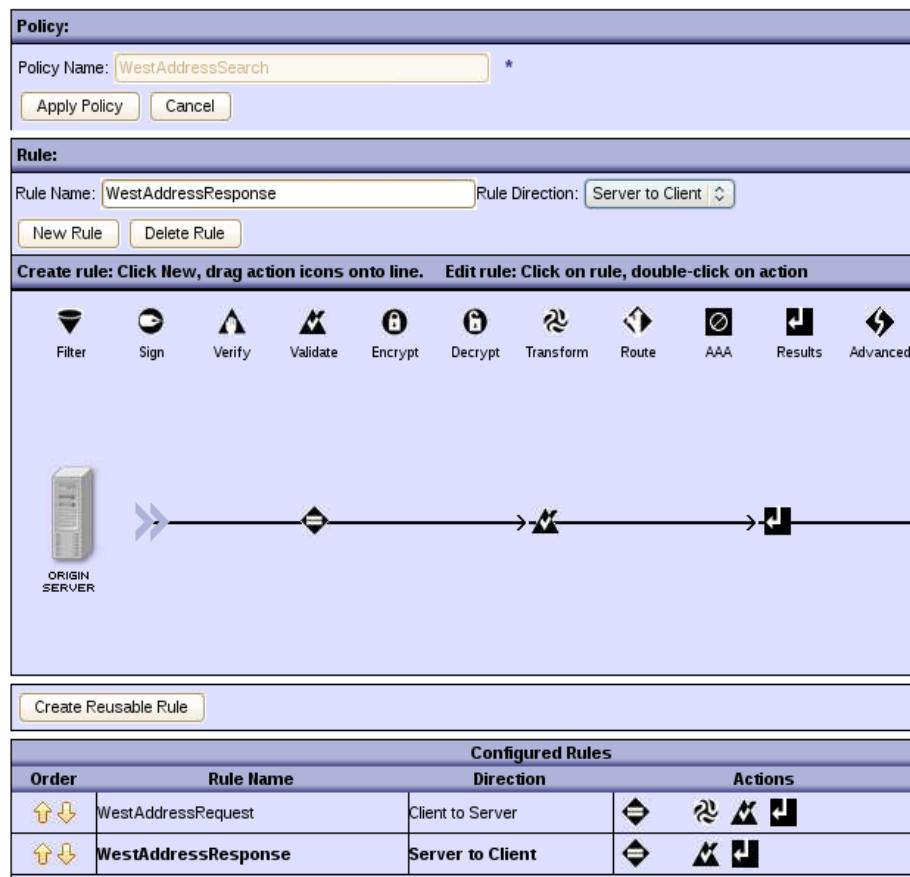


- ___ 13. Create a second document processing rule named `WestAddressResponse`. The response rule returns the message from the back side server through the DataPower appliance back to the client.
- ___ a. In the processing rule editor for `AddressSearchMPGWPolicy`, click **New Rule** and rename the rule to: `WestAddressResponse`
 - ___ b. Set the processing rule direction to **Server to Client**.
 - ___ c. Double-click the **Match** action icon to configure it.
 - ___ d. From **Matching Rule**, select **MatchAll**.



- ___ 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.
- A diagram illustrating a sequence of actions. It starts with a diamond icon, followed by a horizontal line, and then a square icon containing a mountain-like symbol, representing the Validate 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 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://<backend_server_ip>:9080/WestAddress/services/AddressSearch`
- ___ 19. Change the **Propagate URI** setting to **off**.

3.3. 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 `findByLocation.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.

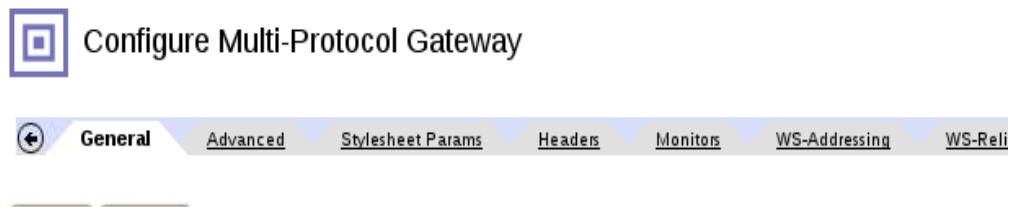
```
gedit findByLocation.xml &
```
 - ___ b. Change the state value of NC to: CA
 - ___ c. Save the file.
- ___ 3. Run the following command:

```
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.

3.4. 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 an MPGW service, click **Add**.
- c. Under General Configuration, enter:
 - **MPGW Name:** AddressRouter
 - **Summary:** Content based routing that uses an MPGW
 - **Type:** dynamic-backend



Multi-Protocol Gateway Name <input type="text" value="AddressRouter"/> *	XML Manager <input type="text" value="default"/>
Summary <input type="text" value="Content based routing using an MPGW"/>	Multi-Protocol Gateway Policy <input type="text" value="(none)"/>
Type <input checked="" type="radio"/> dynamic-backend <input type="radio"/> static-backend	URL Rewrite Policy <input type="text" value="(none)"/>



Note

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.

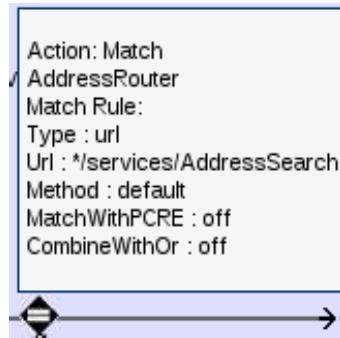
- 2. Create an HTTP front side protocol handler that is named `AddressRouter` as the device port from which this service accepts client requests.
 - a. In the Front Side Protocol section, click **+** (new).
 - b. Select **HTTP Front Side Handler** from the list

- ___ c. Configure the new HTTP front side handler with the following values. Leave all other settings to the default values.
- **Name:** 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.
- ___ a. Click **New** (the [+] button) in the **Multi-protocol Gateway Policy** field to create an MPGW policy.
- ___ b. For **Policy Name**, enter: AddressRouter
- ___ c. Click **Apply Policy**.
- ___ d. Click **New Rule** to create a rule.
- ___ e. Create a request rule by selecting the **Rule Direction** list and selecting **Client to Server**.
- ___ f. In the rule configuration area, a **Match** action is added. Double-click the **Match** action to open the configuration window.
- ___ g. 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 the following figure:



- ___ 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.

```

graph LR
    Diamond(( )) --> Route[Route]

```

 - ___ b. Double-click the **Route action** to configure content based routing.
 - ___ c. The Route action configuration pane 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 pane, which helps create the XPath expression.

XPath Expression	<input type="text"/>	XPath Tool *
Remote Host	<input type="text"/> *	
Remote Port	<input type="text"/> *	
SSL	<input type="radio"/> on <input checked="" type="radio"/> off *	
<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 then click **Continue**.
- ___ i. In **Namespace Handling**, select the **local** option.

If the sample XML document does not show at the bottom of the pane, 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. If the element still contains “CA” from earlier, change it back to “NC”.

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 `<dp_public_ip>` 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="6557"/> *	
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	
<code>local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'NC']</code>	dp_public_ip	6557	off	

- __ o. Use the steps a to n 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.

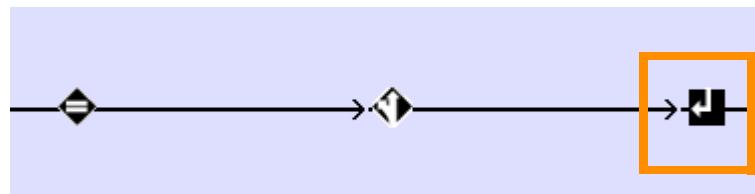
Rules

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

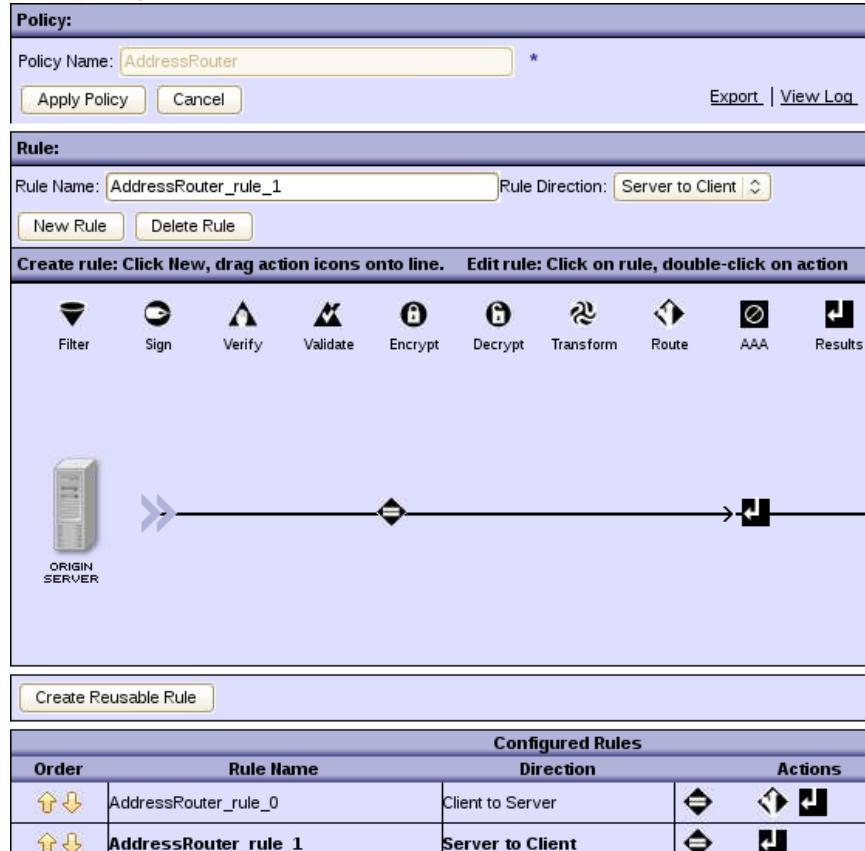
- ___ r. Click **Done** on the “Configure Route Action” pane to complete the configuration of the content based routing function.
- ___ s. Back on the policy editor page, drag a **Results** action to the processing rule, after the **Route** action, to move the request from the INPUT context to the output 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 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. From **Matching Rule**, select **MatchAll**.
 - __ 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. In the Configure MPGW page, click **Apply** to save the AddressRouter MPGW configuration.
- __ j. Click **Save Config** to save your configuration to the appliance.

3.5. 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/
AddressSearch
```

- 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 4. 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 an error policy
- Configure a service policy with an On Error action
- Configure a service policy with an Error rule
- Configure a default service policy at the multi-protocol gateway level

Introduction

The processing within a policy is expected to occasionally have errors. Policies can use **On Error** actions and **error rules** to deal with errors. Also, one can use an error policy to deal with errors. This exercise creates an error policy, and adds an error rule and an **On Error** action to provide documentation on a failure to the multi-protocol gateway policy.

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 3: 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

4.1. Adding error processing

For this part, you add default error processing to the multi-protocol gateway by creating an error policy. The error policy takes control when an error occurs that conforms to the error match condition you create. Error policy is where the default error processing is configured allowing more granular error processing to occur at the error rule and On Error action levels.

Next, 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.

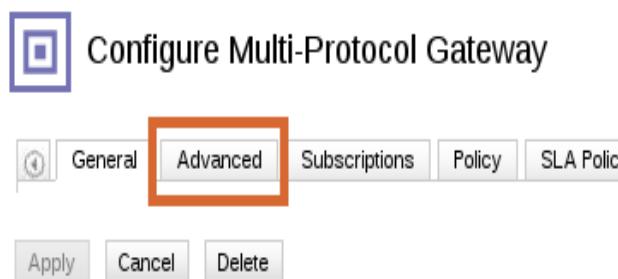
Finally, you add an **On Error** action to the existing request rule. **On Error** actions specify the processing rule that is called during any errors in subsequent actions. The request rule to specify different error handling, processing rules at different steps within the request rule, is allowed.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

This part is divided into the following sections:

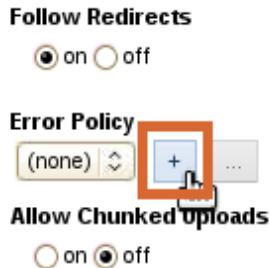
- Create an error policy
- Test the error policy
- Create the error rule and add it to the policy
- Test the error rule
- Add an **On Error** action to the policy
- Test the **On Error** action
- Add an error rule and **On Error** action
- Test the new rule and action

Part 1: Add an Error Policy

- 1. On the **EastAddressSearch** multi-protocol gateway, click the **Advanced** tab.



- __ 2. Scroll down, and add an Error Policy by clicking the (+) key.



- __ 3. Name the new error policy name: EastAddressSearch_ErrorPolicy

- __ 4. Click **Add** to create a Policy Map.

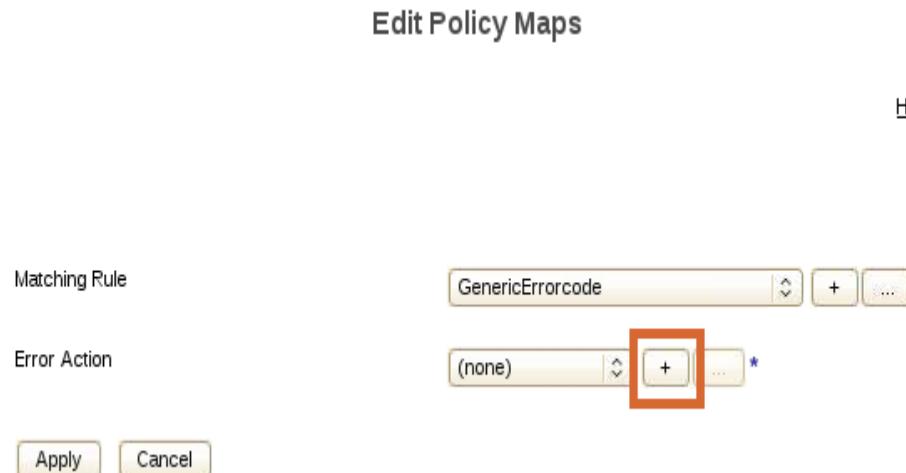
Matching Rule	Error Action
(empty)	Add

- __ c. Click **Add** (the [+]) to create a **Matching Rule**.
- __ d. In the Edit Policy Maps window, click **New** (the [+]) to create a matching rule.
- __ e. In the **Main** tab of the Configure Matching Rule page, enter a name of: GenericErrorcode
- __ f. Select the **Matching Rule** tab. Select **Add** to add a property.
- __ g. 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.

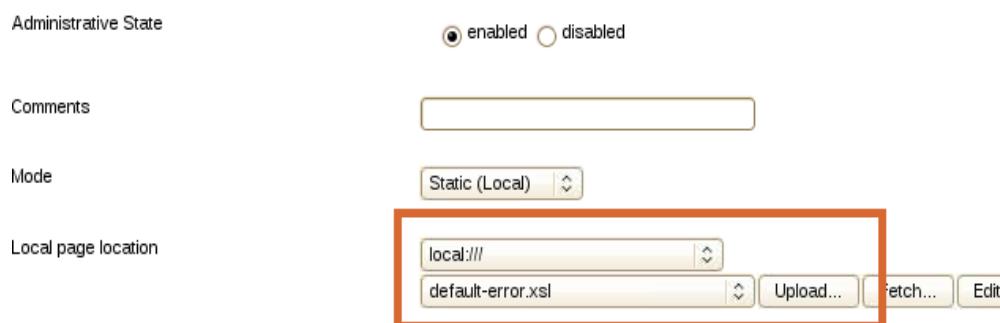
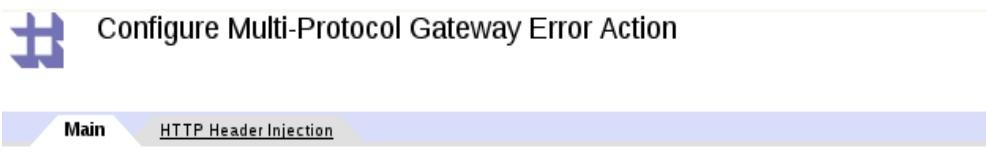
Matching Type	Error Code <input type="text"/> *
	Help Select Code *
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>	

- __ h. Click **Apply**, and then click **Apply** again.

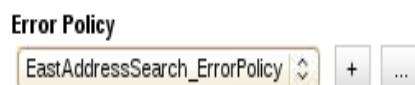
- ___ i. Click **Add** (the [+]) to create an **Error Action**.



- ___ j. In the **Main** tab of the Configure Multi-Protocol Gateway Error Action page, enter:
ErrorPolicyAction
- ___ k. In the **local page location** section, click **Upload**, **Browse**, and go to the file:
<LAB_FILES>/errors/default-error.xsl
- ___ l. Click **Open**, **Upload**, **Continue**, and then click **Done**.



- ___ m. Click **Apply**, **Apply**, **Apply**, and then click **Apply** again to back out of all the configuration windows and save the Error Policy in the multi-protocol gateway.



Part 2: Test the default error policy

- ___ 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_east_port>/EastAddress/services/AddressSearch
```
- ___ 4. You receive the `<Address>` elements as expected.
- ___ 5. Now create an error by editing `findByLocation.xml` and change the message to make it fail schema validation. Change the element `<state>` to `<stateBad>` and be sure to change both the start and end tags.
- ___ 6. 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
```
- ___ 7. Review the default error message that is returned. The error message is the result of the error policy configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:dp="http://www.datapower.com/extensions"
xmlns:dpconfig="http://www.datapower.com/param/config"
extension-element-prefixes="dp"
exclude-result-prefixes="dp dpconfig">
<xsl:output method="html" version="1.0" />
<xsl:template match="/">
  <html>
    <head>
      <title>My Company Application</title>
    </head>
    <body>
      <h2>== Default Error Processing. == </h2>
      <p>
        This page is generated by default-error.xsl,
        which is referenced by the MPGW Error Policy specification in
        EastAddressSearch.
      </p>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

**Important**

Notice the value of the second header (<h2>) which states the == Default Error Processing ==. This distinction in error report processing is important to note. As you configure error rules and an On Error action, you use other style sheets to report the error message. Because the Error Policy is the default error processing, use of the On Error action and Error Rule overrides the Error Policy. Therefore, you can confirm other error processing taking precedence by a different error return message.

- ___ 8. Edit the file `findByLocation.xml` and change the element `<stateBad>` to `<state>` and be sure to change both the start and end tags.

- ___ 9. 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
```

- ___ 10. Confirm the `<Address>` elements as returned as expected.

Part 3: 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 **Multi-Protocol Gateway**.
 - __ b. Click **EastAddressSearch** in the list.
 - __ c. Edit the EastAddressSearch multi-protocol gateway policy by clicking **Edit** (the [...]). The current configuration of the policy is displayed.
 - __ 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 **Match** icon.
 - __ h. In the Configure a Match Action window, select the Matching Rule that you created in an earlier step: `GenericErrorcode`.
 - __ i. Click **Done**.
 - __ j. In 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.

- __ k. Double-click the **Transform** action.
- __ l. Select **Transform with XSLT style sheet**.
- __ m. Click **Upload, Browse**, and go to the file: `<LAB_FILES>/errors/custom-error.xsl`
- __ n. Click **Open, Upload, Continue**, and then click **Done**.
- __ o. 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			
EastAddressSearch_request	Client to Server				
EastAddressSearch_response	Server to Client				
MyEastAddressSearch_ErrorRule	Error				

- __ p. Click **Close Window**.
- __ q. Click **Apply** to save the changes to the multi-protocol gateway.

Part 4: 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.



The screenshot shows the WebSphere DataPower XI52 interface. At the top, there's a toolbar with buttons for Refresh, Flush, Disable Probe, Export Capture, View Log, Send Message, and Close. Below the toolbar is a table with the following data:

view	trans#	type	inbound-url	outbound-url	rule	client-
E	2464303	request	http://172.16.78.44:6957/EastAddress /services/AddressSearch	http://WServer:9080/EastAddress /services/AddressSearch	EastAddressRequest	172.16
E	2464303	error	http://172.16.78.44:6957/EastAddress /services/AddressSearch	http://WServer: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 5: 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.

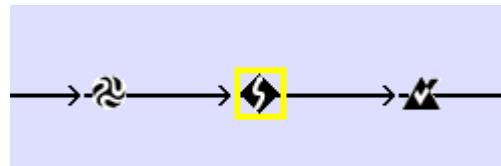
- ___ 1. Go back to the `EastAddressSearch` policy editor.

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 'Rule' configuration page. At the top, there's a header with 'Rule:' and fields for 'Rule Name' (set to 'EastAddressRequest') and 'Rule Direction' (set to 'Client to Server'). Below the header are buttons for 'New Rule' and 'Delete Rule'. A status bar at the bottom says 'Create rule: Click New, drag action icons onto line. Edit rule: Click on rule, double-click on action'. The main area contains a flowchart with nodes: 'CLIENT' (cloud icon) connected to a diamond node, which is connected to a 'Transform' node (represented by a swirl icon), which is connected to a 'Validate' node (represented by a triangle icon), which is connected to an 'ORIGIN SERVER' (server rack icon). Above the flowchart is a row of action icons: Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, Advanced, and a trash can icon. Below the flowchart is a button labeled 'Create Reusable Rule'. At the bottom is a table titled 'Configured Rules' with columns for 'Order', 'Rule Name', 'Direction', and 'Actions'.

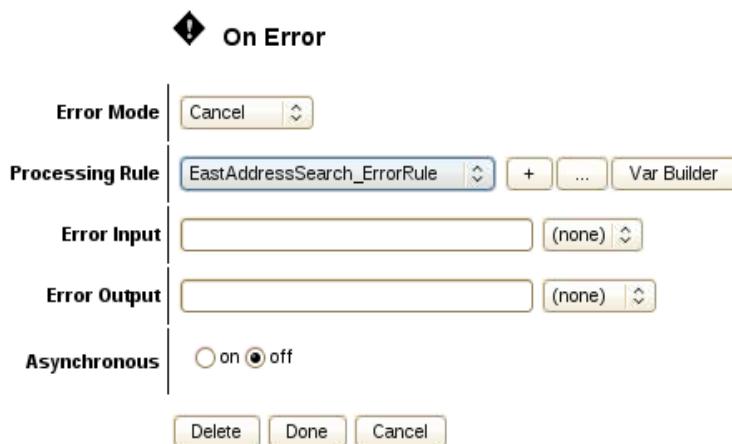
Configured Rules			
Order	Rule Name	Direction	Actions
	EastAddressRequest	Client to Server	delete rule
	EastAddressResponse	Server to Client	delete rule
	EastAddressSearch_ErrorRule	Error	delete rule

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 6: 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/AddressSearch
```

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 receives the 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.

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 7: 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	
1	EastAddressRequest	Client to Server		
2	EastAddressResponse	Server to Client		
3	EastAddressSearch_ErrorRule	Error		
4	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

Error Mode	<input type="button" value="Cancel"/>
Processing Rule	<input type="button" value="EastAddressSearch_filter_ErrorRule"/> <input type="button" value="+"/> <input type="button" value="..."/> <input type="button" value="Var Builder"/>
Error Input	<input type="text"/> (none)
Error Output	<input type="text"/> (none)
Asynchronous	<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 8: 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 5. Creating cryptographic objects and configuring SSL

What this exercise is about

This exercise shows you how to create cryptographic objects in DataPower, and how to use them to configure SSL connections. Asymmetric keys can be created on the appliance, asymmetric and symmetric keys can be uploaded externally. You create the cryptographic objects that you need to support an SSL connection: crypto key, crypto certificate, crypto identification credentials, and crypto validation credentials. These objects are used as part of a crypto profile, which defines one end of an SSL connection. Existing multi-protocol gateways (MPGWs) are modified to support an SSL connection between them.

What you should be able to do

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

- Generate crypto keys by using the WebSphere DataPower cryptographic tools
- Create a crypto identification credential by using a crypto key object and a crypto certificate object
- Validate certificates by using a validation credential object
- 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 end goal of this exercise is to provide an SSL connection between the AddressRouter MPGW and the EastAddressSearch MPGW. To define the SSL endpoints in DataPower, several objects need to be configured. In the first half of the exercise, you create the crypto objects that you need for the SSL connection:

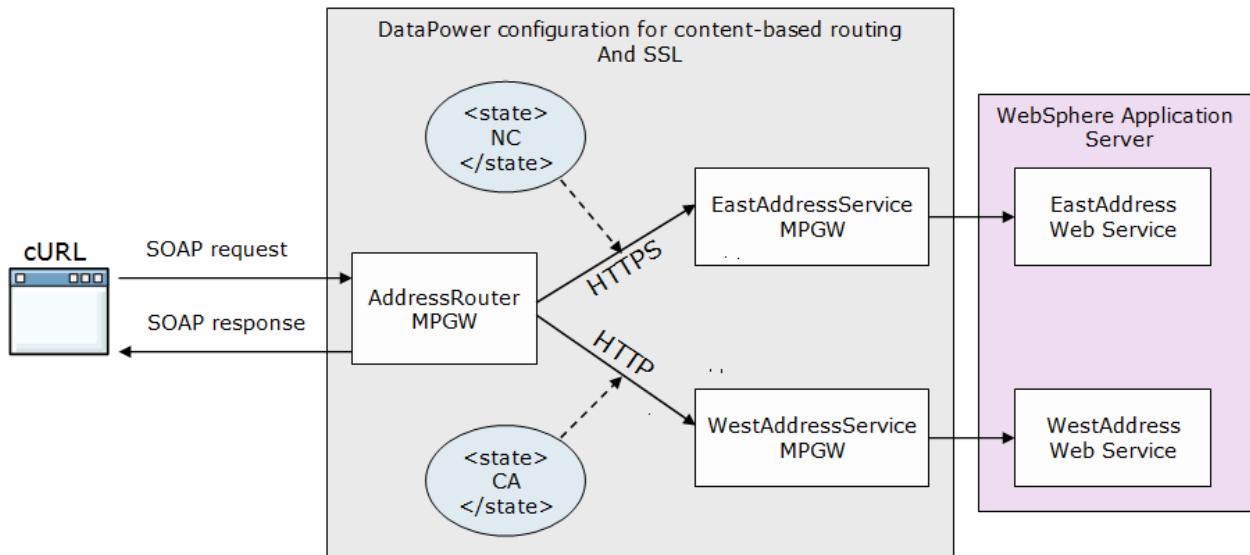
- Generate the asymmetric key and certificate, and create the related crypto objects
- Create an identification credential object and a validation credential object
- Create the SSL crypto profiles that specify the characteristics of the SSL endpoints.

In the second half of the exercise, you modify two of the MPGWs you created in an earlier exercise to use SSL communications. You modify the EastAddressSearch MPGW to support an HTTPS front end, which acts as an SSL server. Next, you modify the AddressRouter MPGW to be the SSL client to the HTTPS handler of the EastAddressSearch MPGW.

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 3: Creating an advanced multi-protocol gateway” for the MPGWs used in this exercise
- Remember to use the domain and port address that you were assigned in the exercise setup. **Do not** use the default domain.
- The references in exercise instructions refer to the following values:
 - `<lab_files>`: location of the student lab files (default is `/home/localuser/dplabs/dev`)
 - `<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 (default is 9090)
 - `<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_content_based_routing_port>`: $6nn9$, port number for the service that provides the routing function

5.1. 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 that is 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. Switch to your own student domain.
- 3. Generate a certificate-key pair on the DataPower appliance.
 - a. In the DataPower WebGUI vertical navigation bar, click **Administration > Miscellaneous > Crypto tools**.



Note

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

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

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

**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.

__ 4. 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. In the Keys and Certificates Management page, in the **Basics** section, click the **Keys** link.

__ d. Verify that you see the `StudentKeyObj` referencing the generated private key file.

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

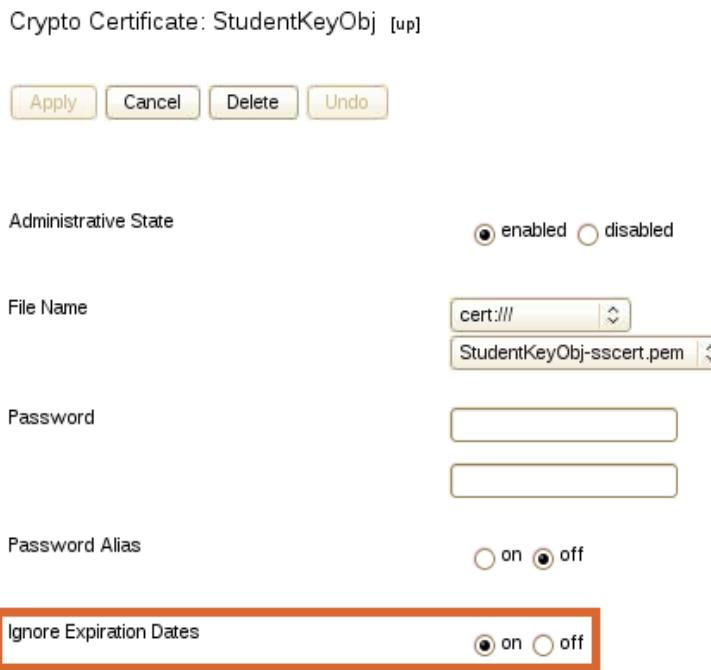
__ e. Click the web browser back button, or **Control Panel > Keys and Certs Management** again.

__ 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**.

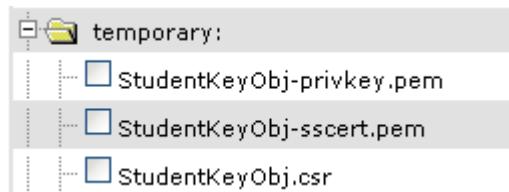


- ___ j. Click **Apply**.


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.

- ___ 5. 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. The StudentKeyObj.csr is the certificate signing request.

5.2. 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 were 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 an SSL endpoint.

- 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

Name

StudentKeyValCred *

Administrative State

enabled disabled

Certificates

StudentKeyObj	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
StudentKeyObj	<input type="button" value="Add"/>	<input type="button" value="+"/>
<input type="button" value="..."/>		

Certificate Validation Mode

Match exact certificate or immediate issuer

Use CRL

on 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

Name <input type="text" value="StudentServerCP"/>	Administrative State <input checked="" type="radio"/> enabled <input type="radio"/> disabled
Identification Credentials <input type="text" value="StudentIdCred"/>	
Validation Credentials <input type="text" value="(none)"/>	
Ciphers <input type="text" value="HIGH:MEDIUM:aNULL:eNULL:@STRENGT"/>	
Options <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <input checked="" type="checkbox"/> Enable default settings <input type="checkbox"/> Disable SSL version 2 <input type="checkbox"/> Disable SSL version 3 <input type="checkbox"/> Disable TLS version 1.0 <input type="checkbox"/> Permit insecure SSL renegotiation to a legacy SSL client <input type="checkbox"/> Enable compression <input type="checkbox"/> Disable TLS version 1.1 <input type="checkbox"/> Disable TLS version 1.2 </div>	
Send Client CA List <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. From **Control Panel > Keys and Certs Management**, click the **Crypto Profile** link.
 - ___ b. On the Configure Crypto Profile page, click **Add** to create another crypto profile.
 - ___ c. 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

Name	<input type="text" value="StudentClientCP"/> *
Administrative State <input checked="" type="radio"/> enabled <input type="radio"/> disabled	
Identification Credentials <div style="border: 1px solid #ccc; padding: 2px; width: fit-content; margin-left: auto; margin-right: 0;"> (none) + ... </div>	
Validation Credentials <div style="border: 1px solid #ccc; padding: 2px; width: fit-content; margin-left: auto; margin-right: 0;"> StudentKeyValCred - ... </div>	
Ciphers <div style="border: 1px solid #ccc; padding: 2px; width: fit-content; margin-left: auto; margin-right: 0;"> HIGH:MEDIUM::aNULL::eNULL:@STRENGTH </div>	
Options <div style="border: 1px solid #ccc; padding: 2px; width: fit-content; margin-left: auto; margin-right: 0;"> <input checked="" type="checkbox"/> Enable default settings <input type="checkbox"/> Disable SSL version 2 <input type="checkbox"/> Disable SSL version 3 </div>	

- ___ d. Click **Apply**.

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

- ___ e. Click **Save Config**.

5.3. 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 changes to `https` and a `-k` argument is 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 the service does not yet have an SSL-aware handler on the receiving end. This problem is expected at this point in 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/  
AddressSearch
```

The same list of NC addresses is returned.

5.4. 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. Open the **EastAddressSearch** multi-protocol gateway configuration page .
- ___ 2. In the Front Side Protocol section of the page, create a handler that uses HTTPS.
 - ___ a. Click **+** to open the selection list of handler types.
 - ___ b. Click **HTTPS (SSL) Front Side Handler**.
 - ___ c. The handler configuration dialog box opens. Enter: `EastAddressSearch_SSL`.
 - ___ d. For the Local IP Address, use `<dp_public_ip>`, or its alias.
 - ___ e. Use the Port Number `<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: `EastAddressSSLProfile`
 - ___ h. For the Reverse (Server) Crypto Profile, select **StudentServerCP**, which you created in an earlier section.

- ___ i. Leave the other settings at their defaults, which do not require client authentication. Click **Apply**.

SSL Proxy Profile

Name	EastAddressSSLProfile
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
SSL Direction	Reverse *
Reverse (Server) Crypto Profile	StudentServerCP
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. In the HTTPS handler configuration page, the SSL Proxy field now contains the new proxy profile object. Click **Apply**.

- ___ 3. The new HTTPS handler now shows as one of the gateway front-end handlers.

Front Side Protocol
EastAddressSearch (HTTP Front Side Handler)
EastAddressSearch_SSL
EastAddressSearch_SSL

- ___ 4. Since this service is not using SSL to communicate with the back-end web service, the SSL Client Crypto Profile setting remains empty.
- ___ 5. Click **Apply** for the service.

5.5. 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-end 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
*   CApth: /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/
AddressSearch
```

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.

5.6. 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>

The screenshot shows a web-based configuration interface for an AddressRouter. At the top, there's a navigation bar with links for Export, View Log, View Status, and Help. Below the navigation bar, a blue header bar has the word 'Rules' in white. The main content area displays a table of XPath expressions. One row in the table has its edit icon highlighted with a red box. The table columns are labeled: XPath Expression, Remote Host, Remote Port, SSL, and Action icons.

	Remote Host	Remote Port	SSL	
Body'/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'NC']	dp_public_ip	6977	off	
Body'/*[local-name()='findByLocation']/*[local-name()='state'][normalize-space(.) = 'CA']	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()='...</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 a previous section.
- ___ 4. Click **Apply** for the service.
- ___ 5. Examine the SSL proxy profiles for the EastAddressSearch service and the AddressRouter service.
- ___ a. Click **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.

5.7. 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 `7971`, which is `<mpgw_east_ssl_port>` for this test. This information 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 generated key and certificate objects on the DataPower appliance. 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.

You used these objects to configure both server-side and client-side SSL on two separate multi-protocol gateways.

Exercise 6. 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.

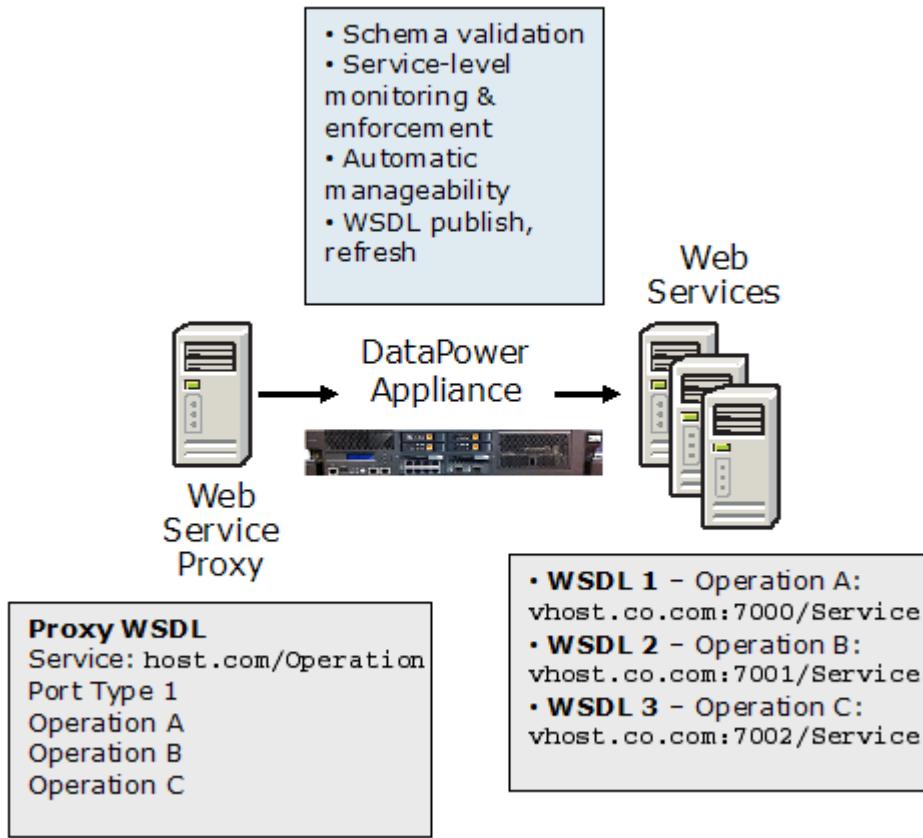
What you should be able to do

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>*: 6nn5, 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). The default is 9080.

6.1. 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 **WSDLs** tab, ensure that the **Add WSDL** option is selected.

WSDLs

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

- __ b. Select `local:///` for the WSDL File URL.
- __ c. Select `EastAddressSearch.wsdl` from the list.

WSDL File URL local:/// EastAddressSearch.wsdl <input type="button" value="Upload..."/> <input type="button" value="Fetch..."/> <input type="button" value="Edit..."/> <input type="button" value="View..."/> <input type="button" value="Browse UDDI"/>
Use WS-Policy References <input checked="" type="radio"/> on <input type="radio"/> off
WS-Policy Parameter Set <input type="button" value="none"/> <input type="button" value="+"/> <input type="button" value="..."/>
WS-Policy Enforcement Mode <input type="button" value="Enforce"/>
SLA Enforcement Mode <input type="button" value="Allow"/>
<input type="button" value="Next"/>

**Note**

This WSDL was loaded in Exercise 1. If it is not in `local:///`, use **Upload** to retrieve the WSDL 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="+"/> ...	/EastAddress/services/AddressSearch

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

- ___ d. Configure the new HTTP front-end handler with the following values. Leave all other settings to the default values.
- **Name:** AddressSearchFSH
 - **Local IP Address:** Host alias of `dp_public_ip`
 - **Port Number:** `<wsp_proxy_port>`

HTTP Front Side Handler

Name	<input type="text" value="AddressSearchFSH"/> *
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Local IP Address	<input type="text" value="dp_public_ip"/> <input type="button" value="Select Alias"/>
Port Number	<input type="text" value="8975"/> *
HTTP Version to Client	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="text" 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	traintest.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

6.2. 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 6.1, "Create a web service proxy for EastAddressSearch web service," on page 6-5** 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.



Note

If necessary, upload the WSDL document for the `WestAddressSearch` web service, which is in `<lab_files>/setup/WestAddressSearch.wsdl`, and reuse the `AddressSearchFSH` endpoint handler.

- 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.**

6.3. 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
<hr/>		
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:

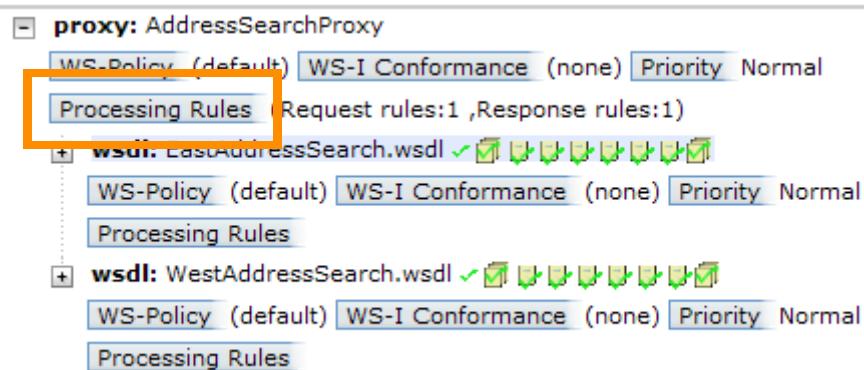
```
<wsdl:service name="AddressSearchService">
<wsdl:port binding="impl:AddressSearchSoapBinding" name="AddressSearch">
<wsdlsoap:address
location="http://training.ibm.com:9080/EastAddress/services/AddressSearch"/>
</wsdl:port>
</wsdl:service>
```

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 URI, 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 have 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.

- **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
- Effective Value** **Local Value**
- Enable this component
- Publish in WSDL
- Schema validate faults messages
- Schema validate request messages
- Schema validate response messages
- Do not schema validate SOAP headers
- Use WS-Addressing
- Use WS-ReliableMessaging
- Accept MTOM / XOP Optimized Messages
- CLOSE
- Done**
- 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**

- __ 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.

6.4. 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.



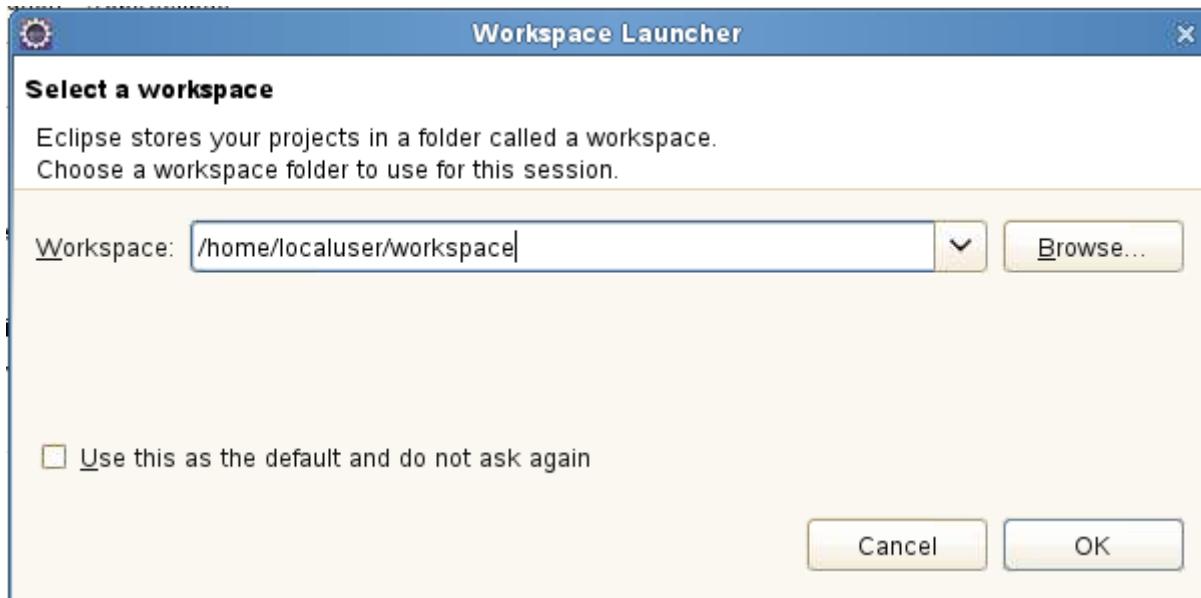
Information

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>/WSecurity 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.



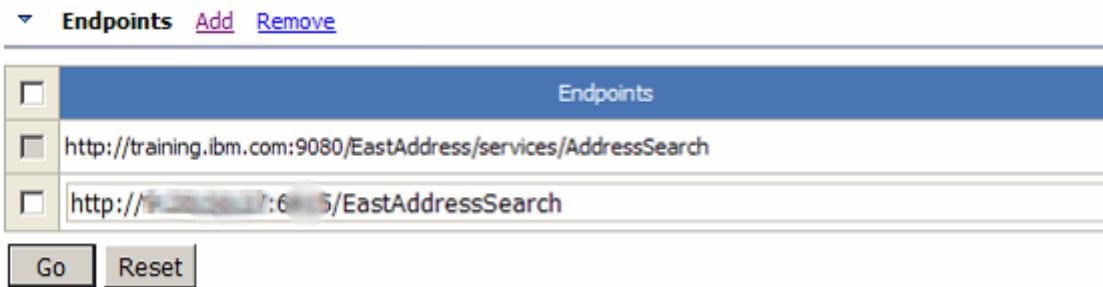
- b. Locate the `EastAddressSearch.wsdl` file in the workspace.
- c. If you are not in the Java EE perspective, switch to it now by clicking **Window > Open Perspective > Other** and then selecting **Java EE**.

- __ d. Open the menu for `EastAddressSearch.wsdl` by right-clicking the selection. Click **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>/EastAddressSearch`, 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
findByName
retrieveAll
findByLocation

- 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>`.

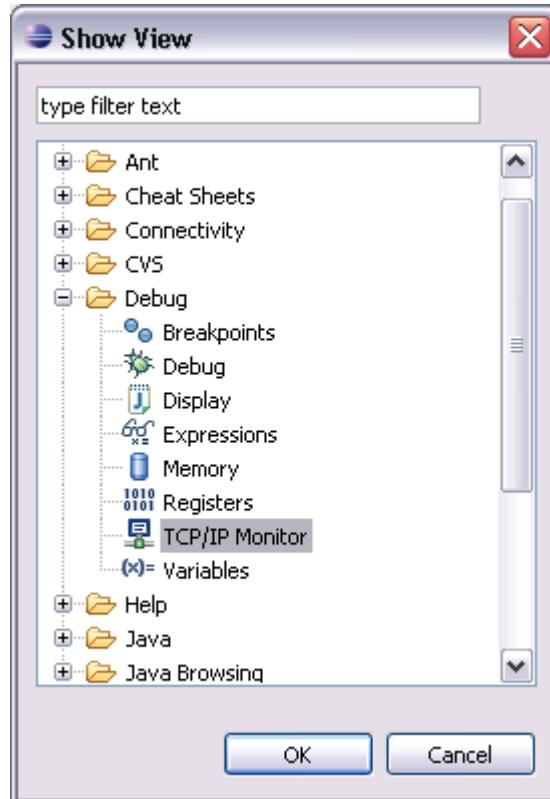


Note

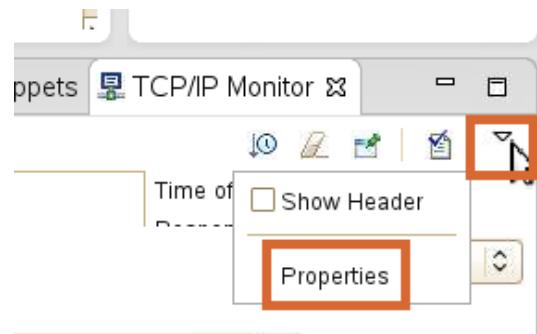
If you get an internal error on the request, verify that the URI specified in the endpoint is `/EastAddressSearch`, rather than `/EastAddress/services/AddressSearch`. `/EastAddressSearch` is what the web service proxy is looking for.

- 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 clicking **Window > Show View > Other**.

- __ b. In the Show View window, expand **Debug** and select **TCP/IP Monitor**.

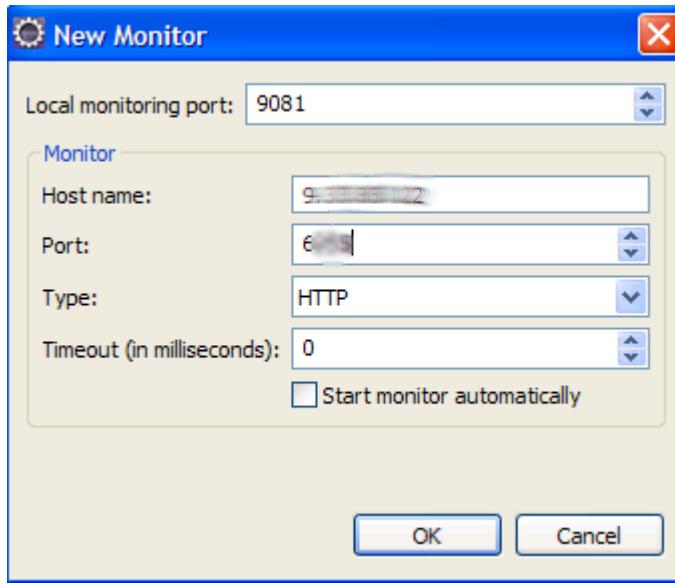


- __ c. Click **OK**. The TCP/IP Monitor view opens.
- __ d. Ensure the TCP/IP Monitor tab is selected. Then, click the drop-down arrow under the tab. When the menu appears, select **Properties**. This action opens the TCP/IP Monitor Preference page.



- __ 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. Select the new monitor entry in the list and 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.

6.5. Test the WestAddressSearch web service

Repeat the steps from **Section 6.4, "Test the EastAddressSearch web service," on page 6-15** 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 6.4, "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, r

```

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" />

```

6.6. 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`.

The screenshot shows the WSDL policy tree view in the DataPower WebGUI. The tree structure is as follows:

- wsdl: WestAddressSearch.wsdl** (highlighted with a red box)
 - service: {http://west.address.training.ibm.com}AddressSearchService**
 - port: {http://west.address.training.ibm.com}AddressSearch**
 - port-operation: findByName** (highlighted with a red box)
 - port-operation: findByName** (highlighted with a red box)

Below the tree, there are tabs for WS-Policy (default), WS-I Conformance (none), and Priority Normal. Under each port-operation node, there is a "Processing Rules" link, which is also highlighted with a red box.

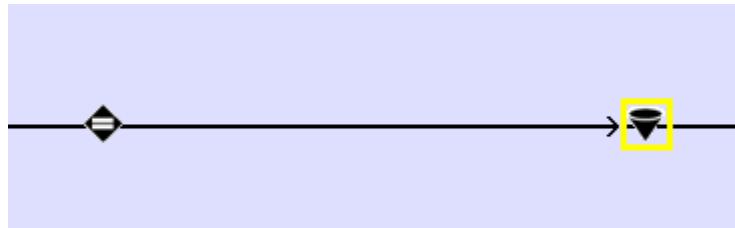
- c. Click **Processing Rules** underneath **port-operation: findByName**. The policy configuration area automatically reloads, and the policy editor displays.

- ___ 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 click **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

- ___ 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.
- ___ 4. 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 default request and response rules are created at the proxy level. The **findByName** operation had a request rule at the operation level. Hence, that request rule was executed. Since no response rule 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	

- ___ 5. 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 the click **Close**.
- ___ 12. On the multi-step probe window, click **Close**.
- ___ 13. Click **Save Config** to save your configuration to the appliance.
- ___ 14. Stop the TCP/IP Monitor.
- ___ 15. Log out of the WebGUI when you are finished.
- ___ 16. Close Eclipse.

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 with Web Services Explorer in Eclipse.

Exercise 7. 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 6: 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>*: 6nn5, port number of the web service proxy

7.1. 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.

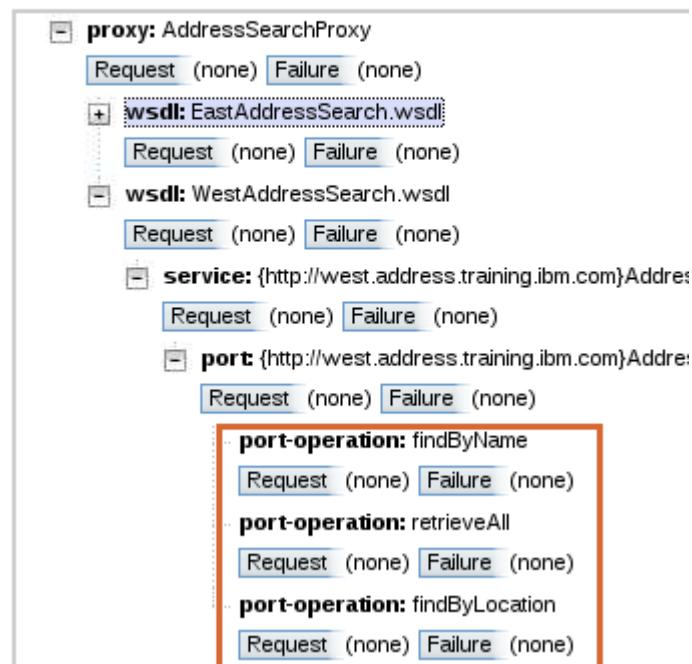
- ___ 1. Use **Troubleshooting** on the Control Panel to set the Log Level to **debug**. This level gives you the most detailed log entries.
- ___ 2. Open **AddressSearchProxy** from the Web Service Proxy icon on the Control Panel.
- ___ 3. Select the **SLM Policy** tab.



- ___ 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: \"\" " -H "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 make it send 10 pairs of cURL commands:

```
./driveSLM.sh <dp_public_ip> <wsp_proxy_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 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.

7.2. 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. The Configure Log Target page is displayed.
 - ___ 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`

Name	<input type="text" value="SLMtarget"/>
<hr/>	
General Configuration	
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text" value="log for SLM only"/>
Target Type	<input type="text" value="File"/>
Log Format	<input type="text" value="XML"/>
Timestamp Format	<input type="text" value="syslog"/>
Feedback Detection	<input type="radio"/> on <input checked="" type="radio"/> off
Identical Event Detection	<input type="radio"/> on <input checked="" type="radio"/> off
<hr/>	
Destination Configuration	
File Name	<input type="text" value="logtemp:///SLMtarget.log"/>

- ___ 5. Set the following values on the **Event Subscriptions** tab:
 - ___ a. Click **Add** in the Events Subscription List.
 - ___ b. Event Category: `slm`

- __ c. Minimum Event Priority: debug

Event Category	Minimum Event Priority
slm	debug

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 is displayed.

- __ b. In the Terminal window, resend the script command to send the cURL commands 10 times:

```
./driveSLM.sh <dp_public_ip> <wsp_proxy_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**.



Note

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.

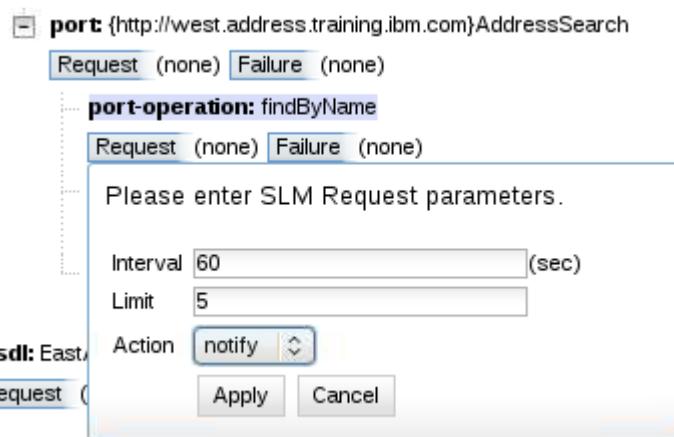
- __ e. The SLMtarget log is still empty, since there is no SLM criteria yet. Switch the log back to **default-log**.

7.3. Add SLM criteria to the web service proxy

Although there is an **SLM action** in the default proxy-level request rule, no criteria is set by default. You must add the criteria if you want to manage the traffic.

In the **WestAddress** WSDL, you want to be notified when the `findByName` operation exceeds five requests per minute, and you want to reject `findByLocation` requests when the rate hits five 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 box, 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 box, 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.



SLM

Use this pane to define service level monitor (SLM) policies to comply with your implemented s

Auto Generated SLM Statements

Define the request or failure SLM policy.

- proxy: AddressSearchProxy
 - [Request (none) Failure (none)]
- wsdl: WestAddressSearch.wsdl
 - [Request (none) Failure (none)]
- service: {http://west.address.training.ibm.com}AddressSearchService
 - [Request (none) Failure (none)]
- 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**.

7.4. 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_proxy_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="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</state><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></iteration>
```

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.xsl' 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 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. Notice that there is none.

- ___ 7. For the **Target** field, change it from **default-log** to **SLMtarget**.

System Log

[Help](#)

[Reset 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							
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.

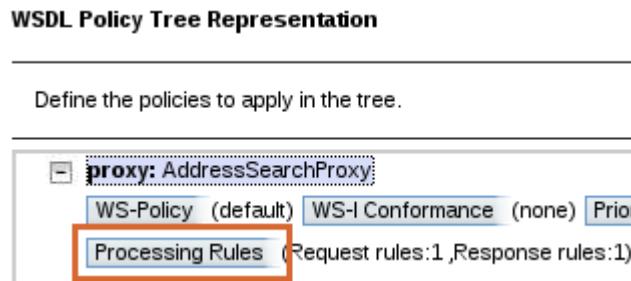
- ___ 8. Notice that no entries are related to the `findByName` requests, which the SLM criteria should generate. That situation is investigated in the next section.

7.5. 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.

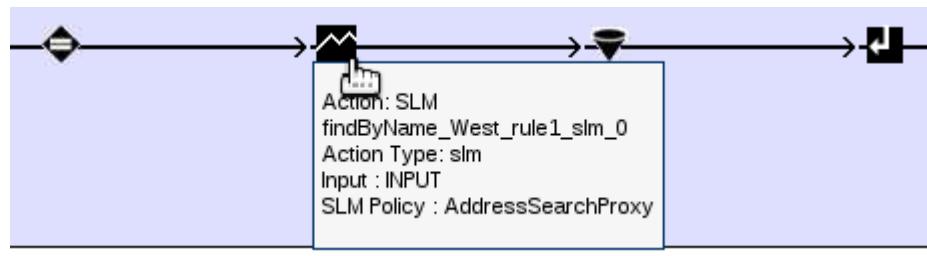


- ___ 4. The policy editor opens near the bottom of the page. Notice the **SLM action** in the default rule. This action is the SLM action that affects the `findByLocation` requests.
- ___ 5. Hover the mouse over the SLM action. Notice that the name of the SLM policy is **AddressSearchProxy**, which 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. The rule is displayed in the policy editor at the bottom of the page. Notice that there is no SLM Action in the rule and hence the SLM statement for this operation had no effect. To correct this operation, 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 require.



- ___ 10. Click **Apply** at the top of the page.

7.6. 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_proxy_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.

System Log Target: **SLMtarget** Filter: **(none)** (none)

1:29:26 on 2012-09-11

Category	level	tid	direction	client	msgid	message	Show last 50 100
12							
	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	0x80e003d4	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**.

7.7. 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, click **Objects > Monitoring > SLM Policy**.

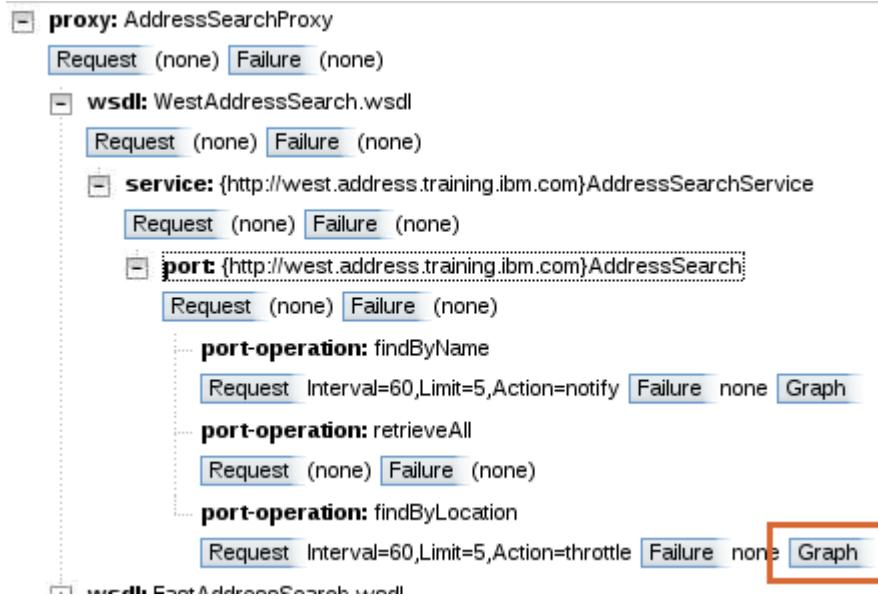
This selection displays the catalog list of the SLM policy objects that are defined in this domain. By default, a web service proxy creates an SLM policy object with the same name as the web service proxy. For a multi-protocol gateway, there is no default SLM policy object, so you must explicitly create 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 created.
- 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 box to create an SLM statement.
- 5. When you are finished reviewing the SLM policy object, return to the catalog list page.

7.8. 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 handy 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 **Graph** for the **findByLocation** operation.

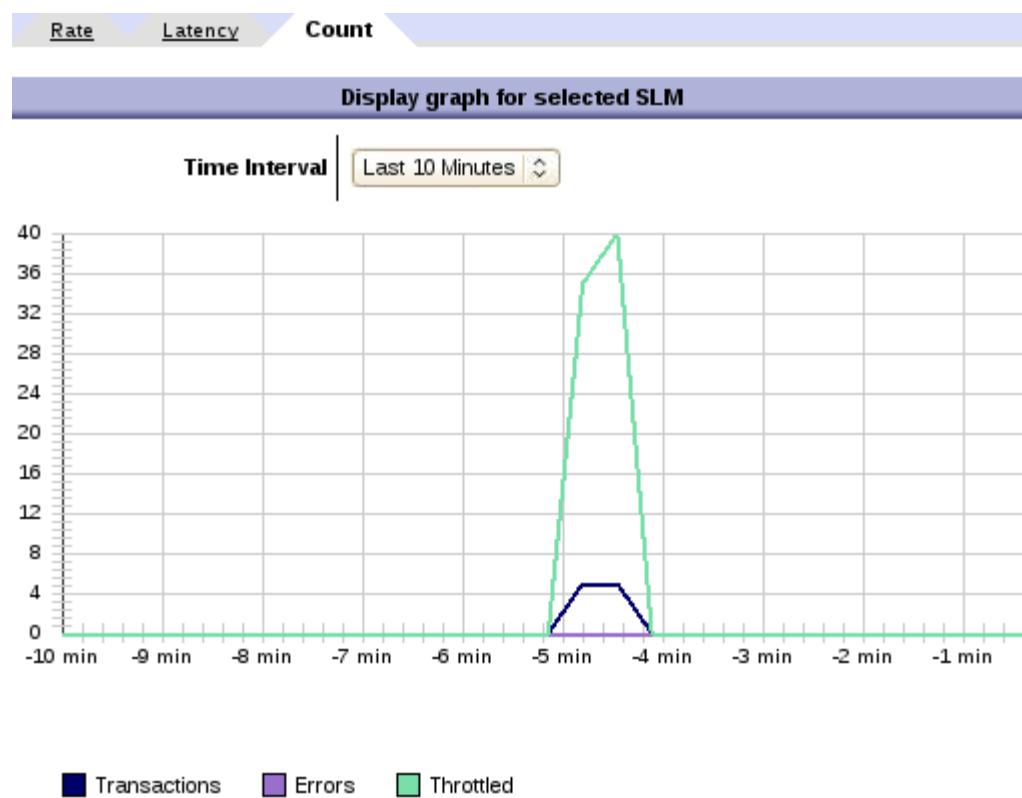
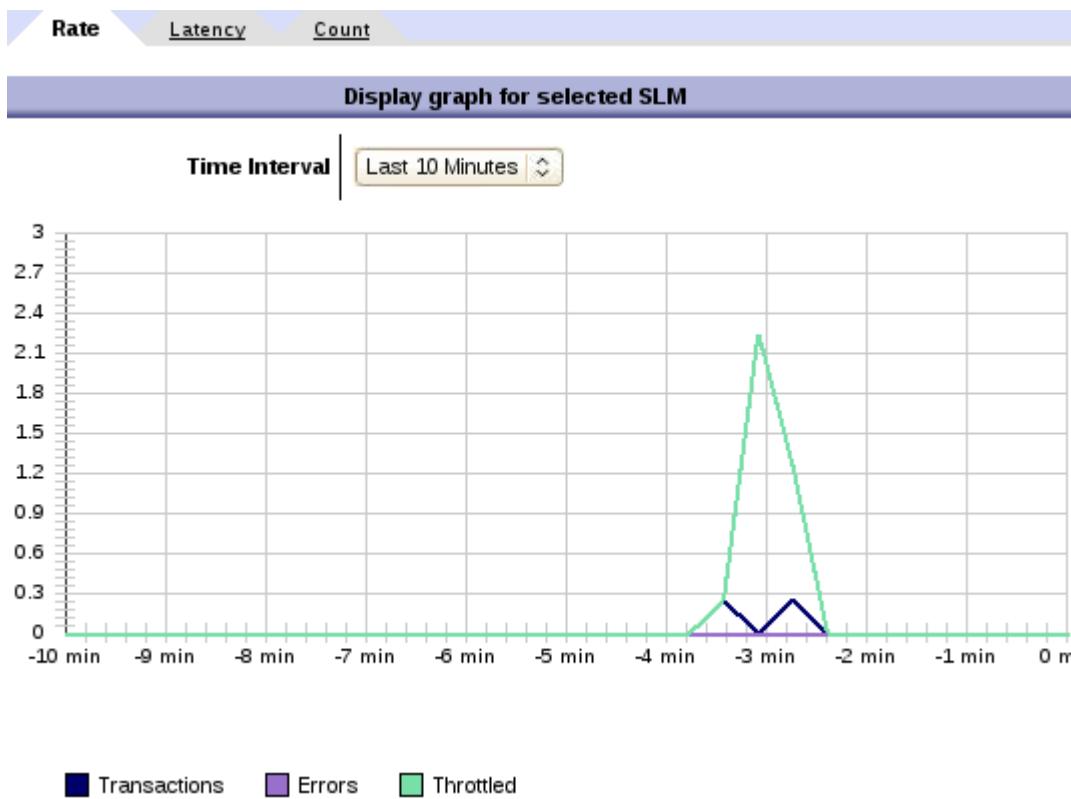


- 5. In the Terminal window, run the test script numerous times in succession, varying the number of times the cURL commands are sent:

```
./driveSLM.sh <dp_public_ip> <wsp_proxy_port> 10
./driveSLM.sh <dp_public_ip> <wsp_proxy_port> 30
./driveSLM.sh <dp_public_ip> <wsp_proxy_port> 15
./driveSLM.sh <dp_public_ip> <wsp_proxy_port> 5
./driveSLM.sh <dp_public_ip> <wsp_proxy_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 8. 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 6: 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>*: *6nn5*, port number of the web service proxy that was created in the previous exercise
 - *<mpgw_crypto_port>*: *7nn2*, port number of the multi-protocol gateway that encrypts and signs messages

8.1. 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. Click **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

The screenshot shows the 'Crypto Key' configuration page. The 'Name' field is highlighted with an orange border and contains the value 'AliceKey'. Below it, the 'Administrative State' is set to 'enabled'. Under 'File Name', a dropdown menu is open, showing 'cert:/// Alice-privkey.pem' which is also highlighted with an orange border. There is an 'Upload...' button next to the dropdown.

- __ 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**.

The screenshot shows a confirmation dialog box with the title 'Crypto Key: AliceKey [up]'. The word '[up]' is highlighted with an orange border. Below the title are four buttons: 'Apply', 'Cancel', 'Delete', and 'Undo'.

**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. Click **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.

Name	<input type="text" value="AliceCert"/> *
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
File Name	<input type="button" value="cert:///"/> <input type="button" value="Alice-sscert.pem"/> <input type="button" value="Details..."/>

- ___ 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.

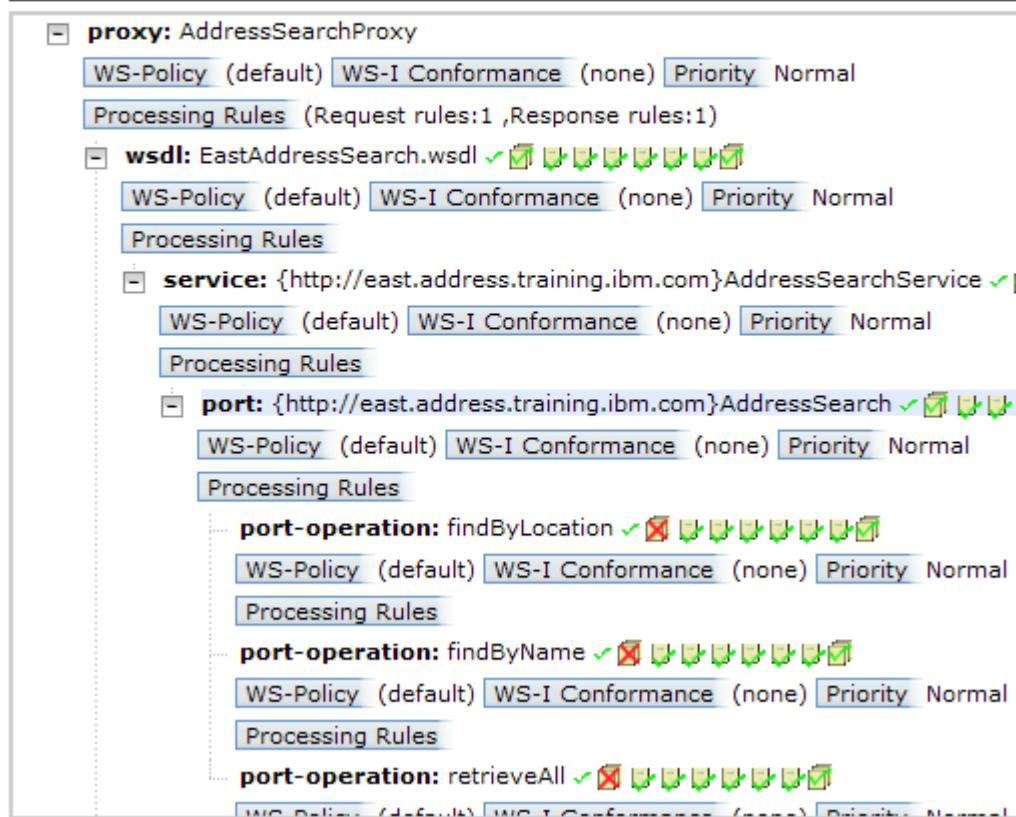
- ___ f. Click **Save Config**.

8.2. 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.

8.3. 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 box, enter: `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	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Local IP Address	dp_public_ip <input type="button" value="Select Alias"/>
Port Number	7072 *
HTTP Version to Client	HTTP 1.1 <input type="button" value=""/>

- __ k. Click **Apply**. The dialog box 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 that is named `match_url_encrypt` that matches the URL `/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

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

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

Front Side Protocol

<input type="text" value="CryptoMPGW_HTTP_FSH (HTTP Front Side Handler)"/>
<input type="text"/>

- ___ 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. Create a matching rule that is named `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.

Envelope Method

- WSSec Encryption
- Standard XML Encryption
- Advanced
- *

Message Type

- SOAP Message
- Raw XML Document
- Selected Elements (Field-Level)
- Advanced
- *

Document Crypto Map

(none) + ... *

Asynchronous

on off

{http://www.datapower.com/param/config}recipient 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 it

- <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

<input type="button" value="Apply"/>	<input type="button" value="Cancel"/>
<hr/>	
Name	<input type="text" value="Encrypto_DCM"/> *
<hr/>	
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Operation	<input type="text" value="Encrypt (WS-Security)"/> *
<hr/>	
XPath Expression	<input type="text" value="/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByName']/*[local-name()='name']"/> /*[local-name()='Envelope']/*[local-name()='name']
<input type="button" value="Add"/> <input type="button" value="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.

8.4. 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. Create a matching rule that is named **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**

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

## 8.5. 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.

Next, you 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-enc** message.



### Note

In the previous section, you used the `CryptoMPGW` multi-protocol gateway to create a message that is 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 the EAR file that causes WebSphere Application Server to use “true” or “false” for the **mustUnderstand** attribute. The DataPower service generates the correct value of `1`, while the back-end server searches 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. From the **Decrypt Key** list, select **AliceKey**.

### General Configuration

**Comments**

**Type**

Dynamic Backend  
 Static Backend  
 Static from WSDL  
\*

**Decrypt Key**

AliceKey

+
...

**EncryptedKeySHA1 Cache Lifetime**

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>
```



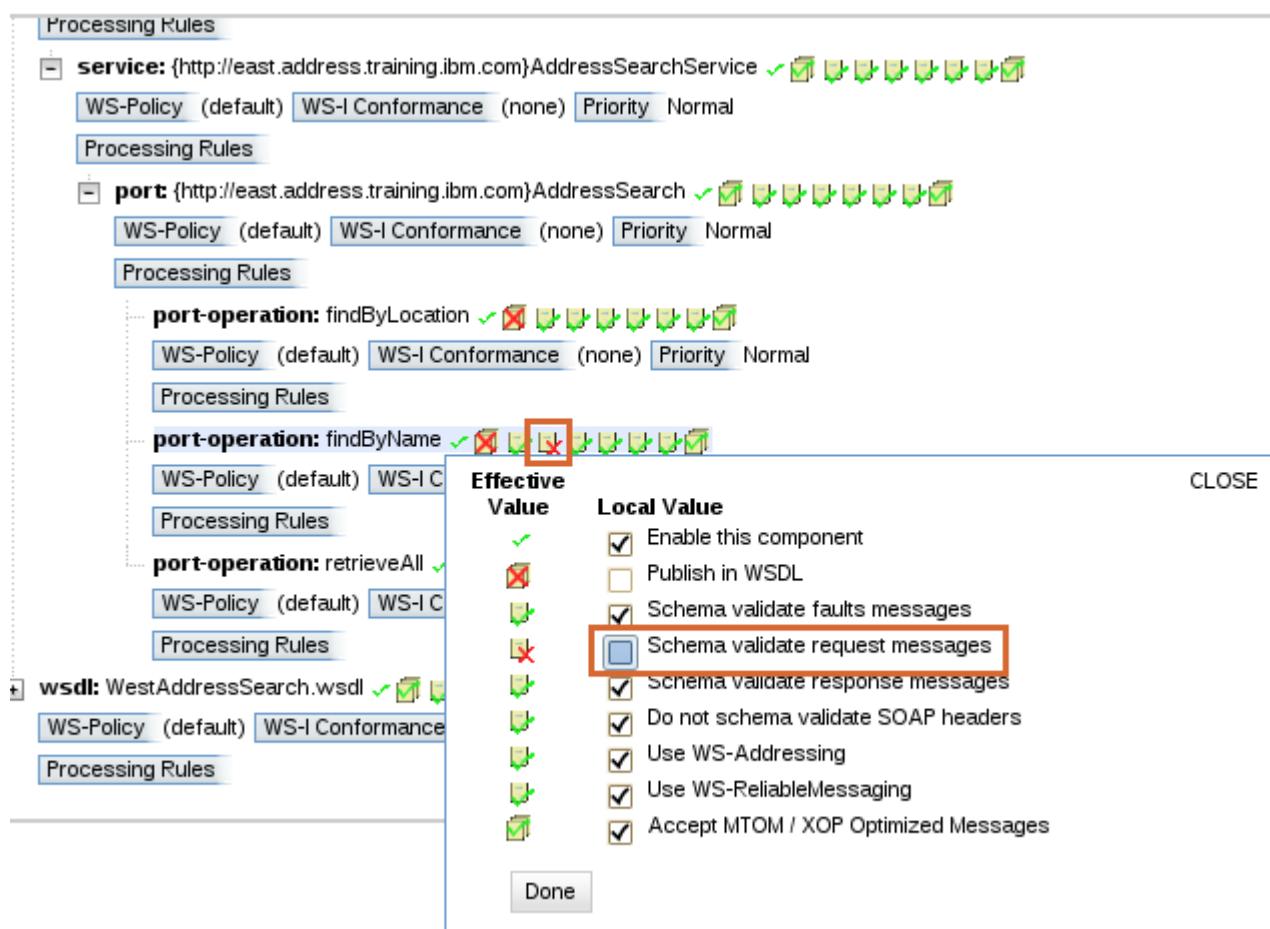
### Information

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.

## 8.6. 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 only specific elements within the message. 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.



- d. Click **CLOSE**.

**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, verify 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**. The Configure Document Crypto Map page is displayed. The XPath expression that you generated is populated in the **XPath Expression** field.
- \_\_\_ l. Click **Apply** to save the crypto map.
- \_\_\_ m. In the Configure Decrypt Action dialog box, from the Decrypt Key list, select `AliceKey`. If the Decrypt Key field is not visible, click the **Advanced** tab.

**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><b>*</b>                                                                                                                                                                                                             |
| <b>Document Crypto Map</b> | <input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="Name_DCM"/> <input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px;" type="button" value="+"/> <input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px;" type="button" value="..."/> *<br><input type="radio"/> on <input checked="" type="radio"/> off |
| <b>Asynchronous</b>        | <input type="radio"/> on <input checked="" type="radio"/> off                                                                                                                                                                                                                                                                                                                   |
| <b>Decrypt Key</b>         | <input style="border: 1px solid #ccc; padding: 2px 10px;" type="button" value="AliceKey"/> <input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px;" type="button" value="+"/> <input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px;" 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 go 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`.

- \_\_\_ 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.

```
<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>

<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.

## 8.7. 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 **CLOSE**.



### 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 the 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>
```

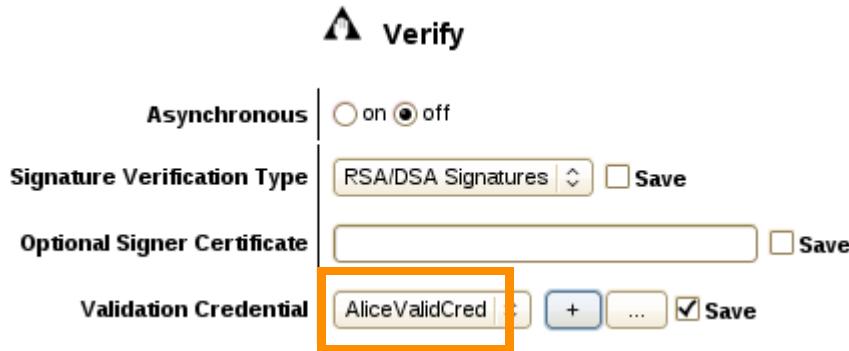
- e. Click **Processing Rules** under the **findByLocation** operation.
- f. Click **New Rule** to create a rule.
- g. Set the Rule Direction to **Client to Server**.
- h. The rule configuration area opens with a **Match** action.
- i. In the rule configuration area, drag a **Verify** action after the **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.



- \_\_ p. Click the **Advanced** tab.
- \_\_ q. Select **off** for **Check Timestamp Expiration**.



### Important

The signed message that you generated earlier inserts a time stamp into the message. If you wait too long to verify the signature, the time stamp 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 go 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.

## 8.8. 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.

Verify that the response is returned correctly in the Terminal window (Steve Anderson in New York).

- \_\_\_ 3. Save the configuration.

## 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 9. 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

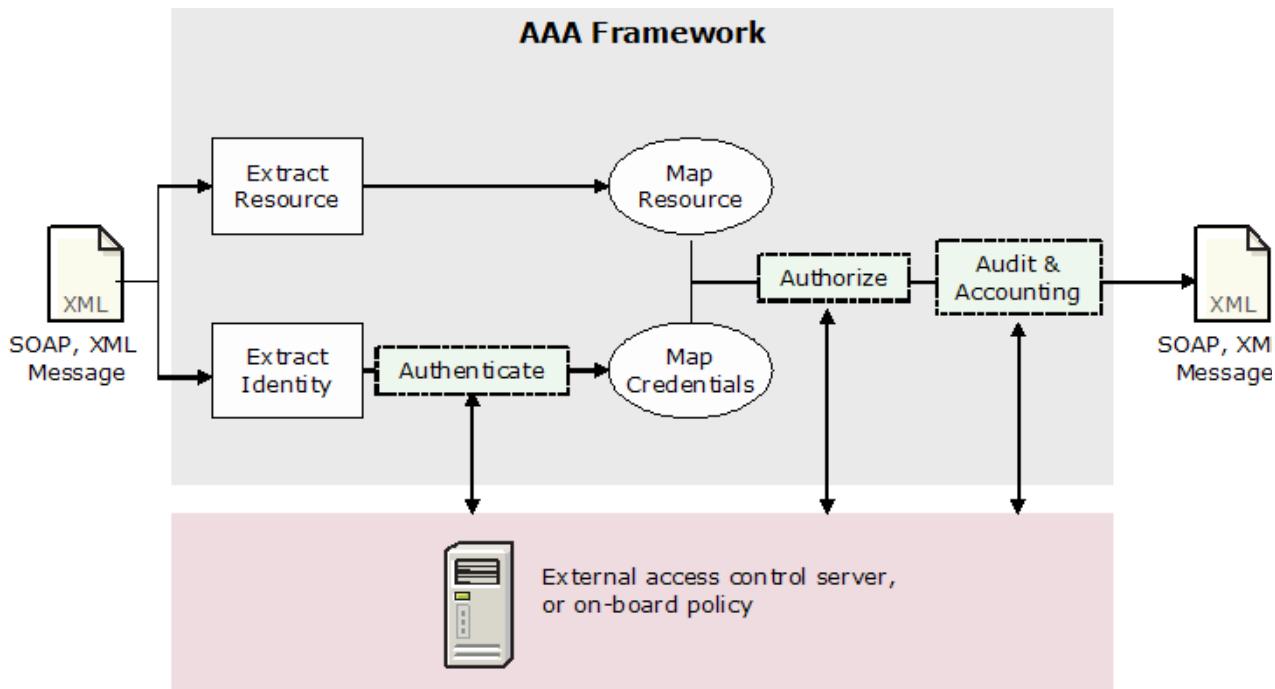
Every instance of the Address Search web service includes an operation to download all address entries. The retrieveAll operation returns a sequence of entries with name and address details. For privacy concerns, normal users should not be allowed to call this operation. Only developers should have access 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

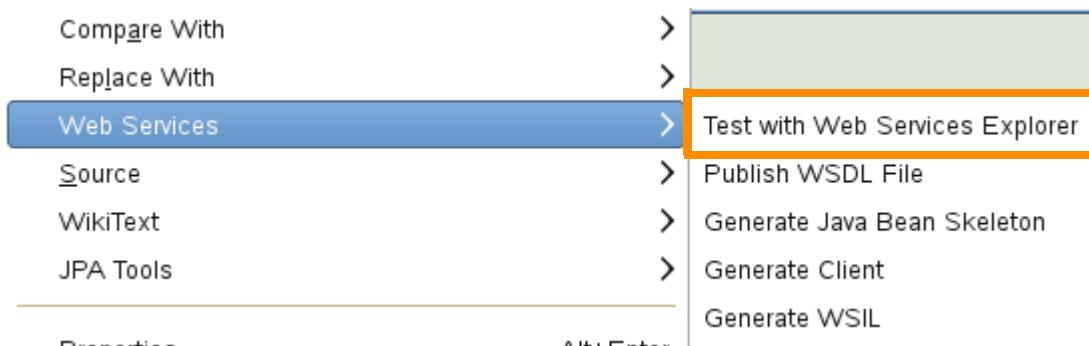
## Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- This exercise also depends on the previous completion of “Exercise 6: 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>*: 6nn5, port number of the web service proxy that is created in a previous exercise

## 9.1. 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 click **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.

Name	Documentation
<a href="#">retrieveAll</a>	--
<a href="#">findByLocation</a>	--
<a href="#">findByName</a>	--

Endpoints	
<input type="checkbox"/>	http://training.ibm.com:9080/EastAddress/services/AddressSearch

**Endpoints**

**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	
<input type="checkbox"/>	http://training.ibm.com:9080/EastAddress/services/AddressSearch
<input type="checkbox"/>	http://172.16.78.44:6555/EastAddressSearch

**Endpoints**

**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. From 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.

Status

[Source](#)

▼ retrieveAllResponse

  ▼ retrieveAllReturn

    ▼ Address

      ▼ details

        city (string): Atlanta

        state (string): GA

        street (string): 1477 Highland Boulevard

        zipCode (string): 30309

    ▼ name

      firstName (string): Timothy

      lastName (string): Green

      title (string): Mr

  ▼ Address

    ▼ details

      city (string): Milwaukee

## 9.2. 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 request. 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

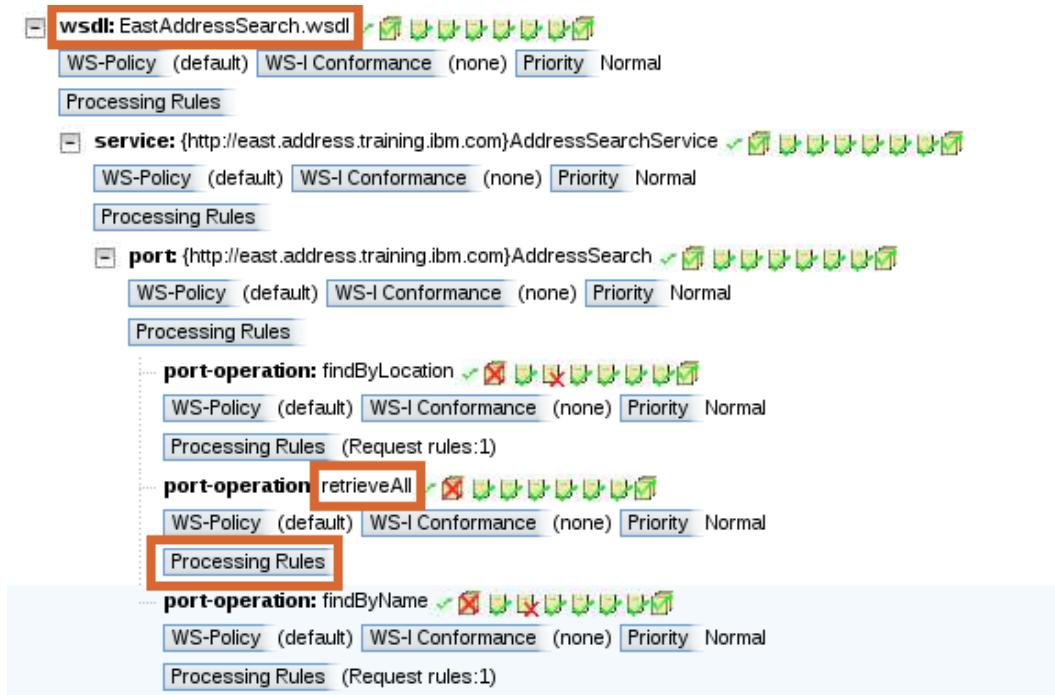
The Web Service Proxy Policy page enforces one or more processing rules at the proxy, WSDL document definition, WSDL service, WSDL port, and WSDL operation levels. There are two default rules for any service that is associated with the web service proxy:

- 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**.

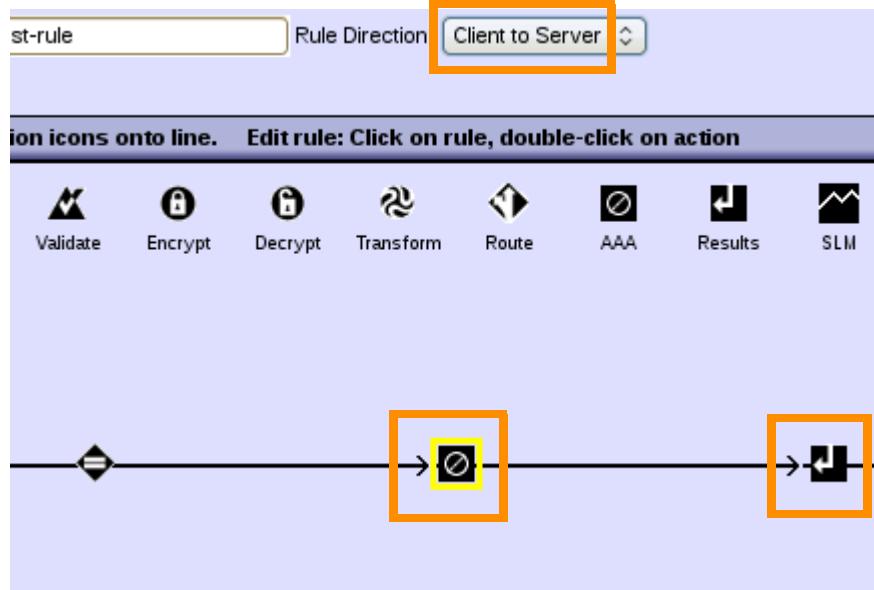
Keep in mind that only one policy rule is applied for each request or response message that passes through the web service proxy. That is, if a request message matches a user-defined rule at the WSDL port level, the proxy does not run the default request rule.

- 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**.



- \_\_ c. Enter `retrieveAll_AAA_Request_Rule` as the new **rule name**.  
 \_\_ d. Change the rule direction to **Client to Server**.  
 \_\_ e. Drag a **AAA** action after the **Match** action.  
 \_\_ f. Drag a **Results** action to the right of the **AAA** action.



- \_\_ 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 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.

- \_\_\_ b. Click **Next**.

- \_\_\_ 9. Use the sample DataPower AAA information XML file, `AAAInfo.xml`, to authenticate the user.

- \_\_\_ a. Select **Use AAA information file** as the method.



#### 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`. The file is included with every DataPower appliance and 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 LTPA token <input type="radio"/> Accept SAML assertion with valid signature <input type="radio"/> Bind to LDAP server <input type="radio"/> Contact ClearTrust server <input type="radio"/> Contact IBM Security Access Manager <input type="radio"/> Contact Netegrity SiteMinder <input type="radio"/> Contact NSS for SAF authentication <input type="radio"/> Contact SAML server for SAML Authentication statement <input type="radio"/> Contact WS-Trust server for WS-Trust token <input type="radio"/> Custom template <input type="radio"/> Pass identity token to authorization phase <input type="radio"/> Previous SAML assertions that corresponds to SAML Browser <input checked="" type="radio"/> Use AAA information file <input type="radio"/> Use certificate from Binary Security Token <input type="radio"/> Use established WS-SecureConversation security context <input type="radio"/> Use RADIUS server <input type="radio"/> Validate Kerberos AP-REQ for server principal <input type="radio"/> Validate signer certificate for digitally signed message <input type="radio"/> Validate SSL certificate from connection peer
URL	<b>store:///</b> <b>AAAInfo.xml</b> <input type="button" value="+"/> <input type="button" value="..."/> <input type="button" value="Upload.."/>

- \_\_ 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 **Local name of the request element** 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.

**AAA Policy Name:** AddressAdmin

**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.**

<b>Method</b>	None	*
---------------	------	---

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. On the Configure An Access Control Policy page, examine the monitoring, logging, and postprocessing options available with the access control policy.



**Note**

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 extra 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.

- 
- \_\_\_ 12. Click **Commit** to save the `AddressAdmin` access control policy, and then click **Done**.
  - \_\_\_ 13. Click **Done** to save the settings for the AAA action in the `retrieveAll` policy rule.
  - \_\_\_ 14. Click **Done** to confirm the settings within the action.



**Information**

---

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 **Apply** on the top of the Web Service Proxy Policy page.

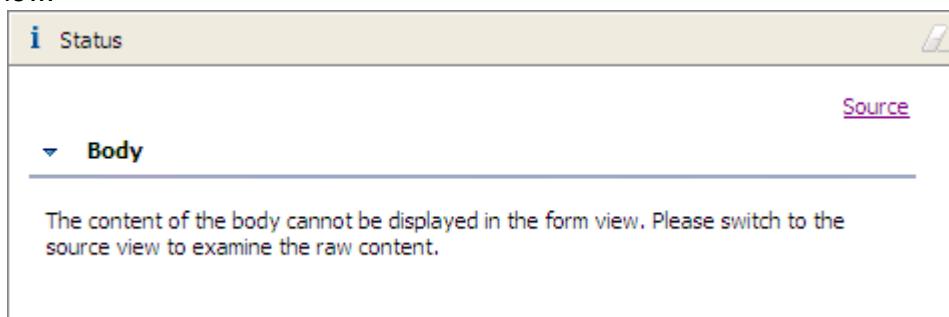
### 9.3. Test the access control policy for a web service operation

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

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. Right-click the `EastAddressSearch.wsdl` WSDL document and then click **Web Services Explorer**.
  - 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:
- The operation fails **AAA Authentication** because no WS-Security user name element is shown in the SOAP request message header.
  - The operation also fails **AAA Authorization** as only authorized clients are allowed to access the specified resource.
  - The second entry from the top identifies the AAA policy rule that rejected the message: **retrieveAll\_AAA\_Request\_Rule**.

- The first entry from the top indicates that the AddressSearchProxy sent back a SOAP fault message to the client.

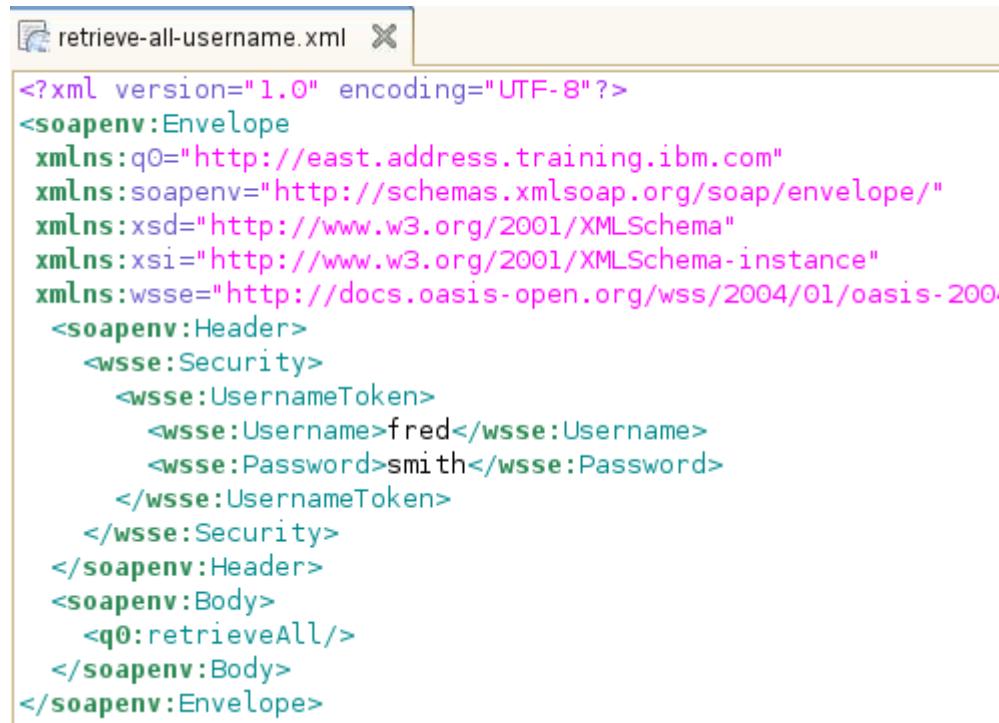
0x00d30003	wsgw (AddressSearchProxy): Rejected by filter; SOAP fault sent
0x80c00009	wsgw (AddressSearchProxy): request retrieveAll_AAA_Request_Rule #1 aaa: 'INPUT AddressAdmin stored in PIPE' failed: Rejected by policy.
0x80c00010	wsgw (AddressSearchProxy): Processing of 'store:///dp/aaapolicy.xls' stopped: Rejected by policy.
0x8380000e	wsgw (AddressSearchProxy): Message rejected
0x80c00037	wsgw (AddressSearchProxy): Reject set: Rejected by policy.
0x01d30002	wsgw (AddressSearchProxy): AAA Authorization Failure
0x838000f2	wsgw (AddressSearchProxy): anyauthenticated authorization failed with credential 'SPECIAL-FORMAT-NOT-PRINTED' for resource 'retrieveAll'
0x838000f0	wsgw (AddressSearchProxy): Mapped-Credentials format is special. One or more 'entry/ child elements are expected.
0x838000ef	wsgw (AddressSearchProxy): Cached Authorize entry
0x838000ee	wsgw (AddressSearchProxy): Authorizing with anyauthenticated
0x838000ec	wsgw (AddressSearchProxy): Authorize Caching is on: absolute
0x838000e8	wsgw (AddressSearchProxy): Mapping resources using none
0x83800099	wsgw (AddressSearchProxy): Mapping credentials using none
0x83800014	wsgw (AddressSearchProxy): Cached Authenticate entry
0x01d30001	wsgw (AddressSearchProxy): AAA Authentication Failure
0x83800015	wsgw (AddressSearchProxy): xmlfile authentication failed with (wssec-username, password='*****')
0x8380004a	wsgw (AddressSearchProxy): No matching Authenticate entries found.

\_\_\_ d. Close the log window.

## 9.4. 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, click **File > Open File**.
  - 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-secext-1.0.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. Examine the `AAAinfo.xml` file to view the authentication entry for **Username** of “fred”.
  - a. Click **Administration > Main > File Management** from the navigation bar.
  - b. Expand the **store:** directory.
  - c. Select the **AAAinfo.xml** entry.
  - d. A new browser page opens to display the contents of the file.

- \_\_\_ e. Locate the entry for **fred**. Notice that this entry matches the username and password that is in the XML request.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
 Licensed Materials - Property of IBM
 IBM WebSphere DataPower Appliances
 Copyright IBM Corporation 2007,2009. All Rights Reserved.
 US Government Users Restricted Rights - Use, duplication or
 restricted by GSA ADP Schedule Contract with IBM Corp.
-->
<AAAInfo xmlns="http://www.datapower.com/AAAInfo">
 <FormatVersion>1</FormatVersion>
 <Filename>store:///AAAInfo.xml</Filename>
 <Summary>This is an example of the file format.</Summary>

 <Authenticate>
 <Username>fred</Username>
 <Password>smith</Password>
 <OutputCredential>admin</OutputCredential>
 </Authenticate>

 <Authenticate>
 <Username>tanvf</Username>

```

- \_\_\_ f. Close the browser page.
- \_\_\_ 3. 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 -H
"Content-type: text/xml"
```
- ___ c. Verify that the SOAP response message succeeds, returning the entire list of address entries.

9.5. 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.

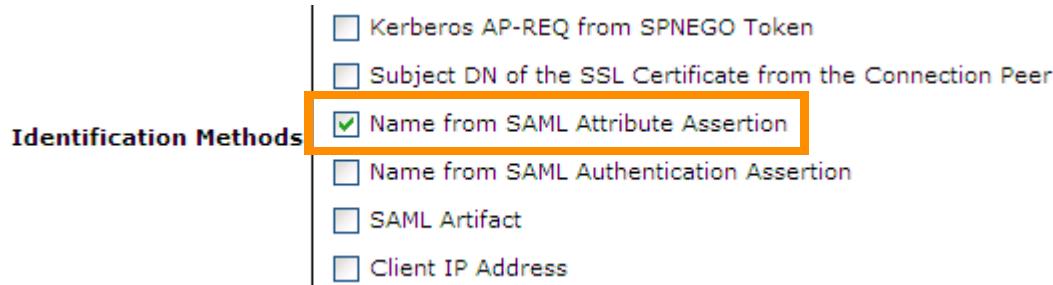
The Security Assertion Markup Language (SAML) provides a framework for creating and exchanging security information in an XML format. DataPower appliances can use SAML tokens, and generate new SAML assertions after performing a AAA action.

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**.

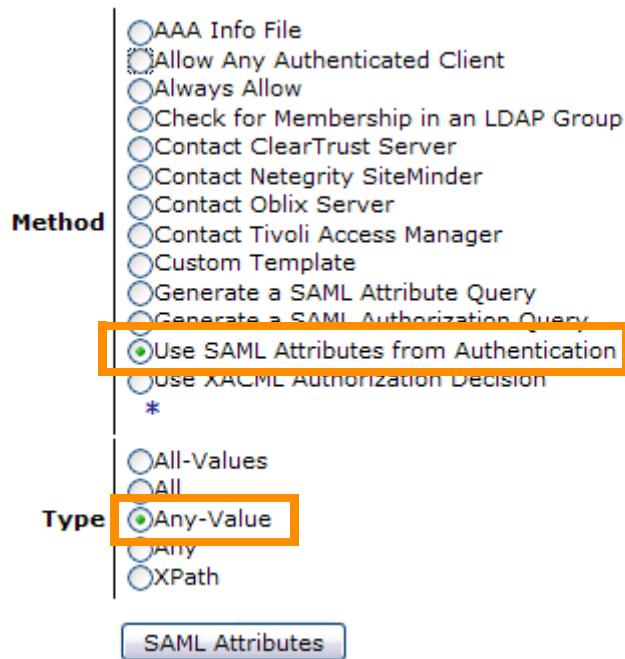


- ___ b. Click **Next**.
- ___ c. In the “Define how to authenticate the user” page, select **Accept a SAML Assertion with a Valid Signature**. This policy accepts the name that is given in the SAML assertion if the digital signature for the assertion is valid.



- ___ 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]
AddressSearchProxy *

Apply Cancel Delete Refresh

[View Log](#) | [View Status](#) | [View C](#)



9.6. 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, click **File > Open File**.
 - 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
QTELMAkGA1UECBMCT04xE DAOBgNVBAcTB1Rvcm9udG8xDDAKBgNVBAoTA01CTTEX
MBUGA1UECxMOU29mdHdhcmUgR3JvdXAxDjAMBgNVBAMTBUFsaWN1MB4XDTA2MDcz
MTE4NTIyMFoXDTA5MDczMDE4NTIyMFowYzELMAkGA1UEBhMCQOExCzAJBgNVBAgT
Ak9OMRAwDgYDVQQHEwdUb3JvbnRvMQwwCgYDVQQKEwNJQkOxFzAVBgnVBAsTD1Nv
ZnR3YXJ1IEdyb3VwMQ4wDAYDVQQDEwVBbG1jZTCBnzANBkgqhkiG9wOBAQEFAAOB
jQAwgYkCgYEAxhpa/OJyJf4+DiPSfjZWAH6rvBRYtpk62iN2SLxdyeWUWQU/BHF
13qBBKincIEnhBDKJvJgCCTDhsQqKUVmKO2pkaV/Av6WXxHVSmcMRp32y7a7atnp5
+1hLJmDyD86zH+HSosEL9tq5hBw6riTdhBW8N68nYSpqRftCXUokSY8CAwEAAaOB
ODCBzTAMBgNVHRMEBTADAQH/MBOGA1UdDgQWBBSeAYCmEf/9jUrf5pr8kcOWoU2n
eTCBkAYDVROjBIGIMIGfGBSeAYCmEf/9jUrf5pr8kcOWoU2neaFnpuGUwYzELMAkG
A1UEBhMCQOExCzAJBgNVBAgTAK9OMRAwDgYDVQQHEwdUb3JvbnRvMQwwCgYDVQQK
EwNJQkOxFzAVBgnVBAsTD1NvZnR3YXJ1IEdyb3VwMQ4wDAYDVQQDEwVBbG1jZYIE
UI2vzTALBgNVHQ8EBAMCArwDQYJKoZIhvcNAQEFBQADgYEAtuSSf1i7wLUEqmTt
mysvv6/MgiIW/9F01VmMgCzxhkMS700ESi+NwIKpBv3sOCYLu6951GkSIHa6dET9Q
FpaYim3IjkUCFWDE1AkpbJbcH45V1wkRkfNhOsuX1QSEUWMtxYDz861E+BaN2ekV
Pn1EO6kPdPL7QO63wcoGO+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 -H  
"Content-type: text/xml"
```
 - ___ 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 that is 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 10. Define a three-legged OAuth scenario that uses DataPower services

What this exercise is about

In this exercise, you define the DataPower objects that are needed to implement a three-legged OAuth scenario: an OAuth client profile, an OAuth client group, a web token service, and a resource server. During the service creation, you create a AAA policy that specifies OAuth. Finally, you test the implementation by invoking a client that runs in a web server.

What you should be able to do

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

- Define an OAuth Client Profile and an OAuth Client Group object
- Create a AAA policy to support the OAuth protocol
- Configure a DataPower web token service
- Configure a DataPower implementation of an OAuth resource server

Introduction

DataPower provides multiple options and objects to support the OAuth 2.0 framework. It also supports the three-legged scenario for OAuth, which is the scenario for which OAuth was devised.

In this exercise, you configure the DataPower objects to support the three-legged scenario. You are supplied with the OAuth client code, which runs as a PHP module in a web server. You interact with the scenario as a resource owner by using a browser. The back-end resource application is implemented as a multi-protocol gateway on the appliance. You create a web token service to act as the OAuth authorization server endpoint and the token endpoint. You also create a multi-protocol gateway to act as the OAuth resource server, which functions as the security gateway for the back-end application. The OAuth client, the PHP module, is defined to the appliance as an OAuth Client object.

The supplied PHP module requires some user interaction to complete its function. This interaction is done to allow you to see some of the underlying interactions. During testing, you act as the resource owner at times, and at other times as part of the PHP module.

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)
- Access to the `<lab_files>` directory

Exercise instructions

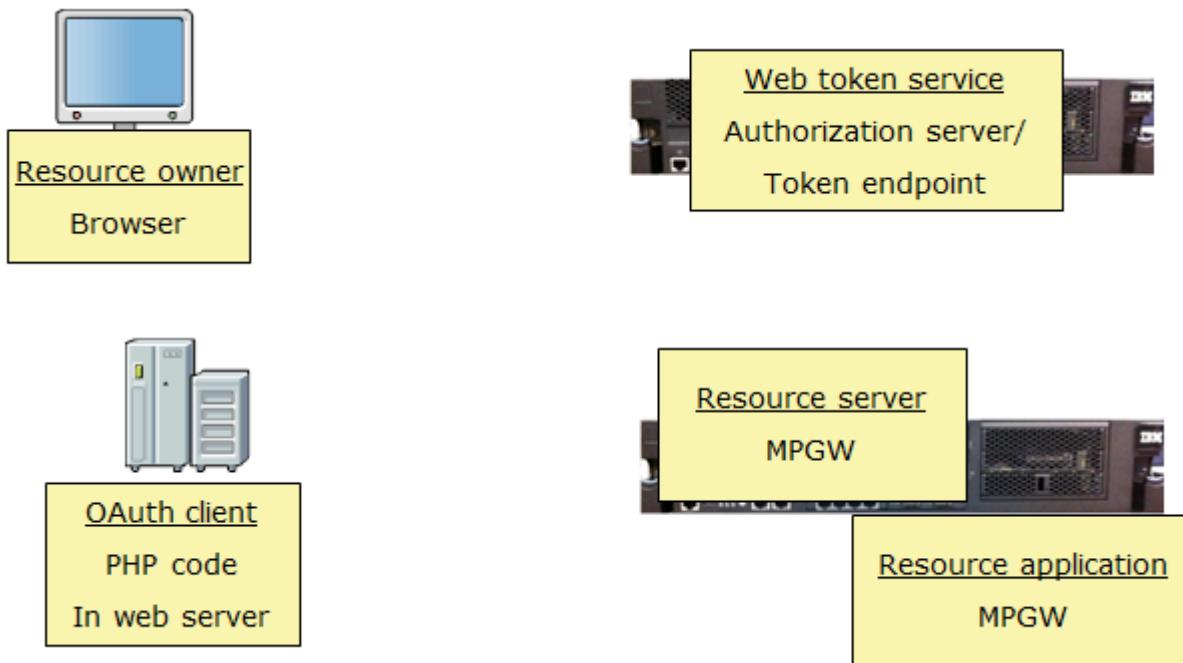
Preface

- 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 public services on the appliance
 - *<studentnn_image_ip>*: IP address of the student image
 - *<oauth_wts_port>*: *7nn0*, port number of the web token service
 - *<oauth_ep_port>*: *7nn3*, port number of the multi-protocol gateway that acts as the OAuth enforcement point for the resource server
 - *<oauth_resource_port>*: *7nn7*, port number of the multi-protocol gateway that acts as the back-end application

10.1.The three-legged architecture for this exercise

For this exercise, you are implementing a three-legged OAuth architecture:

- The resource owner uses a browser as its user agent.
- The OAuth client is implemented in a PHP module under a web server.
- The authorization server is a web token service that supports the authorization and token endpoints.
- The resource server is implemented in a multi-protocol gateway. When the client's access token is validated, it passes the resource request to the back-end resource application.
- The back-end resource application is a multi-protocol gateway. For this example, the resource application does not perform any application function, but it creates a JSON response that is returned to the client and the resource owner.



For educational reasons, the PHP code is not a complete OAuth client. It exposes some of the OAuth protocol interactions to the user, and requires some user interactions to proceed. When you test the code at the end of the exercise, you are acting as the resource owner at times, and as the client at other times.

10.2.Prepare the security objects

The OAuth exercise uses SSL, and a shared secret (symmetric) key. In this section, you define the associated objects that you use in later configurations.

- ___ 1. Log on to the DataPower WebGUI.
- ___ 2. Load the crypto files into the **cert:** directory.
 - ___ a. From the **Control Panel**, click **File Management**.
 - ___ b. On the File Management page, click the **Actions** link for the **cert:** directory, and click **Upload Files**.
 - ___ c. Browse to the <lab_files>/oauth directory, and click **sskey.txt** to upload.
 - ___ d. Repeat the same steps to upload **sslserver.pem**.
- ___ 3. Define the shared secret key object.
 - ___ a. On the **Control Panel**, enter **shared** in the search field that is above the navigation bar.
 - ___ b. Click **Crypto Shared Secret Key** from the resulting list.
 - ___ c. From the **Configure Crypto Shared Secret Key** list, click **Add**.
 - ___ d. For the shared secret key object, enter **oauth-token-ssecret** for the name.
 - ___ e. For File Name, select **sskey.txt** in the cert: directory.
 - ___ f. Click **Apply**.
- ___ 4. Define the crypto (asymmetric) key object.
 - ___ a. On the **Control Panel**, enter **key** in the search field.
 - ___ b. Select **Crypto Key** from the resulting list.
 - ___ c. Click **Add** on the catalog list page.
 - ___ d. Enter the name as **oauth-ssl-server**.
 - ___ e. Select the **sslserver.pem** file in the **cert:** directory.
 - ___ f. Click **Apply**.
- ___ 5. Define the related certificate object.
 - ___ a. On the **Control Panel**, enter **cert** in the search field.
 - ___ b. Select **Crypto Certificate** from the resulting list.
 - ___ c. Click **Add** on the catalog list page.
 - ___ d. Enter the name as **oauth-ssl-server**.
 - ___ e. Select the **sslserver.pem** file in the **cert:** directory.
 - ___ f. Set **Ignore Expiration Dates** to **on**.
 - ___ g. Click **Apply**.
- ___ 6. Define the identification credential object.
 - ___ a. On the **Control Panel**, enter **cred** in the search field.

- ___ b. Select **Crypto Identification Credentials** from the resulting list.
 - ___ c. Click **Add** on the catalog list page.
 - ___ d. Enter the name as oauth-ssl-server.
 - ___ e. For Crypto Key, select **oauth-ssl-server**.
 - ___ f. For Certificate, also select **oauth-ssl-server**.
 - ___ g. Click **Apply**.
- ___ 7. Create the crypto profile object.
- ___ a. On the **Control Panel**, enter `prof` in the search field.
 - ___ b. Select **Crypto Profile** from the resulting list.
 - ___ c. Click **Add** on the catalog list page.
 - ___ d. Enter the name as oauth-ssl-server.
 - ___ e. For the Identification Credentials, select **oauth-ssl-server**.
 - ___ f. Leave the other fields at their defaults.
 - ___ g. Click **Apply**.
- ___ 8. Create the SSL proxy profile object.
- ___ a. On the **Control Panel**, enter `prof` in the search field.
 - ___ b. Select **SSL Proxy Profile** from the resulting list.
 - ___ c. Click **Add** on the catalog list page.
 - ___ d. Enter the name as oauth-ssl-server.
 - ___ e. Set the SSL Direction to **Reverse**.
 - ___ f. For Reverse (Server) Crypto Profile, also select **oauth-ssl-server**.
 - ___ g. Leave the other fields at their defaults.
 - ___ h. Click **Apply**.
- ___ 9. The objects that are needed for the security-related aspects are now defined. They are referenced in later steps as they are needed. Click **Save Config** to persist them.

10.3. Define the OAuth client

- 1. On the Control Panel, enter `oauth` in the search field above the navigation bar.
- 2. Click **OAuth Client Profile**.
- 3. On the Configure OAuth Client Profile catalog list page, click **Add**.
- 4. Define the OAuth Client object:



Information

When a web token service or OAuth-related resource server are configured, they must define the group of predefined OAuth clients that might connect to them. This object defines the characteristics for a specific OAuth client. When the actual OAuth client contacts the OAuth-related service, it passes information that identifies itself. The service looks in its list of OAuth client objects to identify which one it is, determine its operating characteristics, and verify that it really is that client.

- a. Enter the Name as `my-oauth-client`. The name of this object is also the OAuth client ID that the actual OAuth client sends to identify itself.
- b. For the OAuth Role, select **Authorization and Token Endpoints** and **Enforcement Point for Resource Server**. This option specifies how the appliance can act when this client contacts it.
- c. For the Supported Authorization Grant Type, select **Authorization Code**.

Name

*

General

Administrative State

 enabled disabled

Comments

Customized OAuth

OAuth Role

- Authorization and Token Endpoints
- Enforcement Point for Resource Server

*

Supported Authorization Grant Type

- Authorization Code
- Implicit Grant
- Resource Owner Password Credential
- Client Credentials

*

- __ d. Set the Client Type to **Confidential**.
- __ e. For Authentication Method, select **Secret**.
- __ f. Clear the **Generate Client Secret** check box.
- __ g. For Client Secret, enter clientsecretpassword.

**Warning**

Depending on which firmware version the appliance is it, there might be a difference at this point. For V6.0.0.3, you must go into the Authorization and Token Endpoints section of the page and clear the **Generate Client Secret** check box. After you do that, the Client Secret entry field appears.

- __ h. Enter the Scope as /getAccountInfo.
- __ i. Select a Shared Secret of **oauth-token-ssecret**. This symmetric key is used to encrypt the token as it is passed around in the OAuth protocol.
- __ j. For the Redirect URI, enter https://{}\\S+. This regular expression allows any redirect URI to be acceptable for this OAuth client. Be sure to click **Add**.
- __ k. Select the Authorization Form **OAuth-Generate-HTML.xsl** that is in the **store:** directory. This sample style sheet generates the “grant permission” challenge to the user.

| | |
|------------------------|--|
| Client Type | <input type="text" value="Confidential"/> * |
| Authentication Method | <input type="text" value="Secret"/> * |
| Client Secret | <input type="text" value="clientsecretpassword"/> * |
| Customized Scope Check | <input type="checkbox"/> |
| Scope | <input type="text" value="/getAccountInfo"/> * |
| Shared Secret | <input type="text" value="oauth-token-ssecret"/> <input type="button" value="+"/> <input type="button" value="..."/> * |

Authorization and Token Endpoints

| | |
|--------------------|--|
| Redirect URI | <input type="text" value="https://{}\\S+"/> <input type="button" value="X"/>
<input type="text" value="https://{}\\S+"/> <input type="button" value="add"/> |
| Authorization Form | <input type="text" value="store:///"/> <input type="text" value="OAuth-Generate-HTML.xsl"/> |

- __ l. Click the **Advanced** tab to display more settings.

- ___ m. Specify an Additional OAuth Process style sheet. Upload the **add-processing.xsl** file from the <lab_files>/oauth directory into your domain's **local:** directory. This sample style sheet adds an HTTP header of "ResourceOwner" with a value of the verified resource owner name if the processing is during the enforcement point of the resource server.

Name *

General

Caching

Additional OAuth Process

- ___ n. Notice the default life times of the grant code and access token.
___ o. Click **Apply**.

10.4. Define the OAuth client group

When a web token service or an OAuth-enforcing resource server is configured in DataPower, there might be multiple OAuth clients involved. Rather than having to list each client in the services, DataPower provides an OAuth client group object that gathers any related clients together.

- ___ 1. On the **Control Panel**, enter `oauth` in the search field above the navigation bar.
- ___ 2. Click **OAuth Client Group**.
- ___ 3. On the Configure OAuth Client Group catalog list page, click **Add**.
- ___ 4. Define the OAuth Client Group object:
 - ___ a. Enter the Name as `oauth-client-group`.
 - ___ b. Select the OAuth Roles of **Authorization and Token Endpoints** and **Enforcement Point for Resource Server**. This option specifies what roles the associated DataPower services can perform when they are working with this group.
 - ___ c. For the Client list, select the OAuth client object you created: **my-oauth-client**.
 - ___ d. Click **Add**.
 - ___ e. Click **Apply**.
- ___ 5. Click **Save Config** to persist the OAuth client-related objects.

10.5.Define the AAA policy for the web token service

In this section, you create the AAA policy that implements the authorization server and token endpoint AAA behavior for a web token service. This policy is used later in the service policy for the web token service.

The following approach defines the AAA policy object by using the Objects approach, rather than a wizard.

- 1. On the **Control Panel**, enter aaa in the search field above the navigation bar.
- 2. Click **AAA Policy**.
- 3. On the Configure AAA Policy catalog list page, click **Add**.
- 4. Define the AAA policy object:
 - a. Enter the Name as `oauth-az`.
 - b. Click the **Identity extraction** tab.
 - c. Select two Methods: **HTTP Authentication header** and **OAuth**.
 - d. Selecting **HTTP authentication header** causes the HTTP Basic Authentication Realm field to appear. Leave it at its default value of **login**.
 - e. Selecting **OAuth** causes the Registered OAuth clients list. Select the OAuth client group that you defined in a previous step: **oauth-client-group**.
 - f. Click the **Authentication** tab.
 - g. For Method, select **Use AAA information file**.
 - h. When the method is selected, the remaining fields on the tab adjust. For the AAA information file URL, use the sample **AAAInfo.xml** file in the **store:** directory.
 - i. Click the **Credential mapping** tab. Nothing needs to be entered on this tab.
 - j. Click the **Resource extraction** tab.
 - k. Select **Processing metadata**.
 - l. The Processing metadata items choice appears. Select **oauth-scope-metadata**. This option is used in the style sheets to retrieve the requested scope of the request.
 - m. Click the **Resource mapping** tab. Nothing needs to be entered on this tab.
 - n. Click the **Authorization** tab.
 - o. Select a Method of **Allow any authenticated client**.
 - p. Click the **Postprocessing** tab. Nothing needs to be done on this tab.
 - q. Click **Apply** to save the AAA policy for later use.

10.6.Define the web token service

DataPower provides a special service type, the web token service, that is designed to act as an OAuth authorization server and token endpoint. In this section, you configure a specific implementation of a web token service.

- ___ 1. On the **Control Panel**, enter `web_tok` in the search field above the navigation bar.
- ___ 2. Click **New Web Token Service**.
- ___ 3. On the Create a Web Token Service page, enter the Web Token Service Name as `oauth-wts`.
- ___ 4. Click **Next**.
- ___ 5. On the next page, define the front side access to the service:
 - ___ a. For the IP, use `<dp_public_ip>`.
 - ___ b. For Port, enter `<oauth_wts_port>`.
 - ___ c. For the SSL Proxy Profile, select the profile that you built in the earlier step: `oauth-ssl-server`.
 - ___ d. Click the **Add** icon.

| Source Addresses | | | | |
|--------------------------------------|------------------------|------------------------|-------------------------------------|--------|
| IP | Port | SSL | SSL Proxy Profile | Action |
| dp_public_ip | 7530 | (on) | oauth-ssl-server | |
| <input type="text" value="0.0.0.0"/> | <input type="text"/> * | <input type="text"/> * | <input type="text" value="(none)"/> | |

- ___ 6. Click **Next**.
- ___ 7. On the next page, select the AAA policy that you created in an earlier step: **oauth-az**.
- ___ 8. Click **Next**.
- ___ 9. On the next page, review your changes, and click **Commit**.
- ___ 10. Click **Done**.
- ___ 11. Click **Save Config**.

10.7.Define the AAA policy for an enforcement point

In the next section, you will create the multi-protocol gateway service that acts as the enforcement point for OAuth-secured access, and passes the request to the real back-end application.

In this section, you create the AAA policy that implements the enforcement point behavior.

- ___ 1. Upload a style sheet to the **local:** directory that is used for credential mapping.
 - ___ a. On the Control Panel, start to enter “file” into the search field above the navigation bar.
 - ___ b. Click **File Management**.
 - ___ c. Select **Actions** to the right of the **local:** directory, and click **Upload Files**.
 - ___ d. Browse to <lab_files>/oauth, and select **accesstoken-check-resource.xsl**.
- ___ 2. Define the AAA policy object:
 - ___ a. On the **Control Panel**, enter aaa into the search field above the navigation bar.
 - ___ b. Click **AAA Policy**.
 - ___ c. On the Configure AAA Policy catalog list page, click **Add**.
 - ___ d. On the Configure AAA Policy page, enter the Name as `oauth-ep-rs`.
 - ___ e. Click the **Identity extraction** tab.
 - ___ f. Select the Method: **OAuth**.
 - ___ g. Selecting OAuth causes the Registered OAuth clients list. Select the OAuth client group that you defined in a previous step: **oauth-client-group**.
 - ___ h. Click the **Authentication** tab.
 - ___ i. For Method, select **Pass identity token to authorization phase**.
 - ___ j. Click the **Credential mapping** tab. Nothing needs to be entered on this tab.
 - ___ k. Click the **Resource extraction** tab.
 - ___ l. Check **URL sent by client** and **Processing metadata** choice.
 - ___ m. The Processing metadata items choice appears. Select **oauth-scope-metadata**. This option is used in the style sheets to retrieve the requested scope of the request.
 - ___ n. Click the **Resource mapping** tab.
 - ___ o. Set the Method as **Custom**.
 - ___ p. Enter a reference to the resource that you uploaded: **local:///accesstoken-check-resource.xsl**. This style sheet verifies that the URI sent by the client matches the scope that was validated as part of the OAuth checking.
 - ___ q. Click the **Authorization** tab.
 - ___ r. Select a Method of **Allow any authenticated client**.
 - ___ s. Click the **Postprocessing** tab. Nothing needs to be done on this tab.
 - ___ t. Click **Apply** to save the AAA policy for later use.

10.8.Define the enforcement point and resource server

In this section, you define a multi-protocol gateway that acts as the enforcement point for OAuth processing, and calls the actual backend application.

- ___ 1. On the **Control Panel**, enter `multi` in the search field above the navigation bar.
- ___ 2. Click **Edit Multi-Protocol Gateway**.
- ___ 3. On the Configure a Multi-Protocol Gateway catalog list page, click **Add**.
- ___ 4. On the Configure a Multi-Protocol Gateway page, enter the name as `resource-enforcement`.
- ___ 5. Enter the Default Backend URL as `http://127.0.0.1:<oauth_resource_port>`
- ___ 6. Set the Request Type as **Non-XML**.
- ___ 7. Set the Response Type as **JSON**.
- ___ 8. Configure an HTTPS front side handler.
 - ___ a. At the Front Side Protocol field, click the **+** (New) button.
 - ___ b. Select **HTTPS Front Side Handler**.
 - ___ c. For the Name, enter `ep-https`.
 - ___ d. For the Local IP Address, enter the value as `<dp_public_ip>`
 - ___ e. For the Port Number, enter `<oauth_ep_port>`
 - ___ f. Select the **GET method** check box.
 - ___ g. Accept the other default methods and versions.
 - ___ h. Select the SSL Proxy that you configured earlier: **oauth-ssl-server**.
 - ___ i. Click **Apply**.
- ___ 9. Configure the Multi-Protocol Gateway Policy:
 - ___ a. Click the **+** (New) button.
 - ___ b. Enter the Policy Name as `resource-enforcement`
 - ___ c. Click **Apply Policy**.
 - ___ d. Click **New Rule**.
 - ___ e. Set the direction as **Client to Server**.
 - ___ f. Double-click the **Match action** to configure it.
 - ___ g. Configure a Matching Rule that matches the URL **/favicon.ico**. This rule matches an HTTP request for the favicon that might show in the address bar of the browser.
 - ___ h. Drag an **Advanced action** to the rule configuration path.
 - ___ i. Configure it as a **Set Variable** action.
 - ___ j. Set the **var://service/mpgw/skip-backside** to `1`.
 - ___ k. Drag a **Results action** to the path.

- ___ l. Configure the action to pass the **INPUT** input context to the **OUTPUT** output context.
- ___ m. Click **Apply Policy** to save the rule and processing policy.
- ___ n. Click **New Rule**.
- ___ o. Set the direction as **Client to Server**.
- ___ p. Double-click the **Match action** to configure it.
- ___ q. Configure a Matching Rule that matches on all URLs.
- ___ r. Drag an **Advanced action** to the path.
- ___ s. Configure this action as a **Convert Query Params to XML** action.
- ___ t. No further configuration is needed on this action.
- ___ u. Drag a **AAA action** to the path.
- ___ v. Configure it with the AAA policy that you created earlier: **oauth-ep-rs**.
- ___ w. Verify that the AAA action has an output context of **OUTPUT**.
- ___ x. Click **Done**.
- ___ y. Click **Apply Policy**.
- ___ z. Click **New Rule**.
- ___ aa. Set the direction as **Server to Client**.
- ___ ab. Double-click the **Match action** to configure it.
- ___ ac. Configure a Matching Rule that matches on all URLs.
- ___ ad. Drag a **Results action** to the path.
- ___ ae. Configure the action to pass the **INPUT** input context to the **OUTPUT** output context.

| Direction | Actions |
|------------------|---------|
| Client to Server | ◀ ▲ ↗ |
| Client to Server | ◀ △ ⊖ |
| Server to Client | ◀ ↗ |

- ___ af. Click **Apply Policy**.
- ___ ag. Close the policy editor window.
- ___ 10. Click **Apply** save the MPGW.
- ___ 11. Click **Save Config**.

10.9.Define the back-end service

In this section, you define the MPGW that acts as the targeted back-end application. Although this back-end application does not really perform any application functions, it does return:

- The original access token that was passed to the OAuth enforcement point.
 - The Resource Owner HTTP header value that was sent in the request.
- ___ 1. On the Control Panel, enter `multi` in the search field above the navigation bar.
- ___ 2. Click **Edit Multi-Protocol Gateway**.
- ___ 3. On the Configure a Multi-Protocol Gateway catalog list page, click **Add**.
- ___ 4. On the Configure a Multi-Protocol Gateway page, enter the name as `backend-service`
- ___ 5. Specify the Type as **dynamic-backends**.
- ___ 6. Set the Request Type as **Non-XML**.
- ___ 7. Set the Response Type as **JSON**.
- ___ 8. Configure an HTTP front side handler.
 - ___ a. At the Front Side Protocol field, click the **+** (New) button.
 - ___ b. Select **HTTP Front Side Handler**.
 - ___ c. For the Name, enter `backend-http`
 - ___ d. For the Local IP Address, enter `127.0.0.1`
 - ___ e. For the Port Number, enter `<oauth_resource_port>`
 - ___ f. Select the **GET method** check box.
 - ___ g. Accept the other default methods and versions.
 - ___ h. Click **Apply**.
- ___ 9. Configure the Multi-Protocol Gateway Policy:
 - ___ a. Click the **+** (New) button.
 - ___ b. Enter the Policy Name as `backend-service`
 - ___ c. Click **Apply Policy**.
 - ___ d. Click **New Rule**.
 - ___ e. Set the direction as **Client to Server**.
 - ___ f. Double-click the **Match action** to configure it.
 - ___ g. Configure a Matching Rule that matches on all URLs.
 - ___ h. Drag a **Transform action** to the rule configuration path.

- ___ i. Configure the action by uploading the style sheet **json_response.xsl** from <lab_files>/oauth and placing it in the **local:** directory. This style sheet retrieves the access token from the Authorization HTTP header and the resource owner value from the Resource Owner HTTP header. The two retrieved values are placed in a JSONX array.
- ___ j. Drag another **Transform action** to the path.
- ___ k. Configure this action by specifying the style sheet **jsonx2json.xsl**, which is in the **store:** directory. This supplied style sheet converts a JSONX data structure into a JSON structure.
- ___ l. Drag an **Advanced action** to the rule configuration path.
- ___ m. Configure it as a **Set Variable** action.
- ___ n. Set the **var://service/mpgw/skip-backside** to **1**.
- ___ o. Drag a **Results action** to the path.
- ___ p. Configure the action to pass the context that was output from the second Transform action to the **OUTPUT** output context. It is usually “**dpvar_1**”.



- ___ q. When you change the second Transform Action out parameter, the Set Variable input parameter is automatically reconfigured. Configure the input of the Set Variable action to **INPUT**.
- ___ r. Click **Apply Policy**.
- ___ s. Close the policy editor window.
- ___ 10. Click **Apply** to save the MPGW.
- ___ 11. Click **Save Config**.

10.10. Test the solution

In this section, you act as the resource owner, acting through the browser as the user agent, and as part of the OAuth client.

- 1. In the browser, enter the URL to the initialization HTML in the web server:

`http://<studentnn_image_ip>/oauth/code.html`



Warning

Be sure to use the image IP address rather than 127.0.0.1 or localhost, otherwise the cookies that are used in the PHP code do not work correctly.

- 2. In the returned page, enter information that is usually coded within the OAuth client. In this step, you are providing configuration values for the client:

- The endpoint of the web token service:

`https://<dp_public_ip>:<oauth_wts_port>/authz`

- The client ID, which is the OAuth Client object name: **my-oauth-client**

- The redirect URI to the client (PHP server):

`https://<studentnn_image_ip>/oauth/code.php`

- The scope as it is defined in the OAuth client object: **/getAccountInfo**

- The client state, which can be any text string

172.16.80.57/oauth/code.html

WebSphere Console WSRR LDAP HTTP Server

Enter the information to initialize the OAuth client parameters

Enter the url of the AZ server/token endpoint (web token service)

endpoint:

Enter the OAuth client_id that is passed to the web token service

client_id:

Enter the redirect_uri back to the OAuth client (PHP server)

redirect_uri:

Enter the scopes of any resource requests

scope:

Enter an arbitrary state of the OAuth client

state:

The endpoint URL is used by the client to find the web token service. The client ID is sent by the client to identify itself. The redirect URI is sent by the client to the web token service so that the web token service can send the HTTP redirect back to the client. The scope is sent by the client to the web token service so that it can be compared to the scopes that are defined in the client object.

3. Click **Submit**. This operation takes the place of the resource owner requesting a resource.

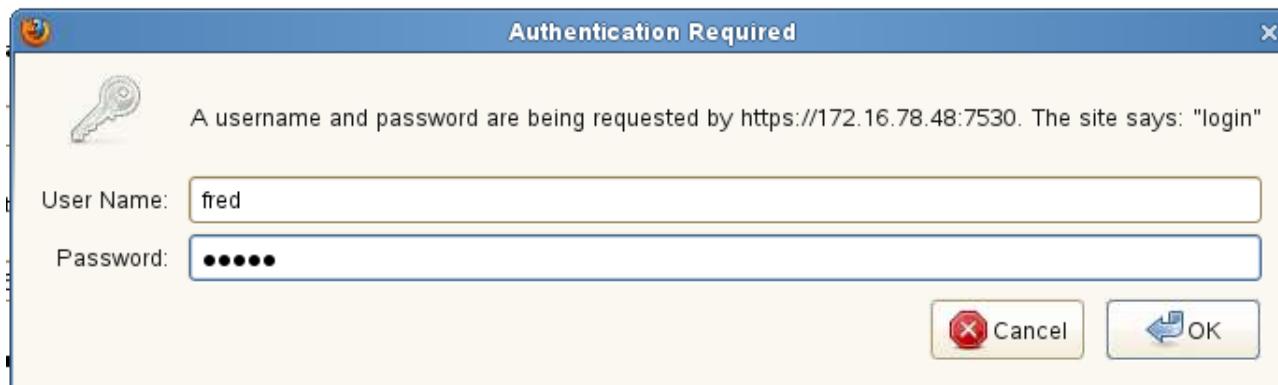


Note

Accept any requests by the browser to accept the SSL certificates. HTTPS is used in several OAuth interactions.

- ___ 4. The client code saves the parameters, and then responds with an HTTP redirection. The redirection is to the web token service endpoint.
- ___ 5. Recall that the AAA policy in the web token service specifies the HTTP basic authorization for the Extract Identity phase. Because there is no userid/password in the request, the service issues a login challenge to the resource owner. Notice that the client is **not** involved in the login. Enter a User Name of fred and a Password of smith.

8.48:7530/authz



- ___ 6. Click **OK**.
- ___ 7. The Authentication phase of the AAA policy specifies the AAAInfo.xml file in the **store:** directory. The XML file contains an entry for **fred/smith**, with an output credential of **admin**. This output credential appears in later steps.

8. Now that you are authenticated to the web token service, it now presents the Permissions dialog box where you, as the resource owner, can grant access. The client has identified itself to the web token service, hence, it can look in the specific client object to find the style sheet that generates the Permissions dialog box. Recall that you entered an authorization form of **store:///OAuth-Generate-HTML.xsl** in your OAuth client object. The web token service runs this style sheet to generate the dialog box.



Request for Permission

Welcome fred

Example Inc. is requesting your permission to access : **/getAccountInfo**.
Select "Allow Access" to grant the request or "No Thanks" to reject it.

Clicking Submit will redirect you to <https://172.16.80.57/oauth/code.php>.

Please contact support@example.com if you experience any issues using this form.

Share with Example Inc. (to utilize this, see comment in the **store:///OAuth-Generate-HTML.xsl**):
Choosing "Allow Access" with no scope selected, DataPower (by default) will treat it as though the resource owner has
 /getAccountInfo

Allow access No thanks

The sample style sheet displays some of the information it has available to it, such as, the user name, requested scope, and URL of the client.

9. Select the "/getAccountInfo" scope, select "Allow access", and click **Submit**. Again, notice that the client is not involved in this interaction; it is strictly between the authorization server and the resource owner.
10. The web token service responds to the client with its authorization grant response.

- ___ 11. Now you act as the client code, not as a resource owner. The client code sends a page to the browser to show what information it has. The new information is the authorization grant code. You enter another piece of client internal knowledge, the client secret of **clientsecretpassword**. The scope field is prefilled with a scope of **/getAccountInfo**. Click Submit.



Authorization Code Grant Example Client

Received an authorization code, now for getting an access_token

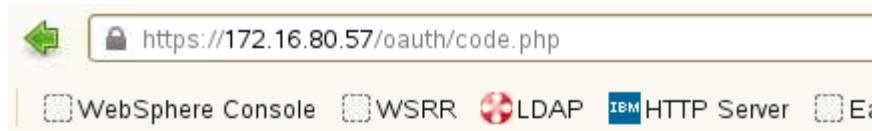
```
code :  
AAKRrvg1QfsHO+NmJd+zGz+5PgP4pWCyLf8FEO1ScuMqF+b5CBnEx/Z  
client_id: my-oauth-client  
redirect_uri: https://172.16.80.57/oauth/code.php  
endpoint: https://172.16.78.48:7530/authz
```

client_secret: **clientsecretpassword**
scope: **/getAccountInfo**
Submit

- ___ 12. The client uses the information to send an access token request to the web token service.
- ___ 13. The web token service validates the grant code, uses the client secret to validate the client ID, compares the scope, and compares the redirect URI to verify that it is talking to the same client entry point.
- ___ 14. The web token service sends the access token response to the client, passing the access token.
- ___ 15. The client sends a page to the browser that displays some of its information. The new pieces are the actual bearer access token, and the token lifetime that was sent from the web token service.

- ___ 16. You act as the client to send the resource request and the access token to the resource server. Enter the endpoint URI of the resource server:

`https://<dp_public_ip>:<oauth_ep_port>/getAccountInfo`



Authorization Code Grant

Authorization Code Grant Example Client

Received an access_token, now for getting resource

```
client_id: my-oauth-client
access_token :
AAEPbXktb2F1dGgtY2xpZW50xw7LkqCUZUo1Nl438Usqu85Ztv.
token_type: bearer
expires_in: 3600
scope: /getAccountInfo
```

resource_endpoint: `https://172.16.78.48:7533/getAccountInfo`

- ___ 17. Click **Submit**.
- ___ 18. The client uses the resource endpoint to call the resource server, and passes the access token. The resource server validates the access token and the requested resource. As part of the AAA policy processing policy, it runs **add-processing.xsl** that is specified in the client object as an extra OAuth process. This style sheet puts the resource owner value in a Resource Owner HTTP header if it is running during resource request processing. If you look at the HTTP headers, you can see the header. It then passes the resource request to the actual back-end application.
- ___ 19. The backend application, in this example, returns the access token and the output credential ("admin") as a JSON structure.
- ___ 20. The resource server returns the JSON structure to the client.

- ___ 21. In this example, the client sends a page that contains the JSON structure.

A screenshot of a web browser window. The address bar shows the URL <https://172.16.80.57/oauth/code.php>. Below the address bar is a navigation bar with icons for back, forward, search, and refresh. To the right of the address bar are several tabs: "WebSphere Console", "WSRR", "LDAP", "HTTP Server", and "E". The main content area of the browser displays the following JSON data:

```
raw result : [ { "id":0, "name":"admin",  
"access_token":"AAEPbXktb2F1dGgtY2xpZW50xw7LkqCUZUo1I  
} ]
```

Authorization Code Grant Example Client

Response from access resource :

```
raw result : [ { "id":0, "name":"admin",  
"access_token":"AAEPbXktb2F1dGgtY2xpZW50xw7LkqCUZUo1I  
} ]
```



Optional

Enable the probes for the MPGWs and the web token service. Rerun the `code.html` to drive the application again. Examine the probes and log to see the details of the interactions.

End of exercise

Exercise review and wrap-up

In this exercise, you defined an OAuth client profile that represented an OAuth client. You specified this client profile as a member of an OAuth client group that is referenced in AAA policies. You created a AAA policy that specifies HTTP basic authorization and OAuth, which you added to a web token service. You created another AAA policy that specifies OAuth to validate an access token, which you used in a resource server. Finally, you tested the configuration by using a browser to start a client running under a web server.

Exercise 11.Using a DataPower pattern

What this exercise is about

In this exercise, you play the role of a pattern deployer. You specify the values for the exposed points of variability (POV) in a specific pattern, and deploy the pattern into a new generated service.

What you should be able to do

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

- Use the DataPower Patterns Console
- Import a pattern
- Specify the values for the points of variability in the pattern
- Deploy the pattern into a generated service

Introduction

A pattern creator created a pattern that generates a multi-protocol gateway (MPGW) that secures access to the EastAddressSearch web service. The security is provided by a AAA policy that uses LDAP to store authentication and authorization information. In this exercise, you generate a service to implement this behavior. You import the pattern into your student domain. You then use the Patterns Console to “deploy” the pattern into a generated service, specifying the values that are exposed by the pattern creator. These values “customize” the pattern into a generated service for your specific situation. You test the deployment by using cURL to invoke a web service.

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

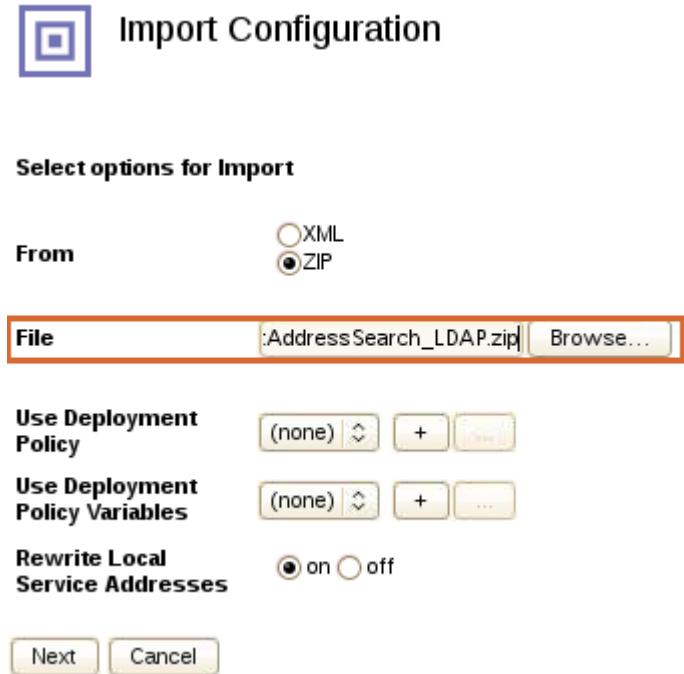
Preface

- Complete the steps in “Exercise 1: Exercise setup” before starting this lab.
- 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)
 - *<mpgw_patterns_port>*: *7nn6*, port number for the EastAddressSearch web service that is generated from a pattern

11.1. Import a pattern into your application domain

Before deploying the pattern, you must first import it.

- ___ 1. Log in to your student domain **studentnn_domain** on the appliance.
- ___ 2. Go to **Administration > Configuration > Import Configuration**.
- ___ 3. For the **File** field, go to **<lab_files>/Patterns**, and select the file **Pattern MPGW_EastAddressSearch_LDAP.zip**.



- ___ 4. Click **Next**.
- ___ 5. The Import Configuration page shows the files that are imported from the .zip file.

The following configuration is new:

Deployment Policy:MPGW_EastAddressSearch_1386614684780
 Pattern:MPGW_EastAddressSearch_1386614684780

The following files are new:

config:///Exemplar MPGW_EastAddressSearch_1386614684780.zip

- The “Exemplar” file is placed in the **config:** directory. It is an exported version of the original source service.
- The “Pattern” configuration contains the specifics of how this pattern must be deployed, and what values are exposed.

- The “Deployment Policy” configuration is the standard DataPower deployment policy object that manages the actual value substitutions in the source configuration to make it the generated service.
- 6. Click **Import**.
- 7. The page should show a successful import. Click **Done**.



Information

Normally, a pattern is imported by using the Patterns Console. This technique to import the pattern is used to allow you to see the constituents of the imported .zip file. The underlying files are not shown when importing a pattern from within the Patterns Console.

The pattern is now available in the Patterns Console.

11.2. Deploy a pattern

- 1. Above the navigation bar, click **Pattern Console**.



- 2. This option opens another tab or window that displays the Pattern Console GUI. Click **Deploy a Pattern**.

The screenshot shows the DataPower Pattern Console GUI. At the top, there is a navigation bar with tabs: "Get Started" (which is selected), "Endpoints", and "Patterns". Below the navigation bar, the page title is "Get started".

The page contains two main sections:

- Create a Pattern**: It features a blue pencil icon and the text "Create a pattern from a service in the domain." followed by a "Learn more" link.
- Deploy a Pattern**: It features a blue arrow pointing right over a grid of squares icon and the text "Create a service from a pattern." followed by a "Learn more" link.

3. This option takes you to the **Patterns** tab. You should see the pattern that you previously imported listed on the left.

The screenshot shows the DataPower Patterns interface. At the top, there are three tabs: 'Get Started', 'Endpoints', and 'Patterns'. The 'Patterns' tab is highlighted with an orange border. Below the tabs, there is a 'New Pattern...' button, a search bar with dropdown menus for 'All Categories' and 'Search for patterns', and a list of imported patterns. One pattern, 'MPGW_EastAddressSearch_LDAP', is selected and highlighted with an orange border. To the right of the pattern list is a 'Documentation:' section containing a brief description of the pattern's purpose. At the bottom right of the interface is a 'Deploy...' button, also highlighted with an orange border.

4. Select the imported pattern **MPGW_EastAddressSearch_LDAP**.
5. Read the documentation pane.
6. Click **Deploy**.
7. On the presented Deploy Pattern window, enter:
- Service name: EastAddressSearchFromPattern
 - Description: My service generated from a pattern
 - Destination URL:
`http://<backend_server_ip>:9080/EastAddress/services/AddressSearch`
 - HTTP Connection - IP address: <dp_public_ip>
 - HTTP Connection - Port: <mpgw_patterns_port>
 - Authenticate with LDAP - Host: <backend_server_ip>
 - Authenticate with LDAP - Port: 389
 - Authenticate with LDAP - LDAP bind DN: cn=root
 - Authenticate with LDAP - LDAP bind password: websphere
 - Authorize with LDAP - Host: <backend_server_ip>
 - Authorize with LDAP - Port: 389
 - Authorize with LDAP - Search scope: Subtree
 - Authorize with LDAP - LDAP bind DN: cn=root
 - Authorize with LDAP - LDAP bind password: websphere



Information

The fields that are listed in the deployment wizard are the “points of variability” that the pattern creator decided to expose to a pattern deployer.

- ___ 8. Click Deploy Pattern.

Deploy Pattern

MPGW_EastAddressSearch_LDAP

* Port:

LDAP bind DN:

LDAP bind password:

Step 5: Authorize with LDAP

This step is generated from the AddressLDAP configuration of the source service.

Host:

* Port:

Search scope:

LDAP bind DN:

LDAP bind password:

Deploy Pattern Cancel

- ___ 9. The service is generated. The Deploy Pattern wizard closes. A temporary pop-up window states that the service was generated successfully.
- ___ 10. Switch back to the WebGUI tab in the browser.
- ___ 11. On the Control Panel, click the **Multi-Protocol Gateway** icon to get the catalog list of the MPGW services in the student domain.

- 12. The newly generated service should appear in the list.

Configure Multi-Protocol Gateway

| Multi-Protocol Gateway Name | Op-State | Logs | Type | Re |
|------------------------------|----------|------|----------------|----|
| EastAddressSearchFromPattern | up | | Static Backend | SC |

Add

- 13. Select the service.

- 14. On the Configure Multi-Protocol Gateway page, notice the following points:

- The Summary field contains the text that you entered the Description field in the Deploy Pattern wizard.
- The name of the Multi-Protocol Gateway Policy is the name of the policy object from the source service (EastAddressSearch), prefixed by the generated MPGW name.
- The Default Backend URL field contains the URL string that you entered in the wizard.
- The Front Side Protocol handler object gets a unique name that uses the same naming pattern that was used for the service policy.

Configure Multi-Protocol Gateway

General Advanced Subscriptions Policy SLA Policy Details Stylesheet Params Headers

Apply

Cancel

Delete

Export | View Log | View Status | S

Multi-Protocol Gateway status: [up]

General Configuration

Multi-Protocol Gateway Name

EastAddressSearchFromPattern *

Summary

My generated service

Type

dynamic-backends *
 static-backend *

XML Manager

default

Multi-Protocol Gateway Policy

EastAddressSearchFromPattern_L

URL Rewrite Policy

(none)

Back side settings

Front side settings

- 15. Open the front side handler object and notice that it uses the HTTP Connection - Port value from the wizard.

- ___ 16. Open the AAA Policy object that is within the AAA action of the service policy. Notice that it uses the values that you entered in the wizard to specify the LDAP connection information.
- ___ 17. Exit the MPGW service.
- ___ 18. Save the configuration.

11.3. Test the generated service

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. Open a Terminal window.
- 2. Change to the Patterns directory in <lab_files>.

```
cd <lab_files>/Patterns
```

- 3. Use cURL to test the generated service.

- a. Enter the following cURL command:

```
curl --data-binary @retrieve-all.xml  
http://<dp_public_ip>:<mpgw_patterns_port>/EastAddressSearch -u  
student: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.

- b. Verify that you get the correct response: East Coast addresses (MA, GA, NC, PA, OH, NY, FL, WI) and people.
 - c. 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 12.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

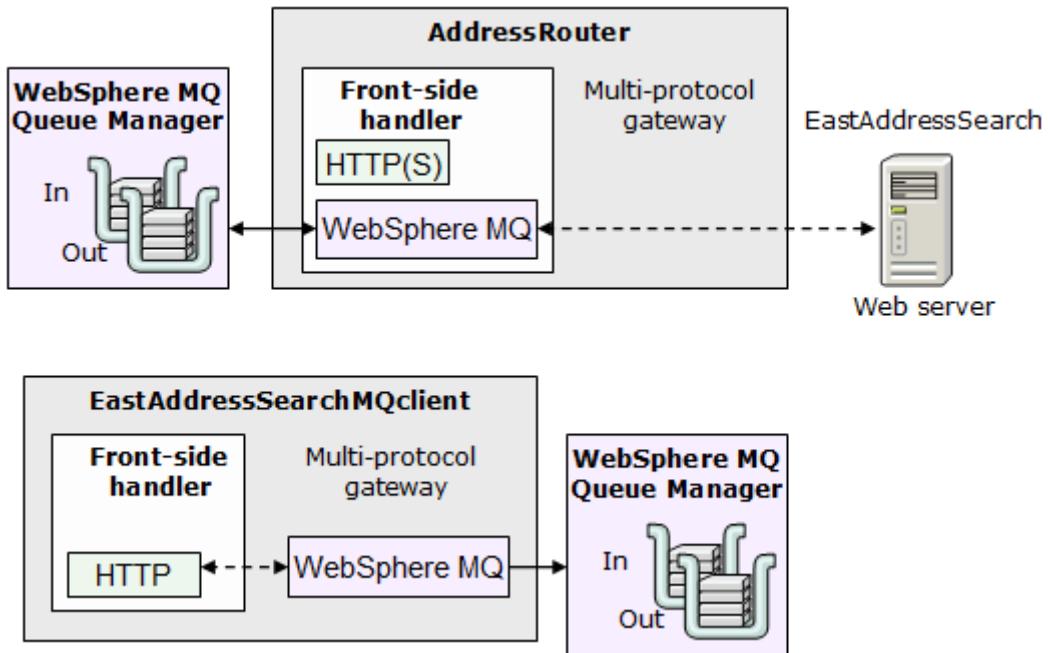
Introduction

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:
 - *<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 server running WebSphere MQ
 - *<mpgw_mq_client_port>*: multi-protocol gateway port for sending requests to WebSphere MQ: 8nn0

12.1. 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.

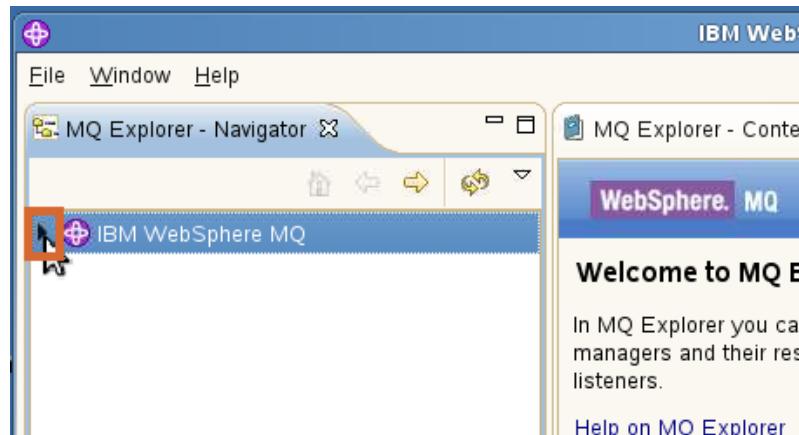
12.2.Verify MQ install and Availability

Open WebSphere MQ Explorer to confirm that the MQ environment is installed and available. Check the queue manager, listener, and channel.

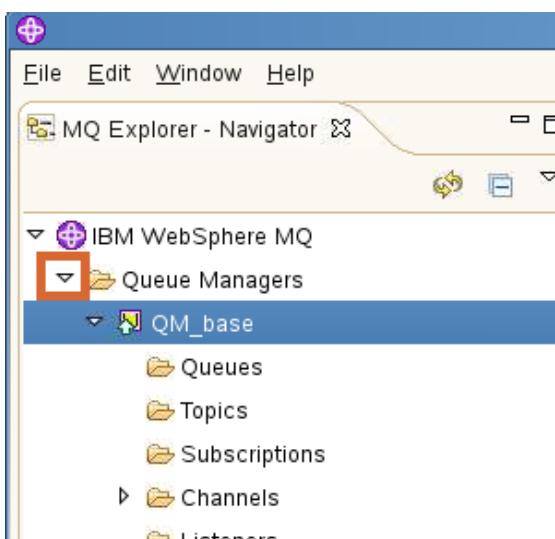
- __ 1. Confirm that the queue manager is installed and running.
 - __ a. Click the WebSphere MQ Explorer icon on the desktop.



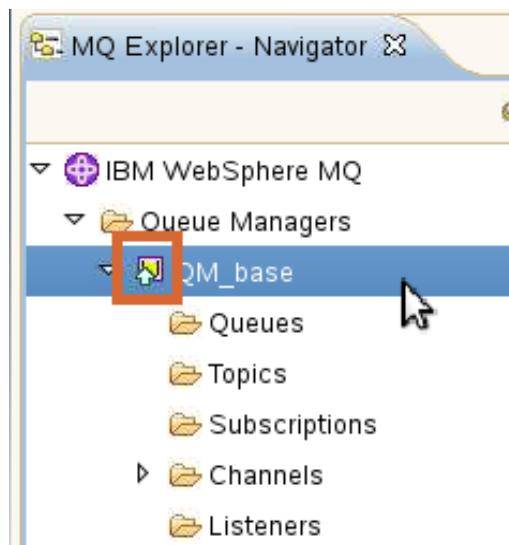
- __ b. Expand IBM WebSphere MQ Explorer.



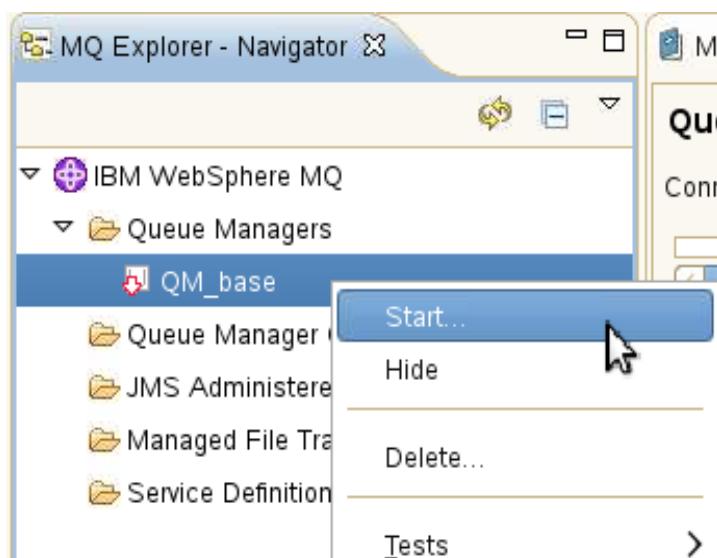
- __ c. Expand Queue Managers.



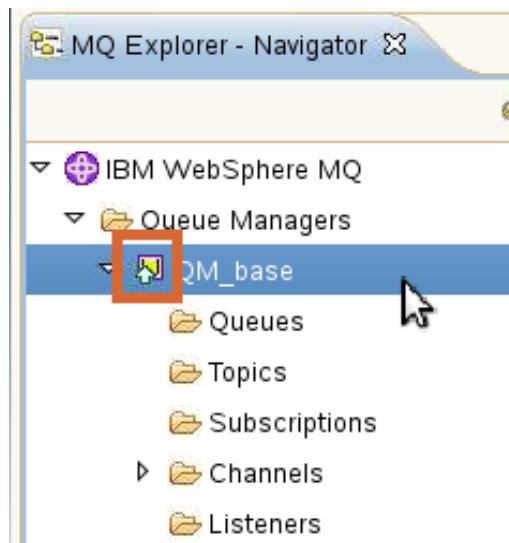
- __ d. The queue manager QM_base should have a green arrow pointing up next to the QM_base name. The green arrow means that the queue managers is up and running. If there is a red down arrow, the queue manager needs to be started.



- __ e. If required, start the queue manager. Right-click the **QM_base** queue manager and click **Start**.

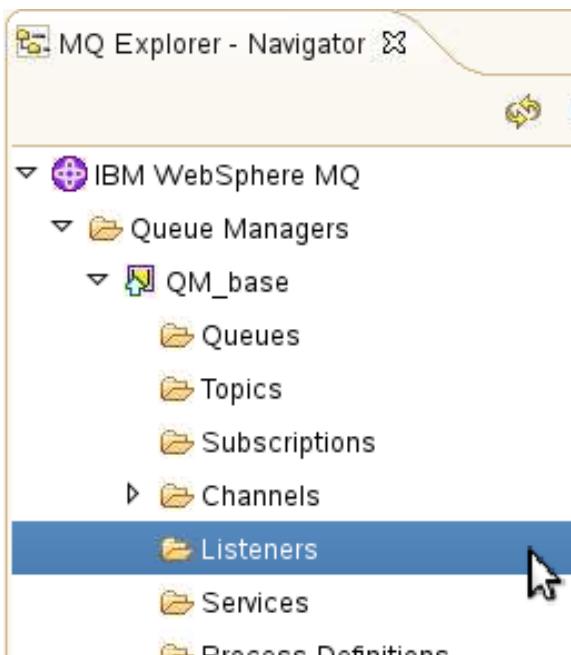


- ___ f. The queue manager is now started. This information is confirmed by the green arrow next to the queue manager name.



- ___ 2. Confirm that the queue listener is installed and running.

- ___ a. Click **Listeners**.



- ___ b. The Listeners now appear in the listener window that is located on the right side of the screen.

| Listener name | Control | Listener status | Xmit |
|----------------------|---------|-----------------|------|
| EASTADDRESS.LISTENER | Manual | Stopped | TCP |

- ___ c. Confirm a green up arrow appears to the left of the EASTADDRESS.LISTENER. If not, the listener must be started.
___ d. If required, start the listener by right-clicking EASTADDRESS.LISTENER and then click **Start**.

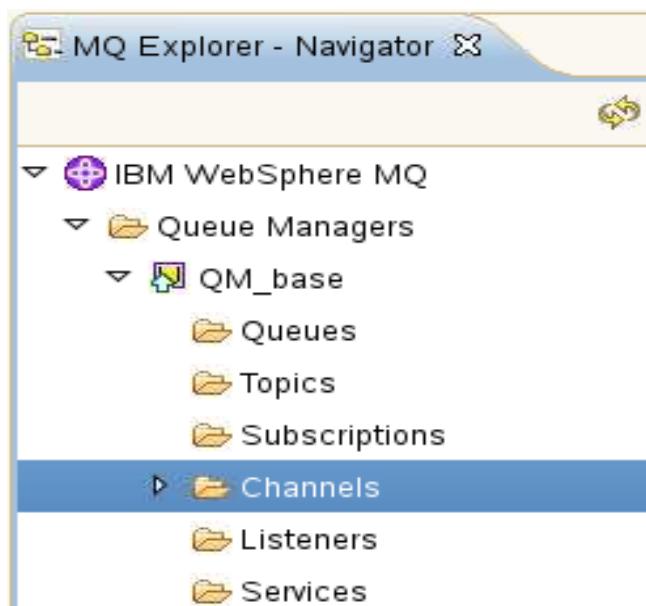
The screenshot shows the context menu for the EASTADDRESS.LISTENER entry in the listeners list. The 'Start' button is highlighted with a mouse cursor.

- ___ e. Click **Okay** to close the listener start confirmation page.
___ f. The listener is now started. This information is confirmed by the green arrow next to the queue manager name.

| Listener name | Control | Listener status | Xmit protocol | Port |
|----------------------|---------|-----------------|---------------|------|
| EASTADDRESS.LISTENER | Manual | Running | TCP | 1414 |

___ 3. Confirm that the channel is installed.

___ a. Click **Channels**.



___ b. The Channels now appears in the channel window that is located on the right side of the screen.

| Channels | | |
|-------------------------------|-------------------|---------------|
| Filter: Standard for Channels | | |
| Channel name | Channel type | Overall chanr |
| EASTADDRESS.CHANNEL | Server-connection | Inactive |

___ c. Confirm that the Channel is down.

The channel remains inactive until the first message is sent to the queue manager. After you define the queue manager on DataPower, a connection is established and the channel is then up. If your DataPower queue manager connection does not come up, a good place to start debugging is to return here and confirm that the channel is up.

12.3.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 MQ front side handler window.

- a. In the EastAddressSearch configuration page, under “Front side settings,” click **+** (new) and select **MQ Front Side Handler**.
- b. Enter the name as: `EastAddressMQFSH`
- c. In the “WebSphere MQ Front Side Handler” window, click **+** (new) to create a WebSphere MQ queue manager object.
- d. Select **Create a 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 “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 |
| General | |
| Administrative State | <input checked="" type="radio"/> enabled <input type="radio"/> disabled |
| Comments | <input type="text"/> |
| Queue Manager | EastAddressQM <input type="button" value="..."/> <input type="button" value="+"/> * |
| Get Queue | EASTADDRESSQUEUEIN01 |
| Put Queue | EASTADDRESSQUEUEOUT01 |
| The number of concurrent MQ connections | <input type="text" value="1"/> |
| Get Message Options | <input type="text" value="4097"/> |
| Polling Interval | <input type="text" value="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 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 finish testing.

12.4. 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 that is named `EastAddressQM` is already defined.

To invoke this service to **get** a message from the queue `EASTADDRESSQUEUEOUT01`, a client enters the following cURL command:

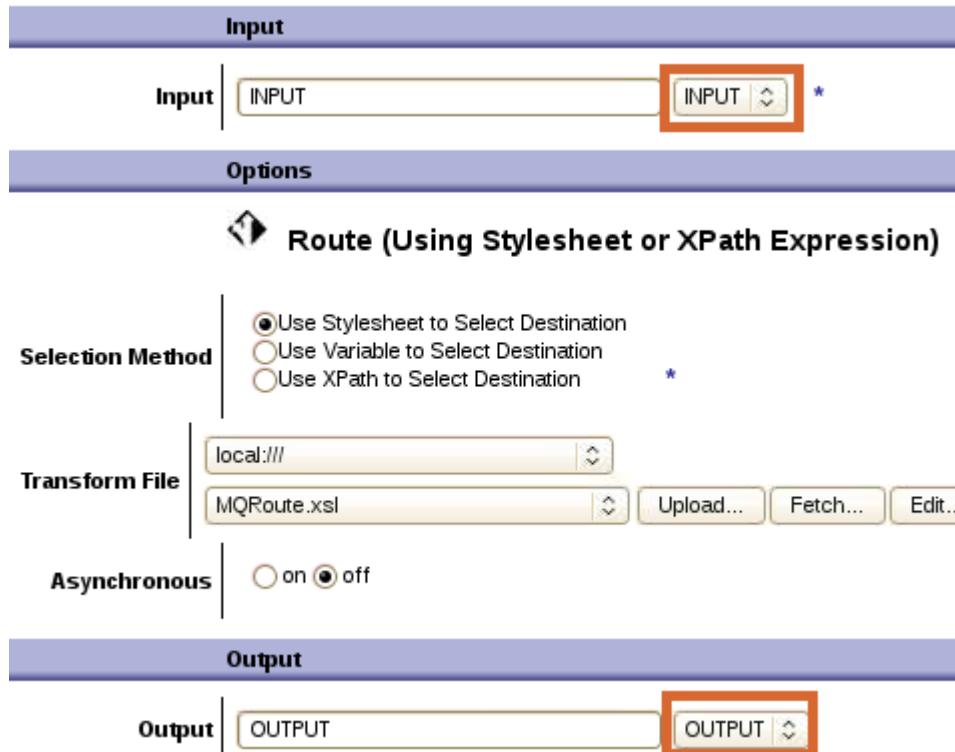
```
curl -H "Operation: GET" -H "Queue: EASTADDRESSQUEUEOUT01" -H "Content-Type: text/xml" -data-binary @[fileName] http://<dp_public_ip>:<mpgw_mq_client_port>/uri
```

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 **Apply Policy**.
- 7. Click **New Rule**.
- 8. Double-click the **Match** action that was created.

- ___ 9. Select an existing match all rule or create a matching rule that matches all URLs (use steps from a previous exercise as a guide).
- ___ 10. Add a **Route** action to the pipeline in the service policy editor.
- ___ 11. Double-click the **Route** action. Make sure that **Use Stylesheet to Select Destination** is selected as the **Selection Method**.
- ___ 12. Upload the XSL style sheet `MQRoute.xsl` from the `<lab_files>/MQ` directory.



- ___ 13. Set **Input** to **INPUT**.
- ___ 14. Set **Output** to **OUTPUT**.
- ___ 15. Click **Done**.
- ___ 16. Add a **Results** action after the route action.
- ___ 17. For **Rule Direction**, select **Client to Server**. This policy contains a request rule only.
- ___ 18. Click **Apply Policy**, and then **Close Window**.
- ___ 19. In the service window, change the **Type** from **static-backend** to **dynamic-backends**.



Note

Because the service policy contains a **Route** action that determines the back-end URL, you change the service type to **dynamic-backends**.

- ___ 20. Scroll down and select the **Response Type** of **Pass-Thru**.
This setting specifies that no response rule is run for the response message.

- ___ 21. Create an HTTP front side handler that is called `EastAddressSearchHTTPClient` and add it to the `EastAddressSearchMQclient` gateway.
 - ___ a. Under “Front side settings,” click + (new) and select **HTTP Front Side Handler**.
 - ___ b. Enter the name `EastAddressSearchHTTPClient` and port number `<mpgw_mq_client_port>`.
 - ___ c. Click **Apply**. The new handler is now in the Front Side Protocol list.
 - ___ d. Click **Apply**.
- ___ 22. Test the MPGW WebSphere MQ client by using cURL to put a message on the `AddressQueueIn<nn>` queue.
 - ___ a. Open a terminal window and go to the `<lab_files>/MQ` directory.
 - ___ b. 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>
```



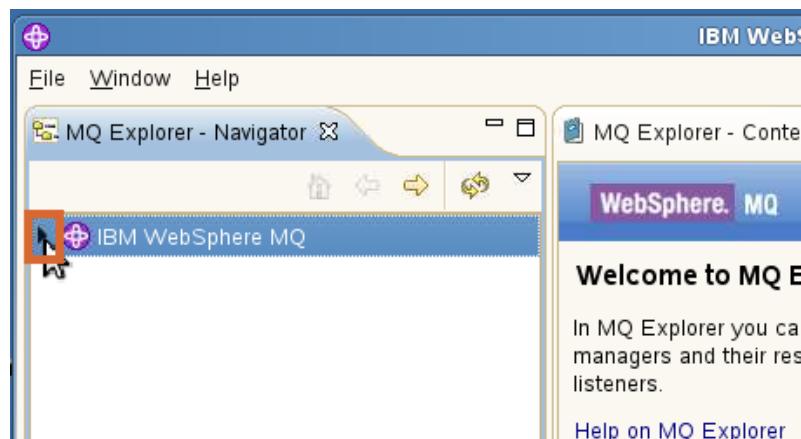
Note

This command puts the `findByLocation.xml` file onto the `EASTADDRESSQUEUEIN<nn>` without waiting for a reply. Therefore, the command finishes execution. Nothing is returned.

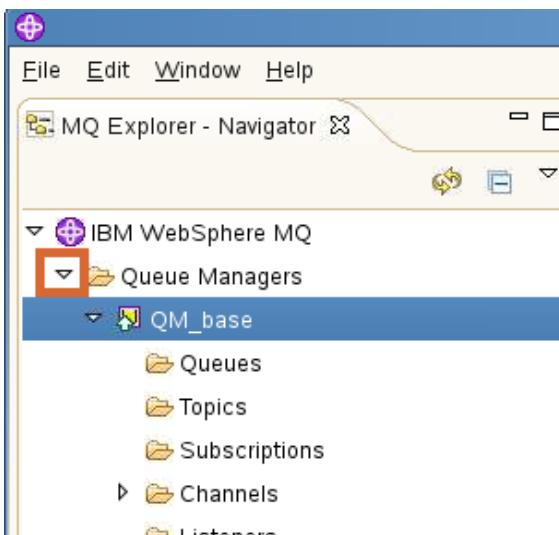
- ___ c. Check the system logs to make sure that no errors were generated after running the command.
- ___ 23. Open WebSphere MQ Explorer to view the number of messages that exist on the different queues.
 - ___ a. If MQ Explorer is not already running, click the WebSphere MQ Explorer icon on the desktop.



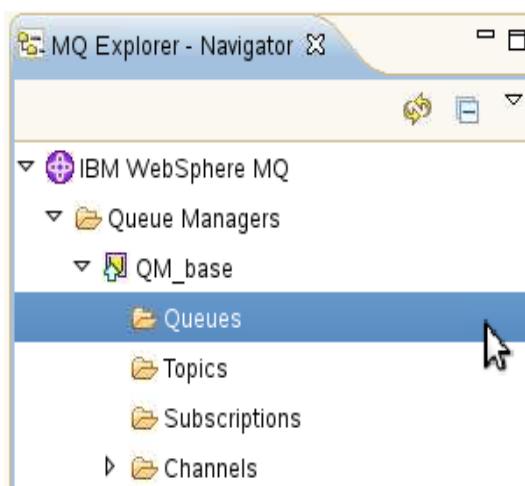
- __ b. Expand IBM WebSphere MQ Explorer.



- __ c. Expand Queue Managers.



- __ d. Click the **Queues** folder that is located under the remote QM_base to see all of the queue.



- ___ e. 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.

| 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 |
| EASTADDRESSQUEUEIN01 | Local | 0 | 1 | 1 |
| 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 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 information 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 q1 (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 q1" commands.
 - c. When you are finished with the command processor, enter `end` to terminate the command processor.

12.5. 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 functions.

- ___ 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 entry field.

Front Side Protocol

| | |
|---|--|
| AddressSearchMPG-HTTP (HTTP Front Side Handler) | |
| AddressSearchMPG-HTTPS (HTTPS (SSL) Front Side Handler) | |
| EastAddressMQFSH (MQ Front Side Handler) [down] | |
| EastAddressMQFSH (MQ Front Side Handler) | |

- ___ 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).

12.6.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 beginning and ending `<state>` tag to `<stateBad>`.


```
<state>NC</state>
          to
          <stateBad>NC</stateBad>
```

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, go 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. Because the `EastAddressSearch` service policy generates an error, a SOAP fault is generated. However, the **Unit of Work** attribute is set to **0**, hence, 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 **Connections** 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 Requeue Queue Name** specify: EASTADDRESSQUEUEERROR<nn>

Units of Work and Backout

Units of Work

Automatic Backout on off

Backout Threshold

Backout Queue Name

- ___ g. Click **Apply**, click **Apply** again, and then close all windows. Click **Apply** in the main service configuration page.



Information

In this step, you configured support for transactions. If any of the 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. Because 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` state tab back to its correct value of `<state>NC</state>`.
- ___ 9. Verify that the `EastAddressSearch` still functions by using the `EastAddressSearchMQclient` from the terminal window to the get and put messages.
- ___ 10. Save the configuration.

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.

Exercise 13. Implementing REST services using DataPower

What this exercise is about

This exercise shows you how to use the DataPower appliance to expose web services with a REST interface. You learn how to process non-XML requests and parse incoming HTTP query parameters. You use these HTTP parameters to build a SOAP request for the back-end web service. You examine style sheets that provide logic for converting an HTTP GET method request to a SOAP request. For the SOAP response from the web service, you code a rule that converts the SOAP response to JSONx, and then to JSON.

What you should be able to do

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

- Create a service policy to parse an HTTP GET request
- Use a style sheet to build a SOAP request from HTTP query parameters
- Convert a SOAP response to a JSON-formatted data structure

Introduction

A key benefit of using a REST interface is its simplicity. Resources are described in the URL, and the action is specified from the HTTP method. For example, the following HTTP methods can perform the actions that are indicated:

- GET: Fetches a resource from the server.
- DELETE: Removes a resource from the server.
- PUT: Modifies, or overwrites, an existing resource. Also, can be used to create a resource at a specific URL.
- POST: Creates a resource in the collection.

IBM WebSphere DataPower SOA Appliances provide functions for enabling a REST interface in front of an existing web service. You can create a multi-protocol gateway service that accepts an HTTP request without any XML, and builds a SOAP request with the original request parameters. The service can also take the SOAP response and convert it to a REST-based response, such as JSON. The exercise provides several style sheets that demonstrate parsing and building SOAP requests.

Reference

Implementing a Web 2.0 RESTful facade enabled with JSON using
WebSphere DataPower SOA Appliances:

http://www.ibm.com/developerworks/websphere/library/techarticles/0912_muschett/0912_muschett.html

Requirements

- Access to the DataPower appliance
- cURL, to send requests to the DataPower appliance
- Access to the <*lab_files*> directory

Exercise instructions

Preface (optional)

- All exercises in this chapter depend on the availability of specific equipment in your classroom.
- *<dp_public_ip>*: IP address of the public services on the appliance
- *<mpgw_rest_port>*: 3nn4, port number of the multi-protocol gateway that mediates between the REST/JSON client and the web services back-end application
- *<backend_server_ip>*: The IP address for the application server hosting the web services and web application.
- *<was_server_port>*: Port number for the services that run in WebSphere Application Server (AddressSearch web services, LDAP, registry)

13.1. Compare the REST interface to the SOAP interface of the back-end web service

In this section, you compare what the back-end web service supports as a request and response with what is used for a RESTful interface. These attributes have been removed for space: *xmlns*, and *xsi*.

Currently, only GET requests are supported on the back-end web service.

findByName request and response

- 1. The SOAP request to find a person by name looks like the following XML:

```
<Envelope>
  <Body>
    <findByName>
      <name>
        <firstName>Sarah</firstName>
        <lastName>Chan</lastName>
        <title>Mrs</title>
      </name>
    </findByName>
  </Body>
</Envelope>
```

- 2. The REST request that equates to this SOAP request is an HTTP GET:

```
http://servername:port/EastAddressSearch/people/
?firstName=Sarah&lastName=Chan&title=Mrs
```

In this URL, you are requesting a member of the **people** collection. The required person is identified by the search parameters, rather than a person ID number.

- ___ 3. The SOAP response for such a query is:

```

<Envelope>
  <Header />
  <Body>
    <findByNameResponse>
      <findByNameReturn>
        <details>
          <city>Cleveland</city>
          <state>OH</state>
          <street>3661 Lincoln Avenue</street>
          <zipCode>44111</zipCode>
        </details>
        <name>
          <firstName>Sarah</firstName>
          <lastName>Chan</lastName>
          <title>Mrs</title>
        </name>
      </findByNameReturn>
    </findByNameResponse>
  </Body>
</Envelope>

```

- ___ 4. The REST response, in the JSON format, that matches the SOAP response is:

```

"Person": {
  "details": {
    "city": "Cleveland",
    "state": "OH",
    "street": "3661 Lincoln Avenue",
    "zipcode": 44111
  },
  "name": {
    "firstname": "Sarah",
    "lastname": "Chan",
    "title": "Mrs"
  }
}

```

findByLocation request and response

- 1. You can request the people that are listed in a specific location (only **state** is supported on the back-end system):

```
<Envelope>
  <Body>
    <findByLocation>
      <city></city>
      <state>NC</state>
    </findByLocation>
  </Body>
</Envelope>
```

- 2. The HTTP GET request from a RESTful perspective is:

`http://servername:port/EastAddressSearch/state/NC/people/`

- 3. The SOAP response would be:

```
<Envelope>
  <Header/>
  <Body>
    <findByLocationResponse >
      <findByLocationReturn>
        <Address>
          <details>
            <city>Charlotte</city>
            <state>NC</state>
            <street>4715 Cherry Drive</street>
            <zipCode>28212</zipCode>
          </details>
          <name>
            <firstName>Wayne</firstName>
            <lastName>Green</lastName>
            <title>Mr</title>
          </name>
        </Address>
      ... additional <Address> elements ...
      </findByLocationReturn>
    </findByLocationResponse>
  </Body>
</Envelope>
```

- ___ 4. The related REST response, that also uses JSON, is:

```

"people": {
    "person": {
        "details": {
            "city": "Charlotte",
            "state": "NC",
            "street": "4715 Cherry Drive",
            "zipcode": 28212
        },
        "name": {
            "firstname": "Wayne",
            "lastname": "Green",
            "title": "Mr"
        }
    },
    ...
    ... additional Person objects
}

```

retrieveAll request and response

- ___ 1. There is a SOAP request to return all people:

```

<Envelope>
    <Body>
        <retrieveAll />
    </Body>
</Envelope>

```

- ___ 2. The HTTP URL would look like:

`http://servername:port/EastAddressSearch/peoples`

___ 3. The SOAP response is:

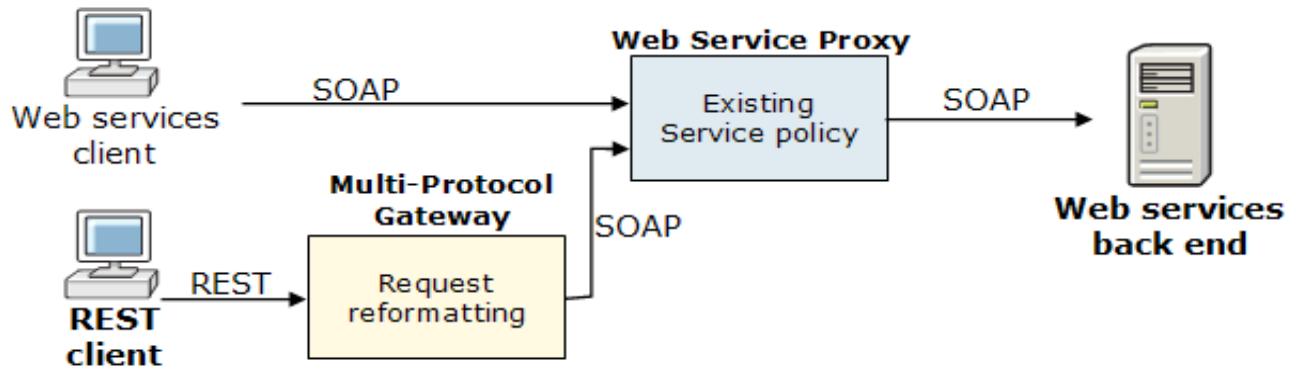
```
<Envelope>
  <Header />
  <Body>
    <retrieveAllResponse >
      <retrieveAllReturn>
        <Address>
          <details>
            <city>Philadelphia</city>
            <state>PA</state>
            <street>47153 Hickory Lane</street>
            <zipCode>19273</zipCode>
          </details>
          <name>
            <firstName>Willie</firstName>
            <lastName>Martinez</lastName>
            <title>Mr</title>
          </name>
        </Address>
      ... additional <Address> elements ...
      </retrieveAllReturn>
    </retrieveAllResponse>
  </Body>
</Envelope>
```

___ 4. The corresponding RESTful JSON response is:

```
"people": {
  "person": {
    "details": {
      "city": "Philadelphia",
      "state": "PA",
      "street": "4153 Hickory Lane",
      "zipcode": 19273
    },
    "name": {
      "firstname": "Willie",
      "lastname": "Martinez",
      "title": "Mr"
    }
  },
  ... additional Person objects
}
```

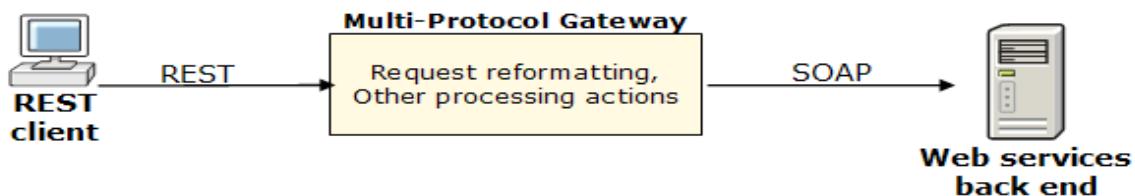
13.2.The high-level design of the REST service

If the back-end web service was already proxied by another DataPower service, such as a web service proxy or a multi-protocol gateway, the service design might look like the facade pattern that was mentioned in the presentation.



In this pattern, an existing web service proxy is already in front of the back-end web services. This web service proxy might provide services that are based on DataPower, such as authentication, routing, and transformation. To allow the REST clients to use the same services as the SOAP clients, the multi-protocol gateway service that provides the REST interface sits in front of the web service proxy. Any enhancement to the web service proxy for the SOAP client also becomes available to the REST client without any change to the REST multi-protocol gateway.

You can use a bridge pattern in a situation in which you have no existing SOAP clients for the web service and are not planning on providing a SOAP interface through the appliance.



The conversion of the REST request to a SOAP request and the response conversion, occurs within the multi-protocol gateway. Any additionally needed enhancements, such as authentication, happens within this service.

This bridge pattern is the design that is used in this exercise.

13.3.Create the request side of the RESTful multi-protocol gateway service

- __ 1. Create a multi-protocol gateway service.
- __ a. Click the **Multi-Protocol Gateway** icon.



- __ b. Click **Add** to configure a new multi-protocol gateway service.
- __ c. Modify the service with the following configuration:
 - Enter the following name: REST_EastAddressSearch
 - Use the existing **default** XML manager.
 - Set the service type to **static-backend**.

Configure Multi-Protocol Gateway

General Advanced Subscriptions Policy SLA Policy Details Stylesheet Params Headers M

Apply Cancel

General Configuration

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

- Scroll down to the **Request Type** and select **Non-XML**.
- Scroll down to the **Response Type** and select **SOAP**. The response from the back-end web service is a SOAP message.

Response Type	Request Type
<input type="radio"/> JSON	<input type="radio"/> JSON
<input type="radio"/> Non-XML	<input checked="" type="radio"/> Non-XML
<input type="radio"/> Pass-Thru	<input type="radio"/> Pass-Thru
<input checked="" type="radio"/> SOAP	<input type="radio"/> SOAP
<input type="radio"/> XML	<input type="radio"/> XML

- ___ d. Specify the **Default Backend URL** as `http://<backend_server_ip>:<was_server_port>`. If you created the Static Host **WSserver**, you can use that in place of `<backend_server_ip>`.
- ___ e. On the **Advanced** tab, set the **Process Messages Whose Body is Empty** setting to **on**, which is necessary for a RESTful interface.

Process Messages Whose Body Is Empty
 on off

- ___ 2. Back on the **General** tab, create an HTTP front-side handler.

- ___ a. Create an HTTP front-side handler object that is called `http_fsh_address_search_rest` with the port number `<mpgw_rest_port>`.
- ___ b. Make sure that you select the **POST, GET, PUT, HEAD, DELETE, and URL with Query Strings** check boxes. Click **Apply**. The handler window closes.

Name	<input type="text" value="http_fsh_address_search_rest"/>
<hr/>	
Administrative State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
<hr/>	
Comments	<input type="text"/>
<hr/>	
Local IP Address	<input type="text" value="0.0.0.0"/>
<hr/>	
Port Number	<input type="text" value="11971"/>
<hr/>	
HTTP Version to Client	<input type="button" value="HTTP 1.1"/>
<hr/>	
Allowed Methods and Versions	<input checked="" type="checkbox"/> HTTP 1.0 <input checked="" type="checkbox"/> HTTP 1.1 <input checked="" type="checkbox"/> POST method <input checked="" type="checkbox"/> GET method <input checked="" type="checkbox"/> PUT method <input checked="" type="checkbox"/> HEAD method <input type="checkbox"/> OPTIONS <input type="checkbox"/> TRACE method <input checked="" type="checkbox"/> DELETE method <input checked="" type="checkbox"/> URL with Query Strings

- ___ 3. Create the multi-protocol gateway service policy.

- ___ a. Click the plus sign (+) button in the Multi-Protocol Gateway Policy field to create a service policy.
- ___ b. Enter the following policy name: `REST2SOAP`, and click **Apply Policy**.
- ___ c. Click **New Rule**.
- ___ d. Enter the following name: `REST_GET_REQ`

- ___ e. Change the rule direction to **Client to Server**.
- ___ 4. Configure a match action that matches a URL and the HTTP method.
 - ___ a. Double-click the **Match** action icon.
 - ___ b. Create a Matching Rule.
 - ___ c. Enter the following name: `match_GET_EastAddressSearch`
 - ___ d. You now create two matching rules. For the first, click **Add**.
 - ___ e. Define a matching rule that matches on URLs of `/EastAddressSearch/*`.
 - ___ f. Click **Add** again to create another rule.
 - ___ g. Create another rule that matches on HTTP method **GET**.

Matching Rule

Matching Type	HTTP Header Tag	HTTP Value Match	URL Match	Error Code	XPath Expression	HTTP Method
URL			<code>/EastAddressSearch/*</code>			default
HTTP Method						GET

**Information**

Both of these rules must be satisfied for the message to match. Why? It is because the default setting for **Boolean or Combinations** is **off**, which indicates AND behavior.

- ___ h. Save the matching rule and Match action.
- ___ 5. Add a Convert Query Parameters action. This action restructures the URL query parameters into an XML nodeset.
 - ___ a. Drag the **Advanced** action to the rule configuration path.
 - ___ b. Double-click the icon, select **Convert Query Params to XML**, and click **Next**.
 - ___ c. Click the plus sign (+) next to the **Input Conversion** option to create an HTTP input conversion map.
 - ___ d. Enter the following name for the new map: `URL-encoded`
 - ___ e. Notice that the Default Encoding defaults to **URL-encoded**. Retain that value.
 - ___ f. Click **Apply**.
 - ___ g. Click **Done**.



Information

This action converts HTTP query parameters into an XML nodeset that can be used as input to a style sheet. For example, look at the following HTTP GET request:

```
curl -G
"http://dphost:port/EastAddressSearch/people/?firstName=Sarah&lastName=Chan&title=
Mrs"
```

It is converted to an XML nodeset that contains the following code:

```
<request>
  <url>/EastAddressSearch/people/?firstName=Victor&lastName=Collins&title=Mr</url>
  <base-url>/EastAddressSearch/people/</base-url>
  <args src="url">
    <arg name="firstName">Victor</arg>
    <arg name="lastName">Collins</arg>
    <arg name="title">Mr</arg>
  </args>
</request>
```

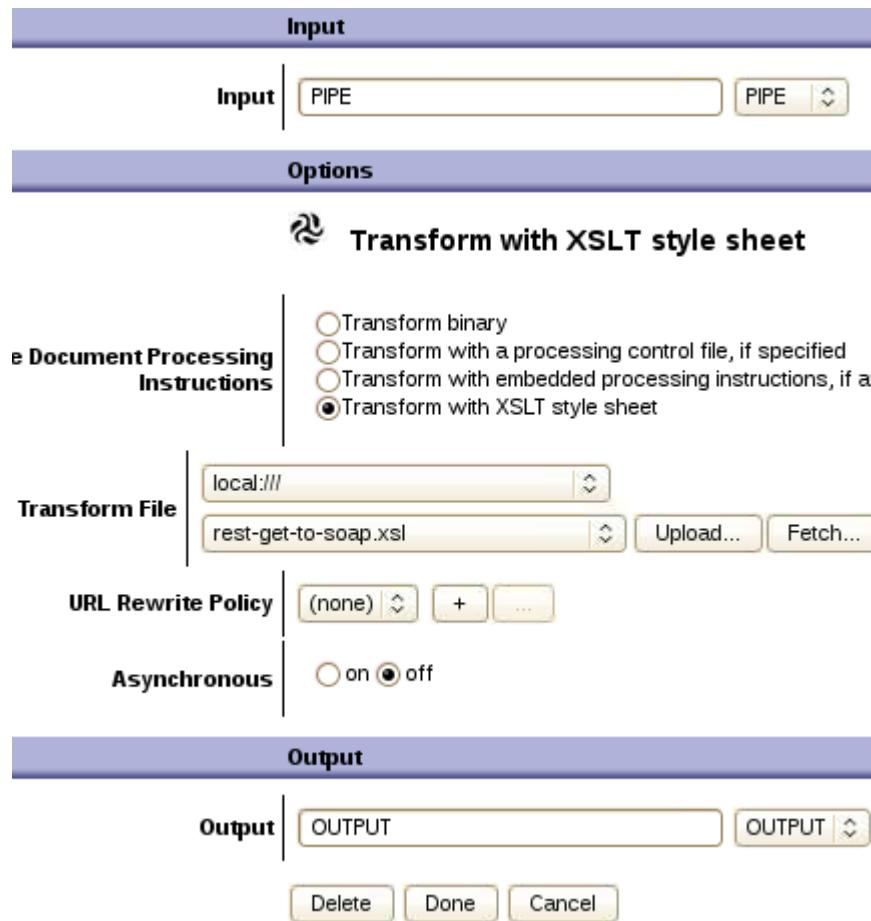
It only converts the parameters after the question mark (?). If you want the URI portion of the URL, then you need to use a DataPower service variable.

6. Add a Transform action to build the SOAP request from the HTTP input.

This style sheet extracts the HTTP method from the request and saves it to a context variable. This context variable decides the rule to invoke; each rule in the service policy is specific to the HTTP method.

- a. Add a Transform action after the Convert Query Params action.
- b. Double-click the **Transform** action to open its configuration.
- c. Make sure that **Transform with XSLT style sheet** is selected.
- d. Select **local://** as the protocol for the **Transform** source.

- __ e. Click **Upload** to load the <lab_files>/REST/rest-get-to-soap.xsl style sheet.



- __ f. Click **Done**.



Information

The rest-get-to-soap.xsl file contains the logic for transforming the HTTP GET request to a SOAP request. You need to create a style sheet for each type of HTTP method to SOAP request.

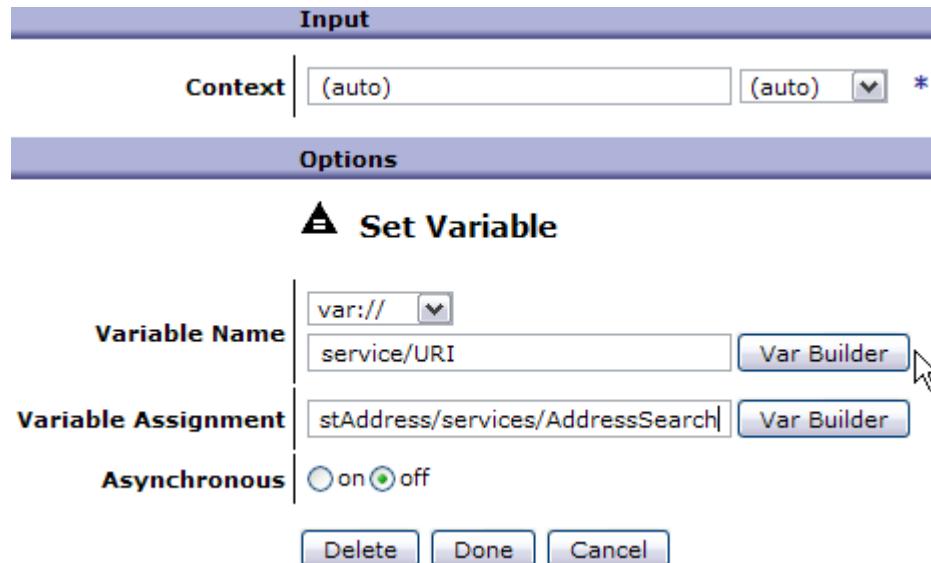
In this style sheet, the service URI is obtained and parsed to extract the parameters. The key function is the `str:tokenize()` extension function, which parses the string that is based on a delimiter and builds an array. For the `findByName` operation, the XML nodeset that are built from the action `Convert Query Params to XML` is also used.

- __ g. Click **Done**.

- __ 7. Add a Method Rewrite action to change the HTTP GET request to an HTTP POST request for the SOAP-based web service on the back-end system.

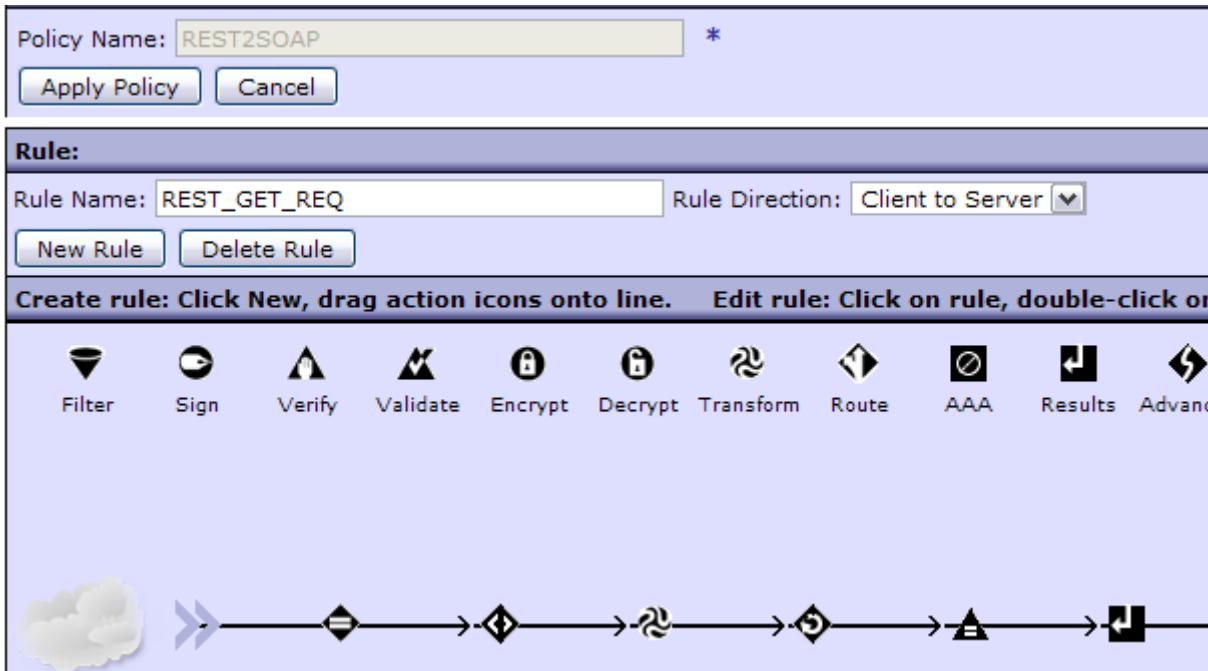
- Drag the **Advanced** action to the rule configuration path.
- Double-click the icon, select **Method Rewrite**, and click **Next**.
- Set the Method as **POST**.

- ___ d. Click **Done**.
- ___ 8. Now set a DataPower variable for the back-end service URI
- Drag the **Advanced** action to the rule configuration path.
 - Double-click the icon, select **Set Variable**, and click **Next**.
 - Use the Var Builder to select the `var://service/URI` variable.
 - Set the value to `EastAddress/services/AddressSearch`. This value is the URI expected by the back-end web service.



- ___ e. Click **Done**.
- ___ 9. Add a Results action after the Set Variable action.
- Drag a **Results** action to the rule configuration path.
 - Double-click the **Results** action to open its configuration.
 - Confirm that the input is set to `dpxvar_1`, or whatever the output context is for the Transform action. Set the output context to **OUTPUT**.
 - Click **Done**.

- ___ 10. The request rule looks like this example:



- ___ 11. The response rules still need to be created, but you can now test the request rule. Before testing the request rule, you need to create a response rule that matches on all URLs, and moves the INPUT context to the OUTPUT context. This response rule allows the response from the back-end server to flow back to the client.
- ___ a. Create a rule that is named REST_test.
 - ___ b. Specify its direction as **Server to Client**.
 - ___ c. Configure the Match action to match on all URLs.
 - ___ d. Add a Results action that moves the INPUT context to the OUTPUT context.
- ___ 12. Save your work up to this point.
- ___ a. Click **Apply Policy** and close the policy editor window.
- ___ 13. You need to specify some of the HTTP headers during your message processing. You use header injection to specify them.
- ___ a. Click the **Headers** tab.
 - ___ b. In the Header Injection Parameters section, click **Add**.
 - ___ c. On the Add a New Header Injection Parameter dialog box, add an HTTP header that is added to the request going to the back-end server. In the **Direction** choice, select **Back**. For the **Header Name** field, enter **Content-Type**. For the **Header Value** field, enter **text/xml**.
 - ___ d. Click **Submit**.
 - ___ e. Add another header going to the back-end system. In the **Direction** choice, select **Back** again. For the **Header Name** field, enter **SOAPAction**. For the **Header Value** field, enter **REST** (an arbitrary value).

**Note**

When you send a SOAP request through the cURL command, you add a SOAPAction header, but with an empty value. If you try to specify the SOAPAction header with an empty value using the **Headers** tab, it does not accept it. For this usage, it is not a problem to have the unneeded value. If a header you inject requires an empty value, you might use a Set Variable action or style sheet to define the header.

- ___ f. Add one more header for the response that is going back to the original client. For the **Direction**, select **Front**. For the **Header Name** field, enter `Content-Type`. For the **Header Value** field, enter `application/json`. This RESTful interface uses JSON for its data formatting.

Header Injection Parameters

Direction	Header Name	Header Value
Back	Content-Type	text/xml
Back	SOAPAction	REST
Front	Content-Type	application/json

Add

- ___ g. Click **Apply** to save the changes to the service definition.
 ___ h. Click **Save Config** to persist the changes.
- ___ 14. Test the HTTP GET request.
- ___ a. Open a Terminal window and navigate to the `<lab_files>/REST` directory.
- The East Address Search web service supports three operations:
- `findByName`: Searches for address information using a first name, last name, and title
 - `findByLocation`: Searches for address information using a state
 - `retrieveAll`: Retrieves all address information
- The “-G” in the cURL command indicates you want an HTTP GET request, rather than the cURL default of HTTP POST. You can insert headers into a cURL command, specifically the Accept header, to specify the response type using `-H "Accept: application/xml"`. This header specifies the return type of XML. If you omit this header, then a SOAP response is returned. Since cURL is calling a web service, it does return a SOAP response.

- __ b. Use cURL to send a request for `findByName`:

```
curl -G
"http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/people/?firstName=Sarah&
lastName=Chan&title=Mrs"
```

**Note**

The quotation marks (" ") are needed around the URL in the cURL command so that cURL includes the URI parameters.

- __ c. Verify that you obtain the correct response.

```
<p796:findByNameResponse
xmlns:p796="http://east.address.training.ibm.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <findByNameReturn>
    <details>
      <city>Cleveland</city>
      <state>OH</state>
      <street>3661 Lincoln Avenue</street>
      <zipCode>44111</zipCode>
    </details>
    <name>
      <firstName>Sarah</firstName>
      <lastName>Chan</lastName>
      <title>Mrs</title>
    </name>
  </findByNameReturn>
</p796:findByNameResponse>
```

- __ d. Use cURL to send a request for `findByLocation`:

```
curl -G
http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/state/NC/people/
```

- __ e. The correct response is a group of `<Address>` elements, all with an "NC" state address.

- __ f. Use cURL to send a request for `retrieveAll`:

```
curl -G http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/peoples/
```

- __ g. The correct response is a group of all `<Address>` elements in the application.

- __ h. Use cURL to send an invalid request ("CA" is not in the East Address web service.)

```
curl -G http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/state/CA
```

- __ i. Verify that you obtain a SOAP Fault response with an `AddressNotFoundException`.

13.4.Create the findByName response side of the RESTful multi-protocol gateway service

Since you are supporting a RESTful interface for the client, a SOAP response is not appropriate. In this section, you create the response rules to convert the SOAP responses to the appropriate JSON responses.

- ___ 1. First, delete the temporary response rule that you coded to enable the previous test.
 - ___ a. Open the multi-protocol gateway policy editor, if it is not open.
 - ___ b. In the Configured Rules section, click the **REST_test** response rule.
 - ___ c. That brings the rule onto the rule configuration path. Click **Delete Rule**.
- ___ 2. Create a response rule to convert a findByName SOAP response to a JSON-formatted response.
 - ___ a. Click **New Rule**, and enter the following name: `REST_GET_NameRESP`
 - ___ b. Set the direction as **Server to Client**.
- ___ 3. Configure a Match action to parse the SOAP response.
 - ___ a. Double-click the **Match** action to configure it.
 - ___ b. Click the plus sign (+) button to create a Matching Rule.
 - ___ c. Enter the following name for the Matching Rule: `findByNameResponse`
 - ___ d. Click the **Matching Rule** tab.
 - ___ e. Click **Add** to define a matching rule.
 - ___ f. On the Edit Matching Rule window, set the Matching Type to **XPath**.
 - ___ g. Click **XPath Tool**.
 - ___ h. On the “Build XPath Expression from sample XML File” window, set **Namespace Handling** to **local**.
 - ___ i. For the **URL of Sample XML Document** field, upload the `findByNameResponse.xml` sample SOAP response file from <lab_files>/REST/.

- __ j. In the sample file area, click <p796:findByNameResponse>. This selection indicates which element you want to check.

The screenshot shows the 'Build XPath Expression from sample XML' dialog. At the top, there's a blue header bar with the title 'Build XPath Expression from sample XML'. Below it is a purple bar labeled 'Select sample XML document'. On the left, there are two sections: 'URL of Sample XML Document' (containing 'local:///findByNameResponse.xml' and buttons for Upload..., Fetch..., Edit...) and 'Namespace Handling' (with radio buttons for local, prefix, and uri, where 'local' is selected). The main area is titled 'Selected XPath Expression' and contains the XPath '/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByNameResponse']'. At the bottom are three buttons: Refresh, Done, and Cancel. Below the dialog is a purple bar labeled 'Content of sample XML file. Click on an element, attribute name, or attribute value to see expression.' It displays the XML structure with the p796:findByNameResponse element highlighted.

- __ k. Click **Done**.
- __ l. Click **Apply** to save the XPath.
- __ m. Click **Apply** to save the Matching Rule.
- __ n. Click **Done** to save the Match action. This match inspects the message to see whether it is a findByName response.
- __ 4. Add a Transform action to create a JSONx version of the SOAP response.
- __ a. Back in the policy editor, add a Transform action after the Match action.

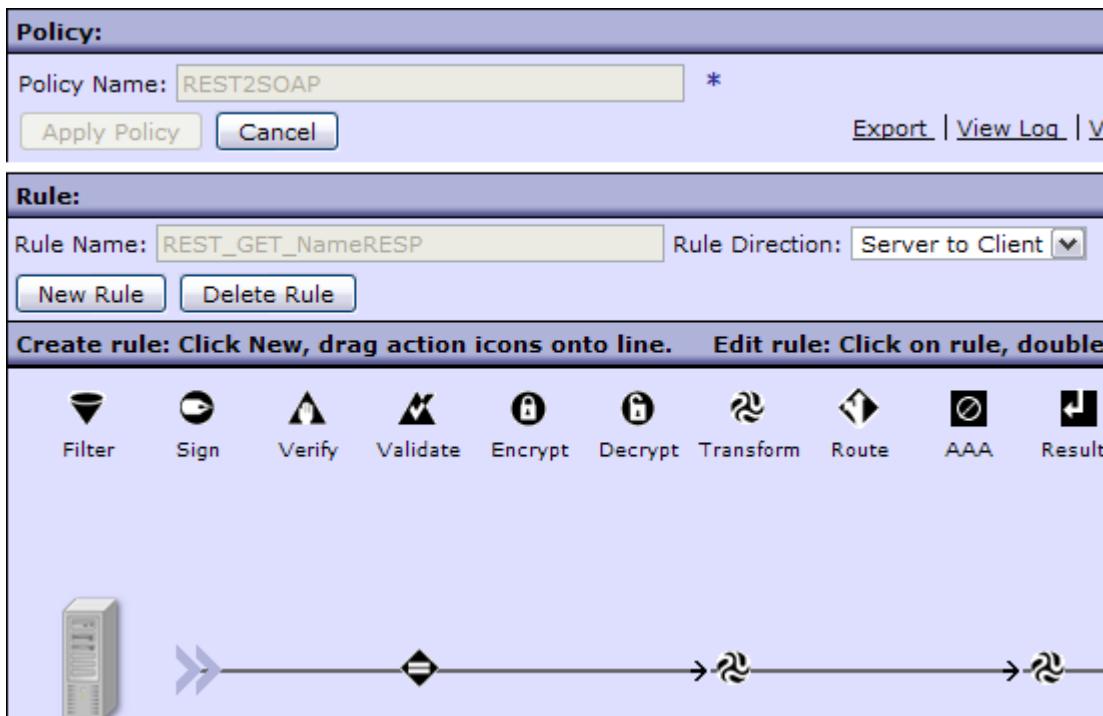
- ___ b. For the style sheet, upload `NameSOAP2JSONX.xsl` from `<lab_files>/REST`. This style sheet parses the SOAP response and builds a JSONx version of the response.
 - ___ c. Click **Done** to save the Transform.
- ___ 5. Add another Transform to reformat the JSONx response into a JSON structure.
- ___ a. Add another Transform action after the previous one.
 - ___ b. For this style sheet, use the supplied `jsonx2json.xsl` file in the `store:///` directory on the appliance.



Warning

Remember that if you want to modify any files in the `store:///` directory, you should always make a copy of the file in the `local:///` directory, and only modify the `local:///` copy. Consider files in the `store:///` directory as read-only.

- ___ c. Since this action is the last action, set the Output context to **OUTPUT**.
- ___ d. Click **Done** to save this action.
- ___ e. On the policy editor, click **Apply Policy**.
- ___ f. Click **Apply** on the Configure Multi-Protocol Gateway window, if it is enabled.



- ___ 6. Test the retrieval-by-name capability.
- ___ a. Since these REST requests are HTTP GET requests, there is no HTTP body to send on the request. You can be in any directory when you enter the cURL command.

- __ b. Enter the following cURL command to get a person record for a specific person.

```
curl -G "http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/people/?firstName=Sarah&lastName=Chan&title=Mrs"
```

- __ c. The response should look like:

```
"person":{  
  "details":{  
    "city":"Cleveland",  
    "state":"OH",  
    "street":"3661 Lincoln Avenue",  
    "zipcode":44111 },  
  "name":{  
    "firstname":"Sarah",  
    "lastname":"Chan",  
    "title":"Mrs" }  
}
```

13.5.Create the findByLocation response side of the RESTful multi-protocol gateway service

Now that you have the first response rule that is configured, the rules for the other SOAP responses are similar. What differs is the Match action and the style sheet to convert the SOAP response to JSONx.

In this section, you configure the rule to handle findByLocation responses.

- ___ 1. Create a response rule to convert a findByLocation SOAP response to a JSON-formatted response.
 - ___ a. Click **New Rule**, and enter the following name: REST_GET_LocationRESP
 - ___ b. Set the direction as **Server to Client**.
- ___ 2. Configure a Match action to parse the SOAP response.
 - ___ a. Double-click the **Match** action to configure it.
 - ___ b. Click the plus sign (+) button to create a Matching Rule.
 - ___ c. Enter the following name for the Matching Rule: findByLocationResponse
 - ___ d. Click the **Matching Rule** tab.
 - ___ e. Click **Add** to define a matching rule.
 - ___ f. On the Edit Matching Rule window, set the Matching Type to **XPath**.
 - ___ g. Click **XPath Tool**.
 - ___ h. On the “Build XPath Expression from sample XML File” window, set **Namespace Handling to local**.
 - ___ i. For the **URL of Sample XML Document** field, upload the `findByLocationResponse.xml` sample SOAP response file from <lab_files>/REST/.

- __ j. In the sample file area, click <p796:findByLocationResponse>. This selection indicates which element you want to check.

URL of Sample XML Document: local:///findByLocationResponse.xml
Namespace Handling: local

Selected XPath Expression

XPath *

```
/*[local-name()='Envelope']/*[local-name()='Body']/*[local-name()='findByLocationResponse']
```

Content of sample XML file.
 Click on an element, attribute name, or attribute value to select an XPath expression.

The XML tree view shows the following structure:

- <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>
 <findByLocationRe>n>
 <Address>
 <details>
 <city>Charlotte</city>
 </details>
 </Address>
 </findByLocationRe>n>
 </p796:findByLocationResponse>
 </soapenv:Body>
 </soapenv:Envelope>

- __ k. Click **Done**.
- __ l. Click **Apply** to save the XPath.
- __ m. Click **Apply** to save the Matching Rule.
- __ n. Click **Done** to save the Match action. This match inspects the message to see whether it is a findByLocation response.
3. Add a Transform action to create a JSONx version of the SOAP response.
- __ a. In the policy editor, add a Transform action after the Match action.
- __ b. For the style sheet, select LocationSOAP2JSONX.xsl from <lab_files>\REST\. This style sheet parses the SOAP response and builds a JSONx version of the response.
- __ c. Click **Done** to save the Transform.

-
- ___ 4. Add another Transform to reformat the JSONx response into a JSON structure.
 - ___ a. Add another Transform action after the previous one.
 - ___ b. For this style sheet, use the supplied `jsonx2json.xsl` file in the `store:///` directory on the appliance.
 - ___ c. Since this action is the last action, set the Output context to **OUTPUT**.
 - ___ d. Click **Done** to save this action.
 - ___ e. On the policy editor, click **Apply Policy**.
 - ___ f. Click **Apply** on the Configure Multi-Protocol Gateway window, if it is enabled.
 - ___ 5. Test the retrieval-by-location capability.

- ___ a. Use cURL to send a request for `findByLocation`:

```
curl -G
```

```
http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/state/NC/people/
```

- __ b. The response is a JSON-formatted list of people in NC:

```
"people":{  
    "person":{  
        "details":{  
            "city": "Charlotte",  
            "state": "NC",  
            "street": "4715 Cherry Drive",  
            "zipcode": 28212 },  
        "name":{  
            "firstname": "Wayne",  
            "lastname": "Green",  
            "title": "Mr" }  
    },  
    "person":{  
        "details":{  
            "city": "Charlotte",  
            "state": "NC",  
            "street": "2649 First Lane",  
            "zipcode": 28206 },  
        "name":{  
            "firstname": "Tina",  
            "lastname": "Cox",  
            "title": "Mrs" }  
    },  
    "person":{  
        ...  
    },  
    "person":{  
        ...  
    },  
    "person":{  
        ...  
    },  
}
```

13.6.Create the retrieveAll response side of the RESTful multi-protocol gateway service

In this section, you configure the rule to handle retrieveAll responses.

- ___ 1. Create a response rule to convert a retrieveAll SOAP response to a JSON-formatted response.
 - ___ a. Click **New Rule**, and enter the following name: REST_GET_AllRESP
 - ___ b. Set the direction as **Server to Client**.
- ___ 2. Configure a Match action to parse the SOAP response.
 - ___ a. Double-click the **Match** action to configure it.
 - ___ b. Click the plus sign (+) button to create a Matching Rule.
 - ___ c. Enter the following name for the Matching Rule: `retrieveAllResponse`
 - ___ d. Click the **Matching Rule** tab.
 - ___ e. Click **Add** to define a matching rule.
 - ___ f. On the Edit Matching Rule window, set the Matching Type to **XPath**.
 - ___ g. Click **XPath Tool**.
 - ___ h. On the “Build XPath Expression from sample XML File” window, set **Namespace Handling** to **local**.
 - ___ i. For the **URL of Sample XML Document** field, upload the `retrieveAllResponse.xml` sample SOAP response file from `<lab_files>/REST/`.
 - ___ j. In the sample file area, click `<p796:retrieveAllResponse>`. This selection indicates which element you want to check.
 - ___ k. Click **Done**.
 - ___ l. Click **Apply** to save the XPath.
 - ___ m. Click **Apply** to save the Matching Rule.
 - ___ n. Click **Done** to save the Match action. This match inspects the message to see whether it is a retrieveAll response.
- ___ 3. Add a Transform action to create a JSONx version of the SOAP response.
 - ___ a. In the policy editor, add a Transform action after the Match action.
 - ___ b. For the style sheet, upload `AllSOAP2JSONX.xsl` from `<lab_files>/REST/`. This style sheet parses the SOAP response and builds a JSONx version of the response.
 - ___ c. Click **Done** to save the Transform.
- ___ 4. Add another Transform to reformat the JSONx response into a JSON structure.
 - ___ a. Add another Transform action after the previous one.
 - ___ b. For this style sheet, use the supplied `jsonnx2json.xsl` file in the `store:///` directory on the appliance.

- ___ c. Since this action is the last action, set the Output context to **OUTPUT**.
 - ___ d. Click **Done** to save this action.
 - ___ e. On the policy editor, click **Apply Policy**.
 - ___ f. Click **Apply** on the Configure Multi-Protocol Gateway window, if it is enabled.
- ___ 5. Test the retrieval by location capability.
- ___ a. Use cURL to send a request for retrieveAll:

```
curl -G http://<dp_public_ip>:<mpgw_rest_port>/EastAddressSearch/people/
```
 - ___ b. The response is a list of all the people in the database:

```
"people":{  
    "person":{  
        "details":{  
            "city":"Philadelphia",  
            "state":"PA",  
            "street":"4153 Hickory Lane",  
            "zipcode":19273 },  
        "name":{  
            "firstname":"Willie",  
            "lastname":"Martinez",  
            "title":"Mr" }  
    },  
    "person":{  
        "details":{  
            "city":"Jacksonville",  
            "state":"FL",  
            "street":"3407 Jackson Drive",  
            "zipcode":32225 },  
        "name":{  
            "firstname":"Phyllis",  
            "lastname":"Taylor",  
            "title":"Ms" }  
    },  
    "person":{ }  
    ...  
    ...  
}
```

13.7.Catch incorrect requests to the RESTful multi-protocol gateway service

A best practice for non-XML input is to reject all invalid requests. You code one more request rule to reject any requests that do not match, and return a specific error string.

- 1. Create a request rule to catch any invalid requests.
 - a. Click **New Rule**, and enter the following name: REST_GET_unmatched
 - b. Set the direction as **Client to Server**.
- 2. Configure a Match action to catch any request.
 - a. Double-click the **Match** action to configure it.
 - b. Click the plus sign (+) button to create a Matching Rule.
 - c. Select an existing Matching Rule that matches on all URLs, or create one.
 - d. Click **Done** to save the Match action.
- 3. Add a Filter action to reject any message that entered this rule.
 - a. In the policy editor, add a Filter action after the Match action.
 - b. For the style sheet, upload filter-reject-all-unmatched.xsl from <lab_files>/REST/. This style sheet issues a <dp:reject> with a REST-related error message.



Information

The filter-reject-all-unmatched style sheet is a copy of the provided store:///filter-reject-all.xsl, with a modified error text message.

- c. This action is the last action, hence, set the Output context to **OUTPUT**.
- d. Click **Done** to save the Filter.
- e. On the policy editor, use the Configured Rules section to place this new rule at the bottom of the list.



Warning

Ensure that this request rule follows the original REST_GET_REQ request rule at a minimum.

- f. On the policy editor, click **Apply Policy**.
- g. Click **Apply** on the Configure Multi-Protocol Gateway window, if it is enabled.

- ___ 4. Send a request to a different application area, such as WestAddressSearch.
 - ___ a. Use cURL to send a request for `retrieveAll`, but to the **WestAddressSearch**, rather than the supported **EastAddressSearch**.

```
curl -G http://<dp_public_ip>:<mpgw_rest_port>/WestAddressSearch/people/
```

- ___ b. The response is a SOAP fault:

```
<?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>Unmatched REST request (from client)</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```



Information

For a service that completely bridges a REST-SOAP interface, the response would be an HTTP status code, most likely a 404, rather than the SOAP fault.

A “production-level” service checks more of the request URI for validity. All responses would be JSON or an HTTP status code, per a REST design.

-
- ___ 5. Save your configuration.

End of exercise

Exercise review and wrap-up

In this exercise, you learned how to send an HTTP request using a REST API on the DataPower appliance. You set up a multi-protocol gateway service that extracted and parsed the HTTP parameters, saving the HTTP method into a context variable. Then, invoking a rule for the HTTP GET method to convert the request into a SOAP request for the back-end EastAddressSearch web service. A style sheet that demonstrated how to build a SOAP request from the HTTP parameters was provided.

Appendix A. Exercise solutions

This appendix describes:

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

Part 1: Dependencies

Certain exercises depend on previous exercises, and on other resources, such as the need for the back-end application server to support service calls.

In all cases, the initial dependency is on **Exercise 1: Exercises setup**, where exercise setup, variable definition, and the numbering of ports are done.

Table 2: Dependencies

Exercise	Depends on exercise	Uses cURL	Uses Eclipse	Needs Application Server
1: Exercises setup		Yes	Yes	Yes*
2: Creating a simple XML firewall	1	Yes		
3: Creating an advanced multi-protocol gateway	1	Yes		Yes
4: Adding error handling to a service policy	1, 3	Yes		Yes
5: Creating cryptographic objects and SSL	1, 3, 6	Yes		Yes
6: Create a Web Service Proxy	1	Yes	Yes	Yes
7: Implementing an SLM monitor in a web service proxy	1, 6	Yes		Yes
8: Web service encryption and digital signatures	1, 6	Yes		Yes
9: Web service authentication and authorization	1, 6	Yes	Yes	Yes
10: Define 3-legged OAuth	1	Yes		Yes
11: Configure DataPower Patterns	1, 3	Yes		Yes
12: Configuring a multi-protocol gateway service with WebSphere MQ	1, 3	Yes		Yes
13: Implementing REST services	1, 6	Yes		Yes

If the class is using the standard images and setup, the East and West Address enterprise applications, WebSphere Application Server, WebSphere MQ, IBM HTTP Server, are running on the student image. Therefore, each student is using a different IP address for their labs. The `backend_server_ip` address and the `studentNN_image_ip` address are the same.

The instructor might create a Host Variable on DataPower so each student can use the variable `WSserverNN` with NN being their student number, as opposed to using the literal IP address.

Exercise 5 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 3: Exercise solution files

Exercise	Compressed solution file name
1: Exercises setup	
2: Creating a simple XML firewall	dev_Ex02_SimpleFW.zip
3: Creating an advanced multi-protocol gateway	dev_Ex03_AdvMPG.zip
4: Adding error handling to a service policy	dev_Ex04_Errors.zip
5: Creating cryptographic objects and SSL	dev_Ex05_CryptoSSL.zip
6: Create a Web Service Proxy	dev_Ex06_WSP.zip
7: Implementing an SLM monitor in a web service proxy	dev_Ex07_SLM.zip
8: Web service encryption and digital signatures	dev_Ex08_WSP_security.zip
9: Web service authentication and authorization	dev_Ex09_AAA.zip
10: Define 3-legged OAuth	dev_Ex10_OAuth.zip
11: Configure DataPower Patterns	N/A
12: Configuring a multi-protocol gateway service with WebSphere MQ	dev_Ex12_MQ.zip
13: Implementing REST services	dev_Ex13_REST.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_OAuth.zip.
- ___ 2. Import the .zip solution file into your application domain.
 - ___ a. From the **Control Panel**, in the vertical navigation bar, click **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. Because private keys (key files) are not exported, you also must create keys and certificates. In some exercise solutions, the key files are exported in the `local:` directory. After import, you move those files into the `cert:` directory.

IBM
®