

WebSphere Lab Jam

Connectivity

WebSphere DataPower

Lab Exercises



An IBM Proof of Technology

Catalog Number

Contents

CONNECTION PARAMETERS SPREADSHEET	5
LAB 1 OVERVIEW.....	6
1.1 INTRODUCTION	6
1.2 REQUIREMENTS	6
1.3 ICONS.....	6
1.4 INTRODUCTION TO WEBSPHERE DATAPOWER SOA APPLIANCES	7
1.5 WEBSPHERE DATAPOWER SOA APPLIANCES FAMILY	7
1.6 ACCESS CONTROL.....	8
1.7 APPLICATION DOMAINS	8
1.8 THE WEBSPHERE DATAPOWER WEBGUI.....	8
1.9 CONFIGURATION PROCEDURES	12
1.10 WEBSPHERE DATAPOWER SERVICES.....	13
1.11 WEBSPHERE DATAPOWER FLASH-BASED FILE SYSTEM	16
1.12 TROUBLESHOOTING TOOLS.....	20
1.13 LOGGING	21
1.14 WEBSPHERE DATAPOWER SOA APPLIANCES FIRMWARE	27
1.15 SUMMARY	28
LAB 2 WORKING WITH XML.....	29
2.1 SERVICE PROCESSING PHASES.....	29
2.2 CREATING THE MULTI-PROTOCOL GATEWAY SERVICE	31
2.3 SCHEMA VALIDATION	38
2.4 SOAP ENVELOPE SCHEMA VALIDATION	42
2.5 CONTENT-BASED FILTERING	43
2.6 TRANSFORMING WITH XSL AND XPATH.....	46
2.7 STYLESHEET CACHING	49
2.8 IMPLICIT XML THREAT PROTECTION.....	49
2.9 SUMMARY	52
LAB 3 SECURING XML MESSAGE CONTENT USING WS-SECURITY	53
3.1 PUBLIC KEY INFRASTRUCTURE (PKI).....	53
3.2 WS-SECURITY DIGITAL SIGNATURES.....	54
3.3 WS-SECURITY ENCRYPTION & DECRYPTION	64
3.4 SUMMARY	68
LAB 4 ACCESS CONTROL FRAMEWORK	69
4.1 EXTRACT IDENTITY & EXTRACT RESOURCE	69
4.2 AUTHENTICATE	69
4.3 CREDENTIAL AND RESOURCE MAPPING	70
4.4 AUTHORIZE	70
4.5 AUDIT & ACCOUNTING	70
4.6 LDAP AUTHENTICATION	70
4.7 SUMMARY	72
APPENDIX A. NOTICES	129
APPENDIX B. TRADEMARKS AND COPYRIGHTS	131

THIS PAGE INTENTIONALLY LEFT BLANK

Connection Parameters Spreadsheet

Student Suffix	DataPower ID's	Initial Password	Change Password
ID's for Morning Lab Session 1			
01	Student01	password	Passw0rd01
02	Student02	password	Passw0rd02
03	Student03	password	Passw0rd03
04	Student04	password	Passw0rd04
05	Student05	password	Passw0rd05
06	Student06	password	Passw0rd06
07	Student07	password	Passw0rd07
08	Student08	password	Passw0rd08
09	Student09	password	Passw0rd09
10	Student10	password	Passw0rd10
11	Student11	password	Passw0rd05
12	Student12	password	Passw0rd06
13	Student13	password	Passw0rd07
14	Student14	password	Passw0rd08
15	Student15	password	Passw0rd09

ID's for Afternoon Lab Session 2

16	Student16	password	Passw0rd11
17	Student17	password	Passw0rd12
18	Student18	password	Passw0rd13
19	Student19	password	Passw0rd14
20	Student20	password	Passw0rd15
21	Student21	password	Passw0rd16
22	Student22	password	Passw0rd17
23	Student23	password	Passw0rd18
24	Student24	password	Passw0rd19
25	Student25	password	Passw0rd20
26	Student26	password	Passw0rd15
27	Student27	password	Passw0rd16
28	Student28	password	Passw0rd17
29	Student29	password	Passw0rd18
30	Student30	password	Passw0rd19

DataPower Appliance Address:	XX.XX.XX:PPPP
DemoServer VMWare IM:	XX.XX.XX

Path to Lab Resources	C:\ labs\
-----------------------	-----------

Lab 1 Overview

This IBM® WebSphere® DataPower® SOA Appliances Proof of Technology (PoT) provides a hands-on experience for those needing to understand how WebSphere DataPower SOA Appliances can help ease and accelerate the deployment of enterprise service oriented architecture (SOA) implementations. Participants gain an appreciation for the ability of WebSphere DataPower to meet the demand for fast, secure, and reliable XML processing by creating various configurations that demonstrate a rich array of built-in functionality.

1.1 Introduction

IBM WebSphere DataPower SOA Appliances represent an important element in IBM's holistic approach to Service Oriented Architecture (SOA). IBM SOA appliances are purpose-built, easy-to-deploy network devices that simplify, help secure, and accelerate your XML and Web services deployments while extending your SOA infrastructure. These new appliances offer an innovative, pragmatic approach to harness the power of SOA while simultaneously enabling you to leverage the value of your existing application, security, and networking infrastructure investments.




1.2 Requirements

To complete the labs in this workbook, you'll need the following:

- A network attached workstation with sufficient memory (2GB minimum).
- VMware Workstation or Viewer to run the supplied student VMware image.
- An Internet browser.
- Network access to a WebSphere DataPower Integration Appliance with Firmware 4.0.1 or greater (XI50, XI52, XI50B, XI50Z).

1.3 Icons

The following symbols appear in this document at places where additional guidance is available.

Icon	Purpose	Explanation
	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
	Information	This symbol indicates information that might not be necessary to complete a step, but is helpful or good to know.
	Trouble-shooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

1.4 Introduction to WebSphere DataPower SOA Appliances

In this lab, you'll gain a high level understanding of the architecture, features, and development concepts related to the family of WebSphere DataPower SOA Appliances. Throughout the lab, you'll get a chance to use the WebSphere DataPower intuitive Web-based user interface (WebGUI) to explore the various aspects associated with appliance configuration and operation.

Upon completing this lab, you'll have a better understanding of:

- The WebSphere DataPower SOA Appliances family.
- Access Control.
- Application Domains.
- WebSphere DataPower Web-based User Interface (WebGUI).
- Configuration Procedures.
- The various WebSphere DataPower services.
- Local file management.
- Logging capabilities.
- Device management options.
- Firmware management.

1.5 WebSphere DataPower SOA Appliances Family

WebSphere DataPower SOA Appliances are a key element in IBM's holistic approach to Service Oriented Architecture (SOA). These appliances are purpose-built, easy-to-deploy network devices to simplify, help secure, and accelerate your XML and Web services deployments.

The DataPower appliance family includes the following:

- WebSphere DataPower XML Security Gateway XS40 - Capable of offloading overtaxed Web and application servers by processing XML, XSD, XPath and XSLT at wirespeed, this appliance enables faster results from application investments. It also delivers a comprehensive set of configurable security and policy enforcement functions including support for the latest WS-* standards.
- WebSphere DataPower Integration Appliances XI50, XI52, XI50B and XI50Z - IBM's hardware ESB, this appliance is built for simplified deployment & hardened security. It includes all of the XML and security functionality found in the DataPower XML Security Gateway appliance and adds capabilities for bridging multiple protocols.
- WebSphere DataPower B2B Appliance XB62 – Extends all of the capabilities of the DataPower Integration Appliance with B2B capabilities by providing AS1/AS2/AS3 messaging & trading partner profile management in a high-performance DMZ-ready appliance.
- WebSphere DataPower Edge Appliance XE82 - Provides an integrated edge-of-network traffic gateway that plans traffic consolidation, monitoring, management, and acceleration for web application delivery.

- WebSphere DataPower Caching Appliance XC10 – This appliance incorporates a large, 240 GB cache into the DataPower line of appliances from IBM, and adds elastic caching functions that enable your business-critical applications to scale cost effectively with consistent performance.

1.6 Access Control

There are three administrative interfaces for configuring WebSphere DataPower SOA Appliances:

- Command line interface
- Web-based graphical interface
- SOAP-based XML management interface

Through the various administrative interfaces, it is possible to access the entire range of configuration and status data. Access to the various administrative interfaces is tightly controlled through a variety of access control methods.

- **Access control list.** Only hosts with addresses in a listed range can access the appliance.
- **Accounts, groups, and access policies.** Local accounts can be created to gain access to the appliance. Groups facilitate an easy way of managing multiple accounts with similar access rights. Group access rights are defined using an *access policy*.
- **Role-based Management.** Extends local access control to use remote authentication and authorization servers, such as LDAP or RADIUS.

1.7 Application Domains

Application domains allow administrators to partition an appliance into multiple logical configurations. For example, in a production environment, a domain may represent a business area like shipping or accounting. In a development environment, each developer may have their own domain for testing. Configurations that are created in one domain are secure from other domains and are not visible.

By default, a newly initialized WebSphere DataPower appliance will have a single domain named **default**. The default domain should only be used for managing the network configuration and the appliance itself.

Application domains allow for easier porting of development domain configurations among appliances without affecting the core network for the appliance. A domain can easily be exported from one appliance and imported into another.

1.8 The WebSphere DataPower WebGUI

The PoT leader will assign a unique student number to you. Your user ID and application domain is based on your assigned number.

- Your *User Name* is **studentNN** where *NN* is your student number. If your student number is 2, then your user name would be **student02**.
- Your password is: **password**
- Your assigned application domain is the same as your user ID.

You're now ready to start exploring the WebSphere DataPower WebGUI. Sign into the WebGUI and change your password using the following steps:

- __1. Navigate your browser to the following secure URL: <https://datapower:9090>
- __2. Put your user name and password in the appropriate fields.
- __3. Select your domain from the dropdown list of domains, and then click **Login**.

Since this is the first time you are logging in, you'll be requested to change your password.

- __4. In the *Old Password* field, type your original password: **password**
- __5. In the remaining two fields, type a new password that you will use for the remainder of this Proof of Technology.
- __6. Click the **Change User Password** button.
- __7. In the confirmation dialog box, click the **Confirm** button.
- __8. In the success dialog box, click the **Close** button.
- __9. Log back into the appliance with your user name, new password and domain. **Don't forget to select your student domain from the dropdown domain list.**

Upon successful login, the DataPower Appliance Control Panel will be shown.

The screenshot displays the WebSphere DataPower XI50 Control Panel. The top navigation bar includes the 'WebSphere. DataPower XI50' logo, the user 'student01 @ TEC', the domain 'Domain: student01', and links for 'Save Config' and 'Logout'. The IBM logo is in the top right corner.

Control Panel

Search

Navigation menu:

- Status
- Services
- Network
- Administration
- Objects

Firmware Rev: XI50.4.0.1.0
Build: 199774
IBM WebSphere DataPower
Copyright IBM Corporation 1999-2011

Control Panel

Services

- Web Service Proxy
- Multi-Protocol Gateway
- XML Firewall
- Web Application Firewall
- XSL Accelerator

Monitoring and Troubleshooting

- View Logs
- Troubleshooting
- Web Services Monitor
- View Status

Files and Administration

- File Management
- System Control
- Import Configuration
- Export Configuration
- Keys & Certs Management

Having trouble? Additional support resources are available.

- Find more information on the [WebSphere DataPower Support Site](#).
- Generate an [error report](#).
- [Contact IBM WebSphere DataPower SOA Support](#).

There are several areas in the WebGUI worth noting:

- The top banner section contains some basic status information, such as the current user and domain.
 - The [Save Config](#) link is used to save all of your changes into the device's flash memory. When you make changes to a configuration, the changes are immediately active, but they are not saved to the flash memory until you click this link.
 - The [Logout](#) link will end the current session. Any changes you made will remain active.
- The left side of the browser window is occupied by the navigation tree. At the top is a link (labeled "Control Panel") for quick access to the control panel. The navigation tree is divided into several sections. Clicking on the section name will expose additional actions within that section.
 - **Status:** provides menu options to view the overall status of the device, network connections, configurations, and many other objects within the system.
 - **Services:** provides options for configuring and managing all of the services available on the appliance.
 - **Network:** provides menu options that help you work with network configuration and settings.
 - **Administration:** provides options that help you administer the device, such as creating domains, users, exporting and importing configurations, etc.
 - **Objects:** contains menu options to create and manage every type of object supported by WebSphere DataPower.

The navigation tree also includes a search box that helps you quickly navigate to a specific function or object.

- The body of the page shows the *Control Panel*. It's divided into three sections, each containing icons for performing frequently used tasks.
 - **Services** - provides access to wizards that step you through the creation of a variety of service objects such as a Web Service Proxy or a Multi-protocol Gateway.
 - **Monitoring and Troubleshooting** - provides easy access to system logs, troubleshooting tools, Web service monitors and device status pages.
 - **File and Administration** - provides easy access to the onboard flash-based file system, a system control panel, import and export tools, and a key and certificates management tool.

1.9 Configuration Procedures

There are three phases to the setup and configuration of a WebSphere DataPower SOA appliance. Each of these phases involves a different set of objects, and often each phase is performed by different enterprise personnel.

1.9.1 Network services and user access configuration phase

In this phase, the various objects that control the Ethernet interfaces, packet routing, time services and emergency failure notification are configured. The basic networking values, such as IP addresses, gateways, etc., are setup during this phase. These objects and settings all reside in the **default** domain of the appliance and are accessible only to users with administrative privileges.

During the configuration phase, administrators will also setup the various application domains, users, groups, and access policies. User access policies determine who can access the appliance to view or alter its configuration.

1.9.2 Application Development Phase

During this phase, architects and developers create the various services that implement the solutions needed to meet enterprise SOA requirements. This phase is often iterative as more and more top level services are configured on the appliance.

Services can be created in a variety of ways depending on the developer's experience level. Configuration wizards provide the fastest means of creating a new service and its related objects. More experienced developers may find it faster to create the configuration objects manually. In this Proof of Technology, you'll create objects both manually and using built-in wizards.

1.9.3 Production Management Phase

This phase occurs when an appliance is moved into a runtime production environment. Administrators commonly require that the appliance provide the means to produce status updates on a regular and timely basis. It must also provide a quick, secure, and reliable means of upgrade, configuration deployment and backup, and that access to the configuration interface is limited. Objects such as Simple Network Management Protocol (SNMP) communities, statistical monitors, and audit logs are configured as the appliance goes into production.

1.10 WebSphere DataPower Services

WebSphere DataPower SOA Appliances provide services to process traffic. This section discusses the various service objects and their typical use cases.

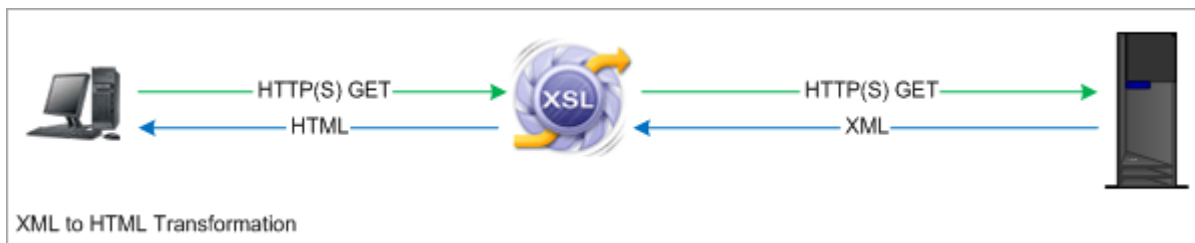
The *Services* section of the control panel contains a group of icons that represent the most commonly used services. The following image shows the service icons available on an XI50:



1.10.1 XSL Accelerator



The XSL Accelerator validates and transforms incoming or outgoing XML documents. An XSL Accelerator service would proxy a backend service, applying all the necessary schema validation and transformations to the incoming document before forwarding the message to the backend service. For response processing (from the server), it can perform content rendering by transforming outbound XML to HTML (or any other markup language) using XSL.

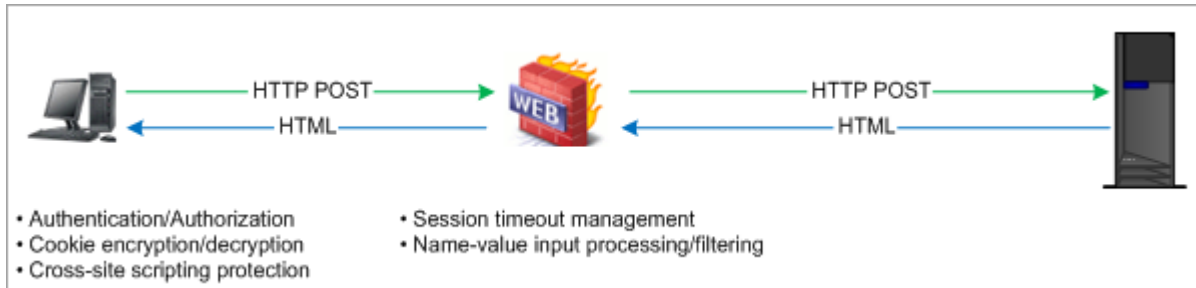


One use case for this service object is XML to HTML rendering. A browser-based client makes a request to a web application. The XSL Accelerator service acts as a proxy between the client and the backend web application server. The GET (or POST) is received by the XSL Accelerator service, and then forwarded to the backend server. The backend server returns raw XML to the XSL Accelerator, which then transforms the XML to HTML using an XSL template. The template may reside on the appliance, or be fetched (and cached) from a remote server.

1.10.2 Web Application Firewall



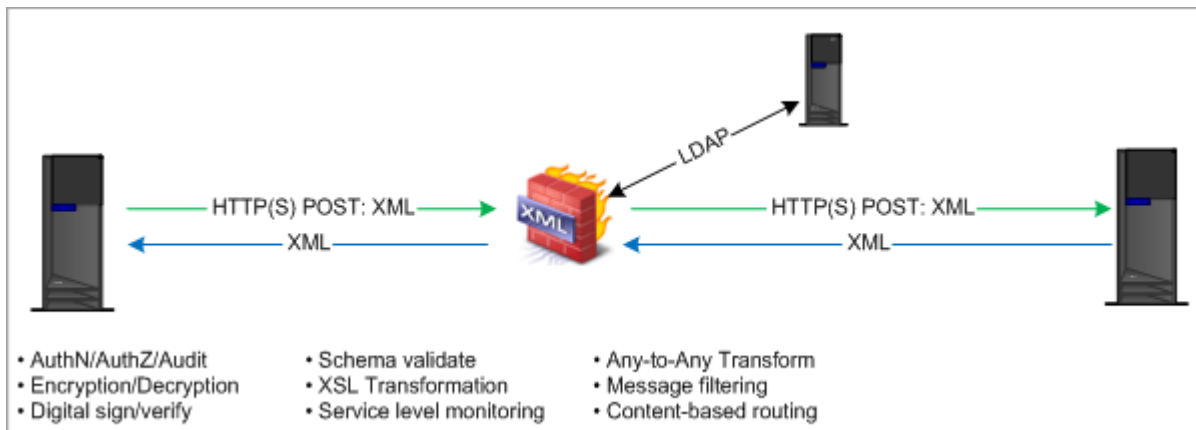
The Web Application Firewall service is designed to provide firewall and security services for standard HTML over HTTP Web application traffic. In addition to protecting against common threats, the Web Application Firewall can enforce specific policies against the data flowing between the browser and the server. For instance, it can enforce cookie existence and value policies, or require that specific form fields contain only certain values.



1.10.3 XML Firewall



The XML Firewall is a general purpose HTTP(S) service that can process both XML and non-XML payloads. A wide array of actions can be applied to both inbound and outbound messages, such as encryption/decryption, digital signatures, XSL transformations, filtering, schema validation, and dynamic routing to name just a few. Checks for XML threats are provided automatically.



Processing policies have access to all HTTP related details (headers, form fields, payload, status, etc.) for both the request and the response and can therefore make decisions or process messages based on the header's existence or contents.

A robust authentication and authorization engine, with built-in integration for a wide variety of policy servers (LDAP, IBM Tivoli® Access Manager, Kerberos/SPNEGO, IBM RACF®, etc.) can apply simple to complex security policies to both inbound and outbound messages. Security protocol mediation, such as HTTP Basic Authentication to SAML, or Kerberos/SPNEGO to IBM Lightweight Third-Party Authentication (LTPA), is easily configured through the WebGUI. There's support for the latest security standards such as XACML, SAML, WS-Security, WS-Policy and WS-I Basic Profile.

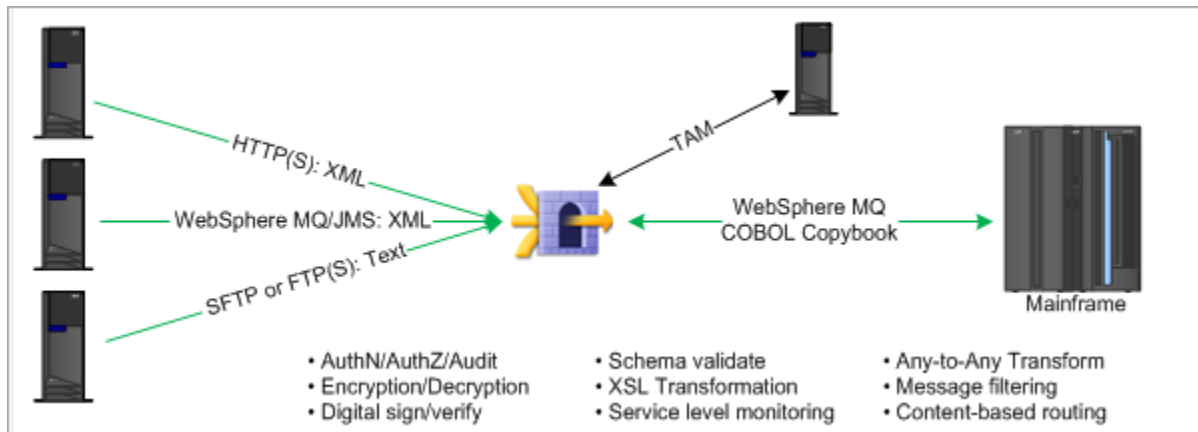
The XML Firewall also includes support for some of the latest WS-* standards, including WS-Reliable Messaging and WS-Addressing.

1.10.4 Multi-Protocol Gateway



The Multi-Protocol Gateway service builds on the XML Firewall's XML and security functionality by adding support for multiple protocols. In addition to HTTP and HTTPS, the Multi-Protocol

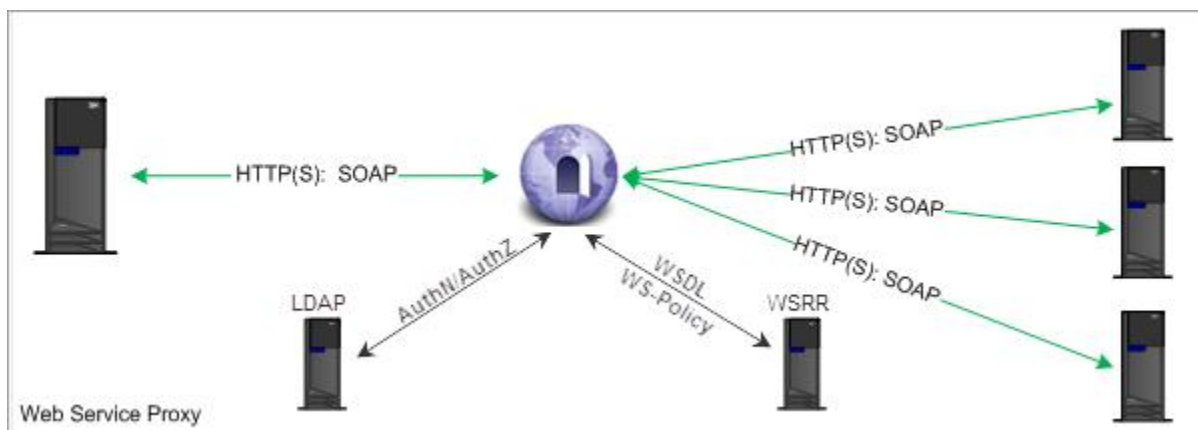
Gateway supports WebSphere MQ, WebSphere JMS, TibcoEMS, FTP(S), SFTP, NFS and IMS. All of these protocols can be mixed and matched as necessary. Messages received over HTTPS can easily be routed to WebSphere MQ or JMS.



1.10.5 Web Service Proxy



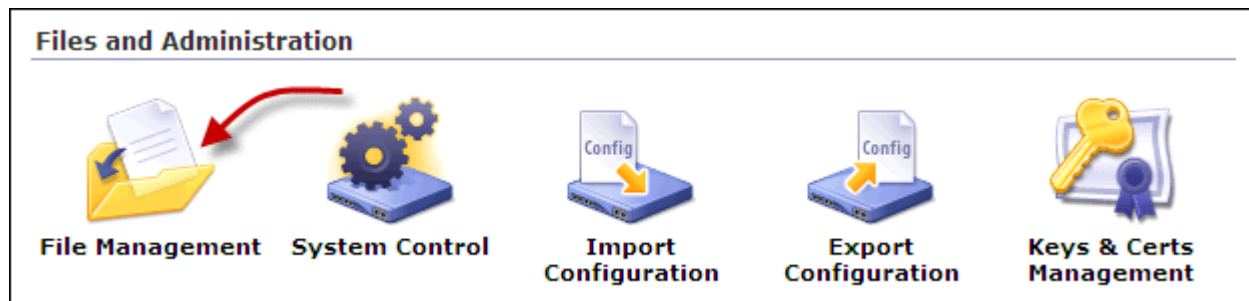
The Web Service Proxy provides all of the same services as a Multi-Protocol Gateway service; however it provides automatic configuration based on one or more Web Service Definition Language (WSDL) files. WSDL files may be obtained through subscriptions to a Universal Description, Discovery, and Integration (UDDI) or WebSphere Service Registry and Repository. A single Web Service Proxy object can act as a single point of entry for multiple WSDLs, automatically routing (or redirecting) the requests to the appropriate backend service.



The Web Service Proxy will automatically apply schema validation to both inbound and outbound messages, further assuring message validity. Processing and security policies can be applied not only at the entire service level, but for individual operations within the service as well.

1.11 WebSphere DataPower Flash-based File System

__1. In the Control Panel, click on the File Management icon.



You should see the file explorer similar to the one below (additional directories may appear depending on installed hardware options).

Manipulate Checked Files: <input type="button" value="Delete"/> <input type="button" value="Copy"/> <input type="button" value="Rename"/> <input type="button" value="Move"/>			
Name	Action	Size	Modified
cert:	Actions...		
chkpoints:	Actions...		
config:	Actions...		
export:	Actions...		
local:	Actions...		
logstore:	Actions...		
logtemp:	Actions...		
pubcert:	Actions...		
sharedcert:	Actions...		
store:	Actions...		
temporary:	Actions...		
Manipulate Checked Files: <input type="button" value="Delete"/> <input type="button" value="Copy"/> <input type="button" value="Rename"/> <input type="button" value="Move"/>			

The Flash-based file system has a set of predefined directories. Some directories are shared across domains, such as the **store:** directory, while others are specific to a single domain such as the **local:** directory. The following is a list of only the most common directories and their contents:

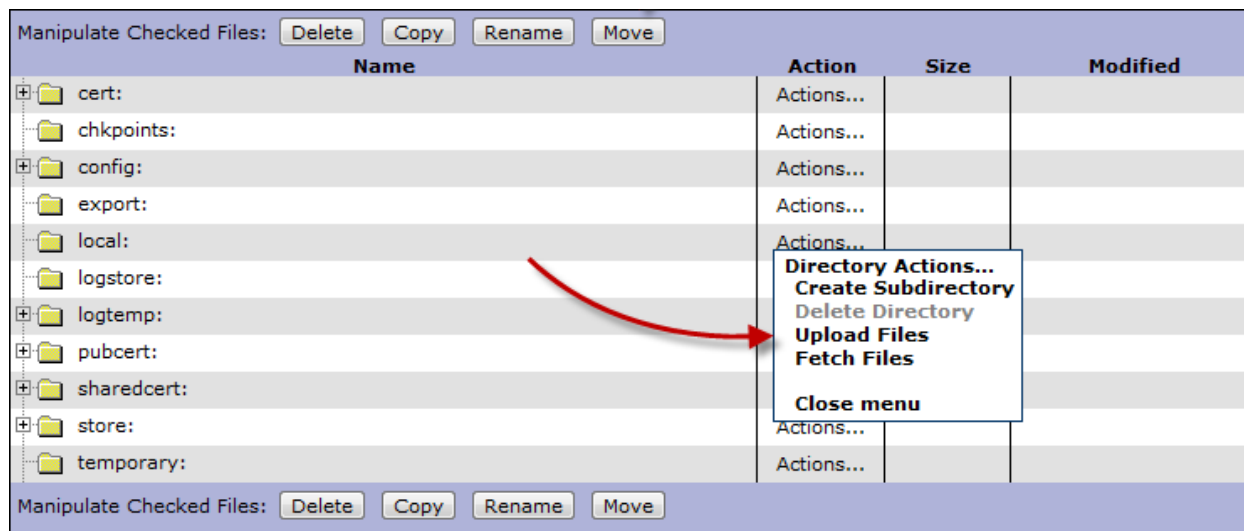
Directory	Usage
cert:	This encrypted directory contains private key and certificate files used by services within the domain. Each application domain contains one cert: directory.
chkpoints:	This directory contains the configuration checkpoint files for the appliance.
config:	This directory contains the configuration files for the appliance. Each application domain contains one config: directory.
local:	This general-purpose directory contains miscellaneous files that are used by the services within the domain, such as XSL, XSD, and WSDL files. Each domain includes exactly one local: directory.
logstore:	This directory contains log files that are stored for future reference.
logtemp:	This directory is the default location of log files, such as the appliance-wide default log. This directory can hold only 13 MB.
pubcert:	This encrypted directory contains the security certificates that are used commonly by Web browsers. This directory is shared across domains
sharedcert:	This encrypted directory contains security certificates that are shared with partners. Each appliance contains only one sharedcert: directory. This directory is shared across domains.
store:	This directory contains example style sheets, default style sheets, and schemas that are used by the appliance. Do not modify the files in this directory. Each appliance contains only one store: directory. By default, this directory is visible to all domains.
temporary:	This directory is used by processing rules as temporary disk space. Each application domain contains one temporary: directory. This directory is not shared across domains.

The Flash-based file system is used for storing WebSphere DataPower firmware and configuration data as well as service-related artifacts such as XSL stylesheets, keys, certificates, and schema definitions.

Static files such as schemas, WSDLs and XSL stylesheets are generally hosted off the box and fetched (and cached) as required. Storing static documents off-box not only reduces flash storage requirements, but greatly simplifies the deployment process when multiple WebSphere DataPower appliances are clustered and share common artifacts.

For this Proof of Technology, you'll need to upload a few files into your local: directory. The following steps will guide you through the process.

- __2. Click on the [Actions](#) link associated with the **local:** directory to reveal the actions pop-up menu (see below).



- __3. Click the [Upload Files](#) link.
- __4. Perform the following steps to upload four files.
- __a. Click on the **Choose File** button, and select `c:\labs\files\ProductService.wsdl`.
 - __b. Click the **Add** button to add the file to the upload queue.
 - __c. Repeat the process for `c:\labs\files\customFilter.xml`.
 - __d. Repeat the process for `c:\labs\files\xacml-policy.xml`.
 - __e. Repeat the process for `c:\labs\files\xacml-request-binding.xml`.
- __5. Click the **Upload** button (or Browse button, depending on your browser) to upload the files into the `local:` directory.
- __6. Click the **Continue** button to dismiss the upload confirmation page.

- __7. Click on the small plus sign to the left of the local: directory and verify that all files were uploaded.

+ cert:	Actions...		
chkpoints:	Actions...		
+ config:	Actions...		
export:	Actions...		
- local:	Actions...		
customFilter.xml	Edit	774	2011-08-18 20:41:36
ProductService.wsdl	Edit	3372	2011-08-18 20:18:47
xacml-policy.xml	Edit	3848	2011-08-20 19:48:45
xacml-request-binding.xml	Edit	2699	2011-08-20 19:48:36
logstore:	Actions...		
+ logtemp:	Actions...		

Now you'll repeat that process and upload several keys and certificates into the **cert:** directory.

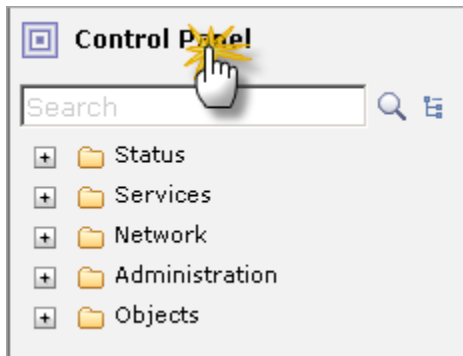
- __8. Click the [Actions...](#) link to the right of **cert:**, then select [Upload Files](#).
- __9. Perform the same steps as before and select the following files:
- __a. c:\labs\keysAndCerts\ProductService-privkey.pem
 - __b. c:\labs\keysAndCerts\ProductService-sscert.pem
 - __c. c:\labs\keysAndCerts\consumer-privkey.pem
 - __d. c:\labs\keysAndCerts\consumer-sscert.pem
 - __e. c:\labs\keysAndCerts\soapUI-sscert.pem
- __10. Make sure you've uploaded all pem files into the cert: directory (see below).

- cert:	Actions...		
consumer-privkey.pem	Details	887	2011-08-20 20:30:06
consumer-sscert.pem	Details	940	2011-08-20 20:30:06
ProductService-privkey.pem	Details	891	2011-08-19 01:11:08
ProductService-sscert.pem	Details	997	2011-08-19 01:11:08
soapUI-sscert.pem	Details	964	2011-08-19 01:12:08

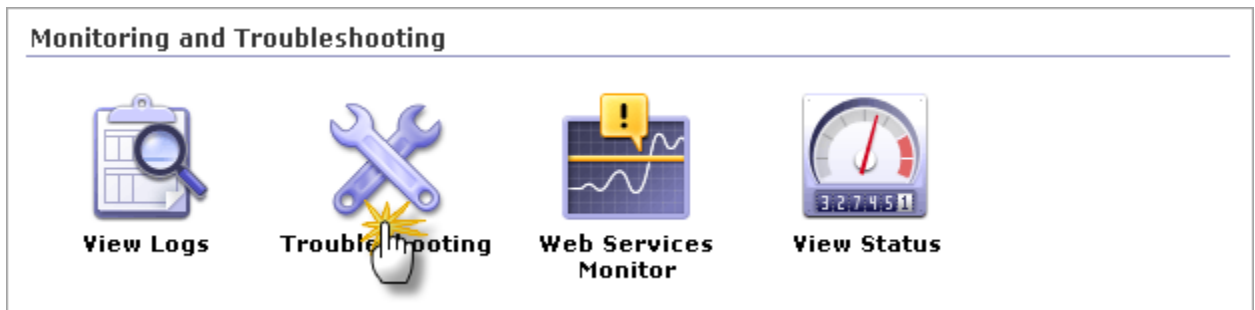
1.12 Troubleshooting Tools

During the development phase, there are often times when a service configuration produces unexpected results. WebSphere DataPower appliances have a number of built-in troubleshooting tools that can help pinpoint the cause of problems.

- ___1. In the Navigation pane (on the left side), click the [Control Panel](#) link to redisplay the control panel.



- ___2. In the *Monitoring and Troubleshooting* section, click on the **Troubleshooting** icon to reveal the troubleshooting tools page.



The Troubleshooting page has several tools used for troubleshooting both configuration and network problems.

- Ping Remote and TCP Connection Test are used primarily for network connectivity troubleshooting.
- Set Log Level is used to change the logging verbosity. This is a domain-wide setting which increases or decreases the granularity of messages that are written to the log. The default log level is *error*.
- Generate Log Event is used to write a specific message to the logs. This is often used for testing log targets (discussed in the next section).
- The Generate Error Report and Send Error Report functions are used when it becomes necessary to engage IBM Support to troubleshoot a problem. Generating an error report will create a special file containing detailed system and trace information used by support engineers.
- View Running Config allows you to see what parameters are currently in effect for the domain.

1.13 Logging

WebSphere DataPower appliances have a built-in publish-subscribe logging mechanism that is robust and flexible. As transactions flow through the appliance, many events occur. Some of these events occur as a result of normal processing, while others occur as a result of an exception such as a transaction being rejected due to an authentication or authorization failure.

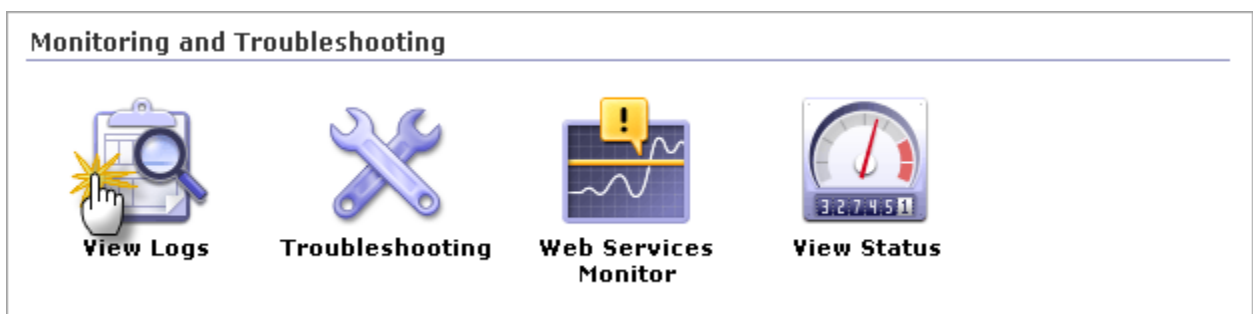
1.13.1 Setting the Logging Level to Debug

By default, the logging level is set so that only messages with a maximum priority of “Error” are written to the system log. In this section, you’ll change the default log level to “debug”, resulting in a much more granular level of logging. This not only is helpful in seeing what steps are executing, but helps in troubleshooting when things aren’t going as expected.

- __1. In the *Logging* section, change the *Log Level* dropdown to: **debug**
- __2. Click the **Set Log Level** button to activate the change.
- __3. In the *Confirmation* window, click the **Confirm** button.
- __4. Click the **Close** button to dismiss the window.

Throughout the various configuration forms, there are links that enable you to view the logs. For example, right above the *Log Level* is a magnifying glass icon that, when clicked, will open a window showing the system log. You can also view the log from the main control panel.

- __5. Click on the [Control Panel](#) link in the upper left corner of the browser window.
- __6. In the *Monitoring and Troubleshooting* section, click on the **View Logs**.



Clicking on the View Logs icon will take you to the system log page, which by default shows the last 50 entries in the default log. The interface enables you to filter the entries by category and/or priority, in order to limit the number of lines.



Since there has been minimal activity in your student domain, your log will likely contain only one or two messages. The following image shows a more active log.

Refresh Log Target: <input type="text" value="default-log"/> Filter: <input type="text" value="(none)"/> <input type="text" value="(none)"/>							
current time: 12:03:41 on 2009-05-15							
time ▼	category	level	tid	dir	client	msgid	message
Fri May 15 2009							
12:02:30	mpgw	info	15890	e	10.0.1.2	0x80e000b6	mpgw (MyTransformer): No match from processing policy 'DoNothing' for code '0x00d30003'
12:02:30	mpgw	error	15890	e	10.0.1.2	0x00d30003	mpgw (MyTransformer): Rejected by filter; SOAP fault sent
12:02:30	multistep	error	15890	>	10.0.1.2	0x80c00008	mpgw (MyTransformer): rule (default): implied action filter: 'INPUT store:///filter-reject-all.xsl' failed: Rejected
12:02:30	xslt	error	15890	>	10.0.1.2	0x80c00010	mpgw (MyTransformer): Execution of 'store:///filter-reject-all.xsl' aborted: Rejected
12:02:30	xmlfilter	info	15890	>	10.0.1.2	0x80c00037	mpgw (MyTransformer): Reject set: Rejected
12:02:30	xslt	debug	15890		10.0.1.2	0x80a002ac	xmlmgr (default): xsltstore:///filter-reject-all.xsl
12:02:30	xslt	debug	15890		10.0.1.2	0x80a002aa	xmlmgr (default): xsltstore:///filter-reject-all.xsl
12:02:30	multistep	debug	15890	>	10.0.1.2	0x80c0004e	mpgw (MyTransformer): Stylesheet URL to compile is 'store:///filter-reject-all.xsl'
12:02:30	xmlparse	debug	15890	>	10.0.1.2	0x80e003ab	mpgw (MyTransformer): Finished parsing: http://10.0.1.25:3001/EchoService/loopback
12:02:30	xmlparse	debug	15890	>	10.0.1.2	0x80e003a6	mpgw (MyTransformer): Parsing document: 'http://10.0.1.25:3001/EchoService/loopback'

For additional filtering, you can click a transaction id (tid), client IP address, or error code. Each of these opens a new window with messages related to the selected value; for example, clicking a transaction ID displays only messages from that transaction.

1.13.2 Log targets

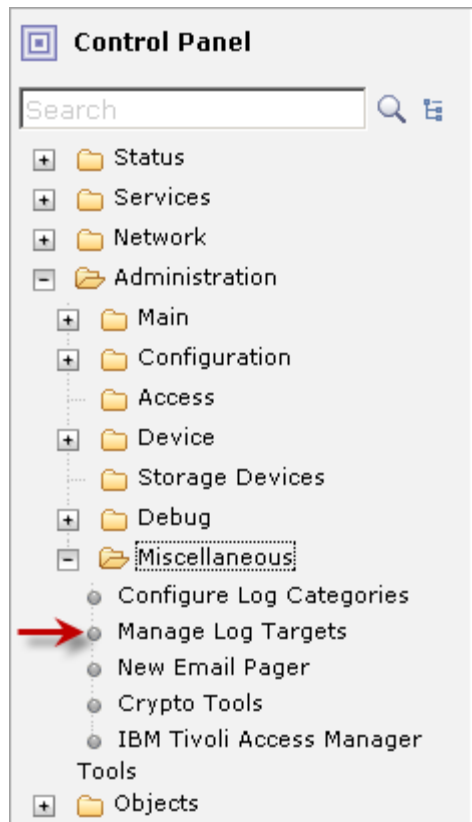
The logging subsystem on WebSphere DataPower SOA Appliances is based on the “publish-subscribe” paradigm that enables distribution of selected messages to various protocols and destinations. Publishers include the DataPower appliance itself as well as the various user-configured services and their supporting objects. For example, the DataPower appliance may log a message to indicate that a network connection is failing. Similarly, a user-configured MQ front side handler may log a message to indicate that the queue manager has become unresponsive.

Log targets act as the subscribers to published messages. Log targets can:

- Capture messages and forward them to a variety of different logging server types such as syslog and syslog-ng.
- Save messages in DataPower's flash memory or on the built-in logging hard drive.
- FTP saved log files off the appliance to an FTP server.
- Subscribe or suppress messages by event codes or category.
- Filter messages for a specific user-configured object.
- Filter messages associated with a specific IP address.

- Trigger a set of actions to occur when a specific log message is received.

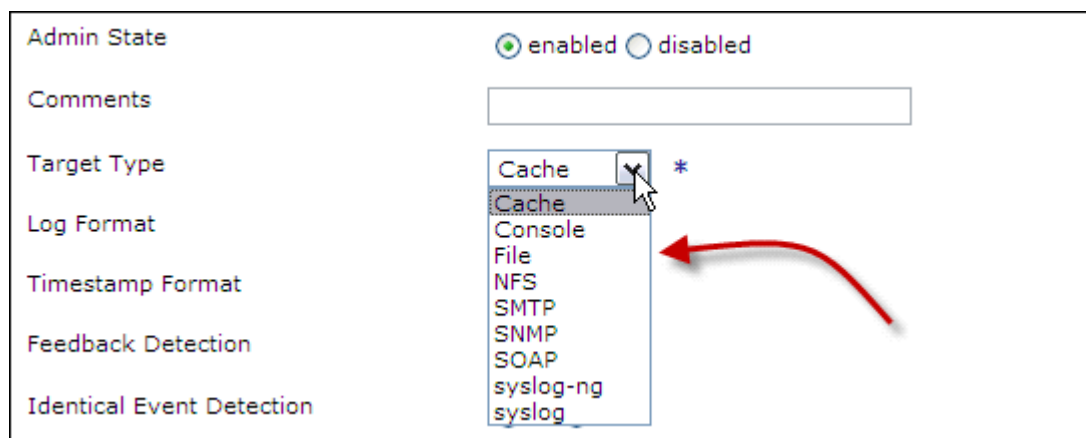
1. Expand the navigation tree to expose the Manage Log Targets option. The path is: Administration → Miscellaneous → Manage Log Targets



2. Click [Manage Log Targets](#).

Click the **Add** button to create a new Log Target.

3. On the *Main* tab, Locate the *Target Type* field and click the dropdown to reveal the list of available log target types that you can create. You should see a list similar to the following image.



The dropdown list shows the various log target types supported by the logging subsystem.














- **Cache:** Writes log entries to system memory (this is how the default log is setup).
- **Console:** Writes log entries to the screen when using Telnet, Secure Shell (SSH), or command line access through the serial port.
- **File:** Writes log entries to a file on the appliance.
- **NFS:** Writes log entries to a file on a remote Network File System (NFS) server.
- **SMTP:** Forwards log entries as email to the configured remote SMTP servers and email addresses. Before sending, the contents of the log can be encrypted or signed.
- **SNMP:** Forwards log entries as SNMP traps to configured recipients.
- **SOAP:** Forwards log entries as SOAP messages.
- **syslog-ng:** Forwards log entries using Transmission Control Protocol (TCP) to a remote syslog daemon.
- **syslog:** Forwards log entries using User Datagram Protocol (UDP) to a remote syslog daemon.

1.13.3 Log Categories

Log targets filter captured messages by event category. The use of categories allows log targets to subscribe to specific messages, such as appliance messages, network messages, or particular service messages. In addition to the predefined log categories specific to WebSphere DataPower objects and operations, you can create your own custom log categories which are more specific to your applications.

1. Back in the *Administration* section of the navigation tree, locate and click on the [Configure Log Categories](#) link.

A list of all predefined log categories will be displayed.

Name	Status	Op-State	Logs	Comments
aaa	saved	up		AAA Policy
all	saved	up		All Categories
audit	saved	up		Audit Trace
auth	saved	up		Authentication
cert-monitor	saved	up		Crypto Certificate Monitor
cli	saved	up		CLI Trace
cluster-service	saved	up		Cluster Service
crypto	saved	up		Crypto Subsystem
fibre-channel	saved	up		Fibre Channel
file	saved	up		File Management
file-capture	saved	up		XML File Capture
file-poller	saved	up		File Poller
ftp	saved	up		FTP Message Exchange

1.13.4 Appliance management

There are a number of methods that administrators can use to manage WebSphere DataPower SOA Appliances. These methods include:

- Manually exporting and importing configurations. Configurations can include a single object, an entire service, an entire domain, or an entire appliance. Configurations can be exported either as zip or XML files. Once exported, a configuration can easily be imported.
- Scripting. Command Line interface (CLI) commands can be scripted and executed to perform DataPower configuration tasks. Scripting can also be accomplished using SOMA (SOAP management interface) and integrated with high level programming languages.
- Appliance Management Protocol (AMP) and WebSphere Appliance Management Toolkit. This includes a set of Java components that can be leveraged to perform common management routines such as backup, restore, etc.
- WebSphere Appliance Management Center. This separately licensed product provides a browser-based graphical user interface allowing for the management of clusters of DataPower appliances. Tasks include configuration synchronization and firmware maintenance.

1.13.5 Backup and Restore

Administrators can use the Export Configuration utility to export a complete appliance back-up or export selected portions of the appliance configuration.

The Import Configuration utility is used to restore a complete appliance back-up or selected portions of an exported configuration.

- __1. At the top of the left navigation pane, click the [Control Panel](#) link.
- __2. In the bottom row of icons, click the *Export Configuration* icon.
- __3. Leave the default selection of *Export configuration and files from the current domain* and click the **Next** button.
- __4. Change the *Export File Name* field to: **MyExport**
- __5. Under the heading *Select configuration objects to export*, make sure **All Objects** is selected; then click the right pointing button to move the selected objects into the *Selected Objects* box.

When you click the right pointing arrow, the right side box will become populated with all of the objects in your domain. The objects in the right box will be the objects that are exported.

- __6. Click the **Next** button. The export file named MyExport is now created and ready for you to download to your workstation.
- __7. Click the **Download** button. You'll be prompted for a location to save the exported file. You can save the file anywhere on your workstation.
- __8. Click the **Done** button.

The file you just downloaded contains a complete backup of your application domain. The MyExport.zip file can now be imported into another WebSphere DataPower appliance to recreate an exact duplicate of your domain.

1.13.6 Device Status

The built-in monitoring subsystem can provide complete details as to the operational status of the appliance, including firmware and library information as well as memory usage, CPU utilization and hardware operational circumstances. All of this information is viewable from within the WebGUI as well as through remote monitoring tools (discussed in the next section).

___9. In the navigation tree, expand the *Status* menu to reveal the various status sections.

___10. Locate and expand the *System* section and explore the various status details.

1.13.7 Remote monitoring

Administrators can monitor the health and activity of the appliance with any of the following protocols:

- SNMP
- Web Services Distributed Management (WSDM)
- WS-Management
- Proprietary SOAP application programming interface (API)

Remote consoles such as SNMP console, or an IBM Tivoli Composite Application Manager for SOA console, can display throughput, CPU and memory usage, transaction latency, and general responsiveness of an appliance with these protocols. The following image shows a third party SNMP Management Information Base (MIB) Browser showing memory usage statistics.

iReasoning MIB Browser

Address: xi50 Advanced... OID: .1.3.6.1.4.1.14685.3.1.8.1.2.2 Operations: Get Next Go

MIB Tree

- RFC1213-MIB.iso.org.dod.internet.mgmt.mib-2
- HOST-RESOURCES-MIB.iso.org.dod.internet.mgmt.mib
- DATAPOWER-STATUS-MIB.iso.org.dod.internet.private
 - dpModules
 - dpStatusMIB
 - dpManagement
 - dpStatus
 - dpStatusActiveUsersTable
 - dpStatusARPStatusTable
 - dpStatusDateTimeStatus
 - dpStatusVersion
 - dpStatusMemoryStatus
 - dpStatusMemoryStatusUsage
 - dpStatusMemoryStatusTotalMemory
 - dpStatusMemoryStatusUsedMemory
 - dpStatusMemoryStatusFreeMemory
 - dpStatusMemoryStatusReqMemory
 - dpStatusMemoryStatusXG4Usage
 - dpStatusMemoryStatusHoldMemory
 - dpStatusRoutingStatusTable
 - dpStatusStylesheetStatusTable

Result Table

Name/OID	Value	Type
dpStatusMemoryStatusUsage.0	14	Gauge
dpStatusMemoryStatusTotalMemory.0	4024120	Gauge
dpStatusMemoryStatusUsedMemory.0	574478	Gauge
dpStatusMemoryStatusFreeMemory.0	3449642	Gauge
dpStatusMemoryStatusReqMemory.0	563980	Gauge
dpStatusMemoryStatusXG4Usage.0	0	Gauge
dpStatusMemoryStatusHoldMemory.0	143350	Gauge
dpStatusStylesheetCachingSummaryXMLManag...	default	OctetString
dpStatusStylesheetCachingSummaryXMLManag...	default-attempt-stream	OctetString
dpStatusStylesheetCachingSummaryCacheSize.2	256	Gauge

Name: dpStatusMemoryStatus
 OID: .1.3.6.1.4.1.14685.3.1.5
 MIB: DATAPOWER-STATUS-MIB
 Syntax:
 Access:
 Status:
 DefVal:

.iso.org.dod.internet.private.enterprises.datapower.dpManagement.dpStatus.dpStatusStylesheetCachingSummaryTable.dpStatusStylesheetCac... 3:13:51 PM 28M of 29M

1.13.8 Configuration Comparison, Checkpoint, and Restore

Administrators can use the Configuration Comparison utility to determine what has changed between current and saved configurations, including previously exported configurations.

Configuration checkpoints can be set at any time within an application domain. An administrator can then compare these checkpoints to any other configuration or roll-back the configuration of a domain to an existing checkpoint.

1.14 WebSphere DataPower SOA Appliances Firmware

Unlike traditional servers which require an operating system and various layers of installed software, WebSphere DataPower SOA Appliances rely on a single firmware image that provides all required functionality. Updating the firmware in a WebSphere DataPower appliance is a fast and simple process. The firmware image is first downloaded from IBM's support site and then uploaded to the appliance. Once uploaded, the authenticity of the firmware is verified, then decrypted, and finally applied. The previously running firmware is maintained on the device in the event a rollback is necessary.

1.15 Summary

In this lab, you learned:

- About the various tools and procedures used to configure WebSphere DataPower SOA Appliances.
- Application domains are used to logically partition a DataPower appliance. A domain can be used for an organizational line of business, or as a location for one or more developers to collaborate when implementing a solution.
- Configuration is accomplished through any of three administrative interfaces: command line (CLI), WebGUI, and SOAP-based XML interface.
- How to upload a file to the local: directory in the Flash-based file system.
- How WebSphere DataPower appliances control access to their administrative interfaces through the use of access control lists, user accounts, groups, and access policies.
- About the three configuration phases: network services/user configuration phase, application development phase, and production management phase.
- About the various WebSphere DataPower services that you can use to create simple to complex processing policies (XSL Accelerator, Web Application Firewall, XML Firewall, Multi-Protocol Gateway, and Web Service Proxy).
- How the built-in logging subsystem is based on the publish-subscribe paradigm, with log targets acting as subscribers to specific message categories.
- How WebSphere DataPower appliances provides complete system status and metrics from the WebGUI.
- That various monitoring protocols such as SNMP, WSDM, and WS-Management are supported.

Lab 2 Working with XML

Prerequisites: This lab requires the completion of lab 1.

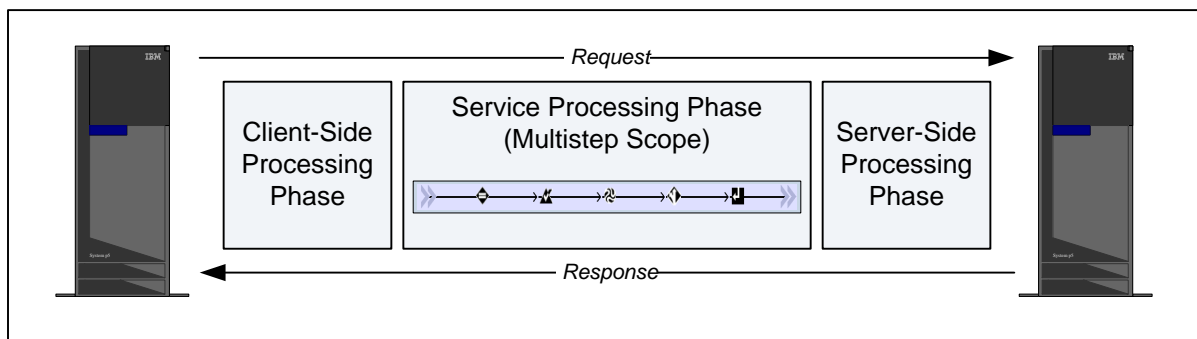
In this lab, you'll create a fully functional Multi-Protocol gateway service that will perform various functions against a request containing an XML (SOAP) payload.

Upon completing this lab, you'll have a better understanding of:

- How messages are processed
- The WebSphere DataPower object-oriented configuration architecture
- The Multi-Protocol Gateway service configuration
- Front-side protocol handlers
- Configuring Processing Policies, Rules, and Actions
- Matching Rules
- Validating XML documents against a schema
- Built-in XML threat protection and virus scanning support
- Content-based Message Filtering
- Transforming XML with XSL and XPath
- XSL caching

2.1 Service Processing Phases

When a service receives a message from a designated IP and port, a sequence of events are set into motion before the message is ultimately forwarded to its intended destination. The events are separated into three distinct phases: client-side processing, service processing, and server-side processing.



2.1.1 Client-Side (Front) Processing Phase

During this phase, the received message will be directed to the service object that is configured for the IP address and port combination on which the message was received. Once the service object (such as a Multi-protocol Gateway or XML Firewall) receives the message, a significant amount of processing of the message occurs. For example:

- If SSL is configured for the service, SSL negotiation and decryption of the data stream will occur.
- SOAP envelope validation.
- Protocol-specific actions such as HTTP header suppression or injection.
- Inspection for known XML threats.

This is not an exhaustive list, but gives an idea of some of the actions that occur upon receiving a message. The results of these pre-processing steps could result in the message being rejected before any message processing is even attempted.

2.1.2 Service Processing Phase

Once the client-side processing phase has completed and accepted the message, the message will be passed to the service's processing policy. This is often referred to as *Multistep processing*. A *Processing Policy* is a list of rules that contain actions that can be applied to a message. Actions are specific operations that are applied to a message such as encryption and decryption, message signing, authentication, etc. As the request message passes through the processing policy, the actions are applied to the message in a specified sequence, ultimately resulting in the message that will be passed to the server-side processing phase.

2.1.3 Server-Side (Back) Processing Phase

If the message makes it to this phase, it has been accepted by the client-side phase and processed by the service phase. It's now ready to be sent to the backend server. Before sending though, some additional steps may be required. Those steps may include:

- Establishing a new SSL connection to the back side server.
- Setting additional headers in the request.
- Mediating protocol versions (i.e. HTTP 1.1 to HTTP 1.0).
- Other protocol related tasks for WebSphere MQ, WebSphere JMS, FTP, NFS, etc.

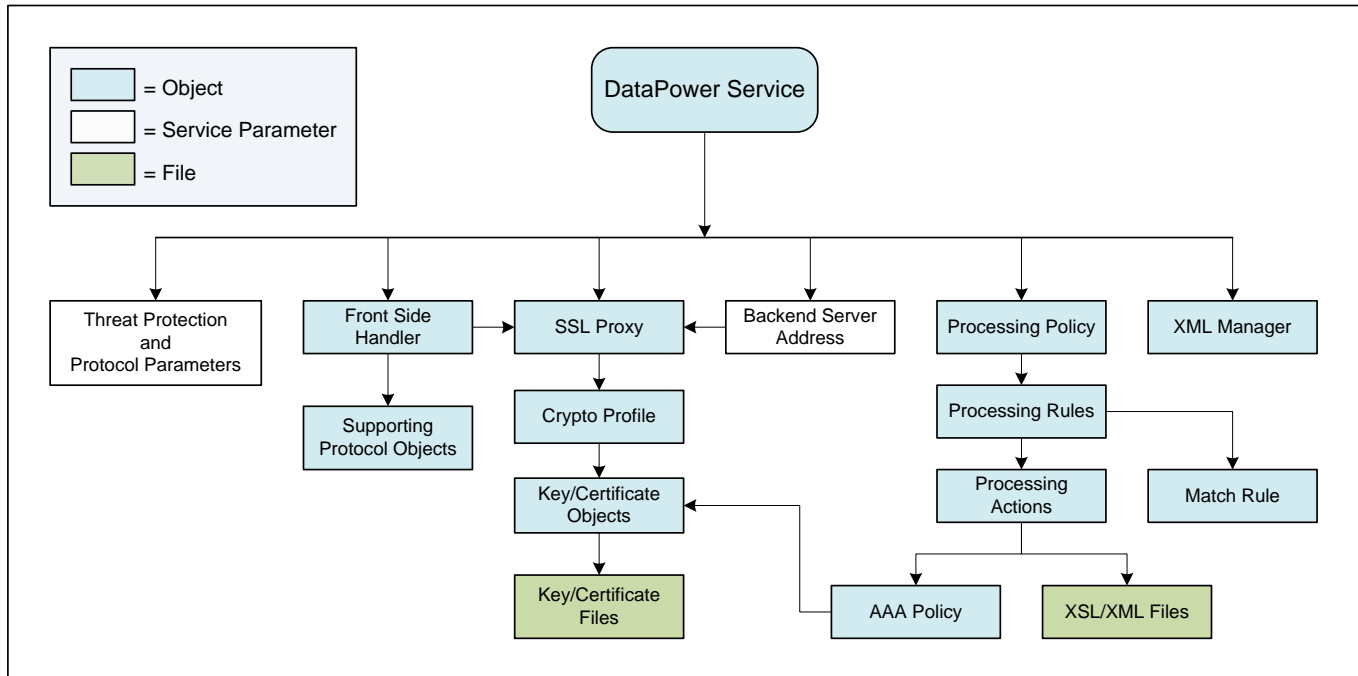
Once all of the server-side processing is complete, the message is sent to the backend destination.

2.1.4 Response Processing

When (and if) a response is received from the backend server, the three phases will occur again to verify the validity of the response, execute a processing policy, and then forward the response back to the original client. The processing phase can be configured to have separate rules for request and response processing.

2.1.5 WebSphere DataPower Configuration Architecture

A single WebSphere DataPower appliance has the ability to host numerous service configurations. The following diagram shows a top-level object hierarchy of a WebSphere DataPower service.



This diagram shows some of the objects associated with a given service. For example, the service could be a Multi-Protocol Gateway that you create for handling requests. The service will use a *Front Side Handler* object which identifies an IP address and port. It also includes an *SSL Proxy* object which includes the necessary objects for SSL encryption. The service has a *Processing Policy* (for the service processing phase), and that policy contains one or more *Processing Rules*, and each rule contains one or more *Processing Actions*. Some of the objects will be created for you as a by-product of configuration wizards, and others will be created by drag and drop actions within the WebGUI.

2.2 Creating the Multi-Protocol Gateway service

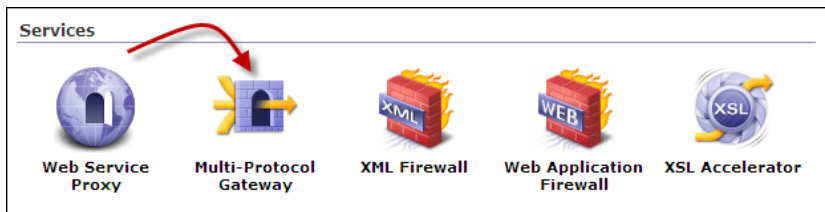
In this section, you'll be creating a service that will receive messages posted from your workstation, and perform a variety of actions against the message's XML payload. There are several steps you'll follow to create the service object:

- Specify the basic information about the Multi-Protocol Gateway Service.
- Create an HTTP Front Side protocol handler to handle HTTP requests.
- Create a Processing Policy and Processing Rule

To get things started, you'll create a service proxy that simply acts as a pass-thru. Whatever you post to the service proxy will get forwarded to an echo service running on the backend server. The response will pass back through your proxy and then be returned to your workstation.

The following steps will guide you through the process of creating and testing your service proxy. If you logged out from the WebGUI, log back in with your assigned user id and password. Make sure to select the matching domain for your user id.

- __1. If the control panel is not visible, click on the [Control Panel](#) link at the top of the left navigation pane.
- __2. Click on the Multi-Protocol Gateway icon.



- __3. Click the **Add** button to create a new Multi-Protocol Gateway service. The *Configure Multi-Protocol Gateway* form will be displayed.
- __4. In the *Multi-Protocol Gateway Name* field, type: `ProductServiceProxy`
- __5. In *Backend URL*, type: `http://demoserver:9080/ProductService/ProductService`



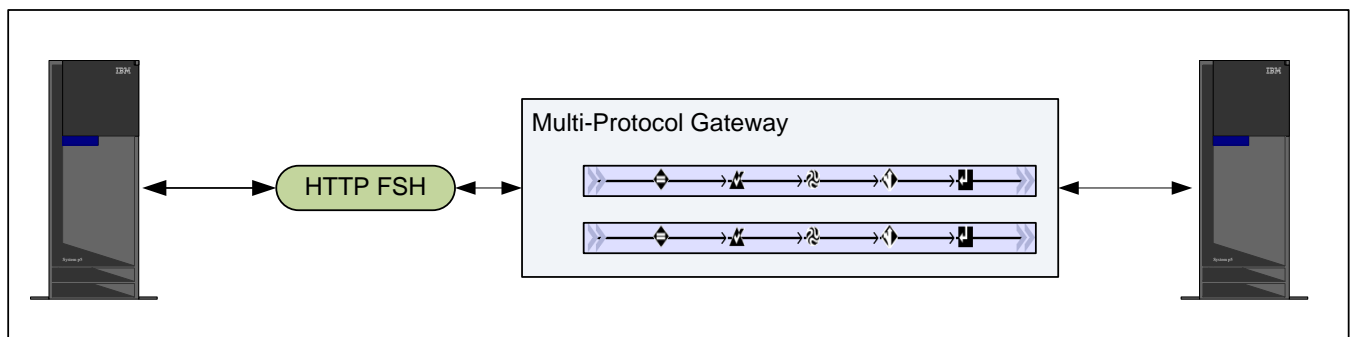
Important!

The URI portion of the URL is case sensitive. Make sure that you type the URL exactly as shown.

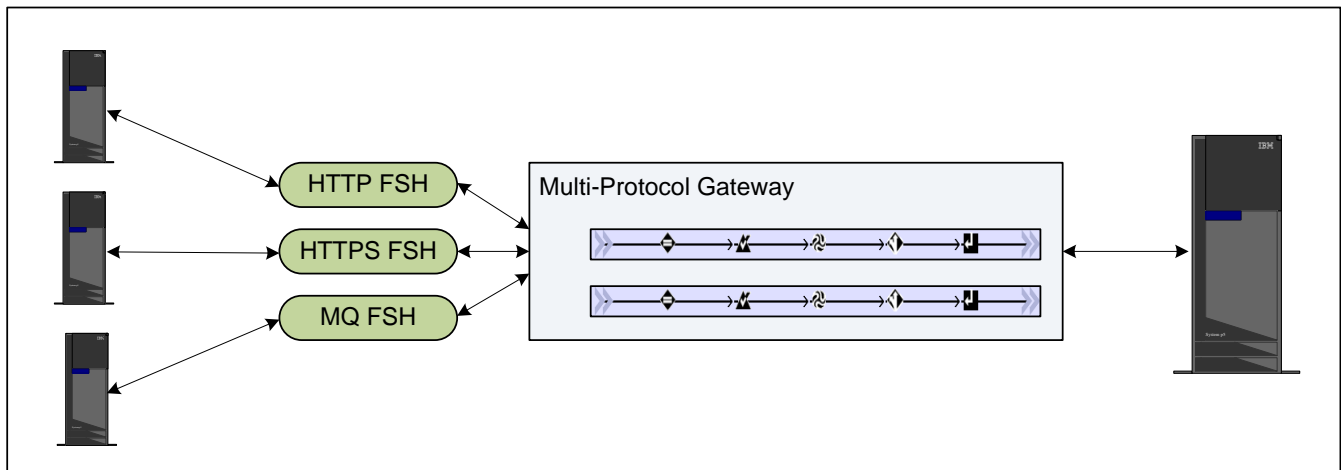
2.2.1 Creating the Front Side Handler (FSH)

The Multi-Protocol Gateway service employs one or more *Front Side Handlers* to manage all inbound traffic. In a simple configuration, there might be a single HTTP front side handler that listens for requests on a specific IP address and port.

In the scenario shown in the following illustration, requests arrive over HTTP and are received by the HTTP front side handler. The HTTP FSH will then pass the request to the Multi-Protocol Gateway (MPGW) for processing



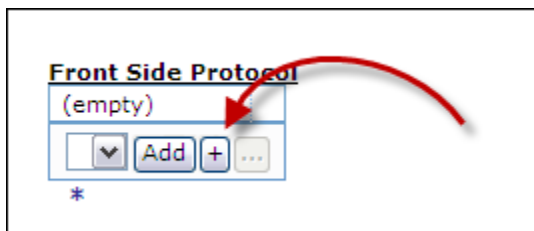
It's also possible to mix and match different types of protocols on the same multi-protocol gateway. For example, you can assign one FSH for HTTP, another for HTTPS, and yet another that acts as a WebSphere MQ client.



The server-side protocol is completely independent of the front-side and can be any of the protocols supported by the appliance.

For this lab exercise, you'll create a single HTTP Front Side Handler and assign it to the multi-protocol gateway.

- ___1. In the middle of the form towards the right is a section labeled *Front side settings*. Locate and click the plus (+) button to create a new front side handler.



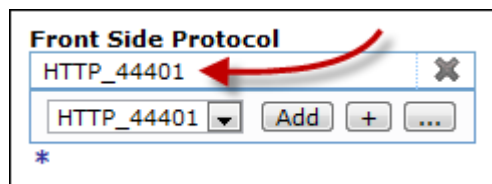
A pop-up list of front side handlers will be displayed. You can see from this list that the Multi-Protocol Gateway service supports many different front-side protocols.

- ___2. In the pop-up list of front side handlers, click: [HTTP Front Side Handler](#)

The options provided in the pop-up window allow you to precisely configure the various settings related to HTTP connections. In addition to the obvious settings such as IP address and port, you can also specify which version of HTTP that the listener will accept, or whether or not to use persistent connections.

- ___3. In the *Name* field, type `HTTP_444nn` where *nn* is your student number. For example, if you are student01, type the name `HTTP_44401`.
- ___4. Leave the Local IP Address field as 0.0.0.0. This will cause the front side handler to listen for traffic on all IP addresses defined on the appliance.

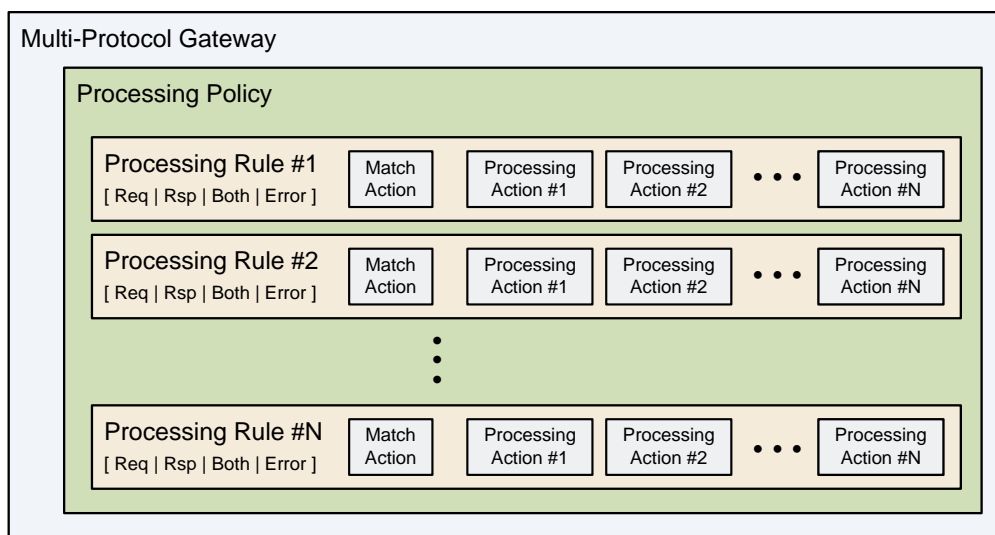
- ___5. In the *Port Number* field, replace the default port 80 with 444nn where *nn* is your student number.
- ___6. Click the **Apply** button in the upper left corner of the form. The new HTTP FSH should be automatically added to the list of Front Side Protocols (see below).



2.2.2 Processing Policies, Rules, and Actions

Each service that you configure will have exactly one *Processing Policy*. The processing policy defines what should happen when a message arrives from either the client (request), or the server (response).

A processing policy is comprised of one or more *Processing Rules*. A processing rule always begins with a *Match Action*, followed by one or more *Processing Actions*. Processing rules are identified as either request, response, both, or error types. A processing rule that is indicated as a request rule will be ignored during response processing. A processing rule that is identified as both will be evaluated for both requests and responses. Error rules are executed only when an error occurs during processing.



The *Match Action* references a *Match Rule* that contains one or more matching criteria (or expressions) that are evaluated to determine whether or not to execute the remaining actions in the processing rule. When more than one match expression is defined, the match rule can specify whether to combine them with Boolean AND or OR semantics. When the match rule is configured to use OR, only one of the match expressions must be True; when AND is specified, all expressions must evaluate to True.

Match Rule – evaluate statements using: AND | OR

Match Expression: URL HTTP Header XPath Error Code
Match Expression: URL HTTP Header XPath Error Code
⋮
Match Expression: URL HTTP Header XPath Error Code

Matching expressions can test the message in several ways. For instance, in this lab you'll be specifying a matching expression that inspects the request URI for a specific pattern. Matching rules support the following types of matching expressions:

- URL: A match template that inspects the URL for a specific pattern.
- HTTP: A match template that inspects the value of a specified HTTP header for a specific pattern.
- HTTP Method: A match template that compares the specified HTTP method (POST, GET, DELETE, etc.) against the value of the HTTP request line.
- Error Code: A match template that matches against specific error codes that may have been raised by previously executed processing rules.
- XPath: A match template that uses the specified XPath expression to inspect the contents of the XML message body.

When a message arrives into the processing policy, the policy will look at each processing rule, starting with the first one, and evaluate its associated match expression. If the match expression evaluates to True, the actions in that rule will be executed, otherwise the policy will look at the next rule. Once a match rule evaluates to True, no other match rules will be evaluated. Only one processing rule will be executed.

- ___1. In the *General Configuration* section of the form on the right side, locate the field labeled *Multi-Protocol Gateway Policy* and click the plus (+) to create a new processing policy.

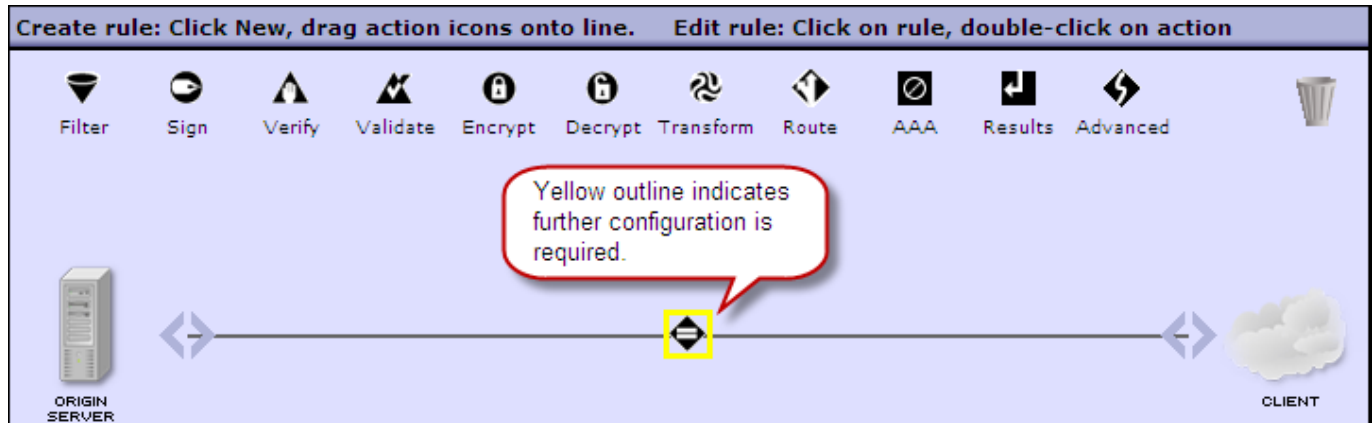
- ___2. In the *Policy Name* field at the top of the policy editor, type: `ProductServicePolicy`

In the following steps, you'll create a rule that will process client requests.

__3. In the *Rule* section, click on the **New Rule** button.

__4. In the *Rule Direction* dropdown, select: **Client to Server**

After you click the new rule button, a blank rule will be created that contains a match action.



For this lab, you'll create a match rule that will match on any inbound URI.

__5. Double click the match action to reveal its configuration form.

__6. In the *Configure a Match Action* form, click on the plus (+) button to create a new matching rule.

__7. In the *Configure Matching Rule* form, in the *Name* field, type: `MatchAnyURI`

__8. At the top of the form, click on the [Matching Rule](#) tab.

__9. At the bottom of the list of matching rules, click the **Add** button to create a new expression.

__10. Leave the *Matching Type* field as URL.

__11. In the *URL Match* field, type: * (The asterisk is a wildcard character that will match anything).

__12. Click the **Apply** button.

__13. In the *Configure Matching Rule* window, click the **Apply** button.

__14. In the *Configure a Match Action* window, click the **Done** button.

In the following steps, you'll create a rule that will process server responses.

__15. In the *Rule* section, click on the **New Rule** button.

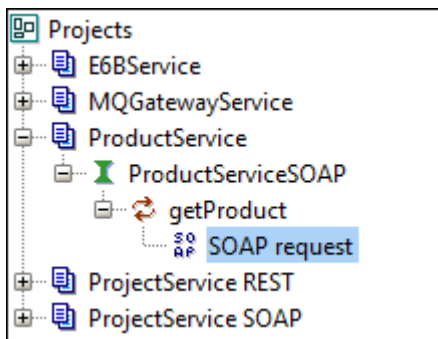
__16. In the *Rule Direction* dropdown, select: **Server to Client**

__17. Double click the match action to open its configuration form.

- __18. In the *Configure Matching Action* form, select the previously created **MatchAnyURI** rule from the dropdown list.
- __19. Click the **Done** button.
- __20. Click the **Apply Policy** button to save these changes. When you do this a *Results* action will be inserted into the processing rule.
- __21. Click the [Close Window](#) link in the upper right corner to dismiss the policy editor.
- __22. In the *Configure Multi-Protocol Gateway* form, click the **Apply** button to activate this new configuration.

You are now ready to test the service you just created.

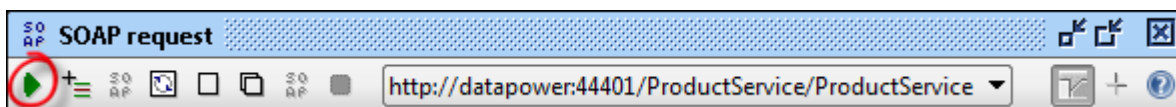
- __23. On the desktop, locate and launch the soapUI application.
- __24. In the project tree, expand the ProductService project until **SOAP request** is visible (see below).



- __25. Double click **SOAP request** to open the request window.
- __26. In the upper right corner of the soapUI window, click the maximize button to enlarge the request window.



- __27. In the endpoint dropdown, select: **http://datapower:444nn/ProductService/ProductService**
- __28. Dropdown the list again, then select **[Edit current...]**.
- __29. Update the port number by replacing *nn* with your student number, then click **OK**.
- __30. Click the green submit button to POST the request to ProductServiceProxy.



If everything worked properly, you should see getProductResponse in the Response tab.

If you received an error, you can try and determine the cause by looking at the logs. There's a convenient [View Log](#) link found towards the top of the Multi-Protocol Gateway configuration page. You can also view the logs from the main control panel by clicking on the View Logs icon.

At this point, you have created a multi-protocol gateway service that acts as a pass-thru and verified that it works. Now you'll add some more interesting functionality to the service.

2.2.3 Save the Running Configuration

Once you have your configuration running properly, it's a good idea to save the configuration to the flash memory. At this point, if the device was shut off or the power was disconnected, all of the work you've done until now would be lost. Saving the configuration causes your domain to be written to the flash memory, making it available after the device is restarted.

- ___1. At the top of the browser window, click on the [Save Config](#) link. You should see a message that says "Configuration successfully saved" (above the first row of service icons).

2.3 Schema Validation

An XML Schema describes the structure of an XML document. Validating an XML document against a schema is one step to assuring that the structure and content of the document is valid and safe. The process of validating an XML document against a schema is generally considered to be processor intensive, resulting in increased server load. For this reason, organizations often disable schema validation in an effort to reduce load (and cost) on application servers, especially when they are running on a mainframe. This is generally considered a security risk.

WebSphere DataPower SOA Appliances solve this problem by providing wirespeed schema validation to messages before they reach the application server. Messages that fail validation are rejected by default (this behavior can be customized).

In this section, you'll add a new processing rule to your service that will ultimately perform a variety of actions against the SOAP request.

Now you'll add a schema validate action to the processing rule. You'll configure the Validate action to use the embedded schema in the WSDL you uploaded in the first lab.

- ___1. Click on the ellipsis (...) button in the Multi-Protocol Gateway Policy field.



- __2. Expand the policy editor so that you can see all the configured rules at the bottom. Make sure the “Client to Server” rule is selected (it will be bold).

Rule:

Rule Name: Rule Direction:

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

Configured Rules

Order	Rule Name	Direction	Actions	
↑↓	ProductServicePolicy_rule_0	Client to Server	Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, Advanced	delete rule
↑↓	ProductServicePolicy_rule_1	Server to Client	Filter, Sign, Verify, Validate, Encrypt, Decrypt, Transform, Route, AAA, Results, Advanced	delete rule

- __3. Click and drag a Validate action and drop it to the right of the matching action.

- __4. Double click the new validate action (outlined in yellow) to provide the missing configuration details.

There are several methods listed for the Schema Validation method. This is a good opportunity to see the appliance's online help.

- __5. Move the mouse over the field label **Schema Validation Method**. You should notice that it is actually a hyperlink. Almost all field labels in the WebGUI are hyperlinks and when clicked, will pop up a help window to explain the various options for that field.
- __6. Click the [Schema Validation Method](#) label to show the help text. Close the help text window by clicking its close button.

- ___7. Select the radio button associated with: **Validate Document via WSDL URL**. Selecting this option causes DataPower to validate the message against the schema found within a WSDL.
- ___8. In the *WSDL URL*, make sure the upper dropdown contains **local:///**. In the lower dropdown list, select **ProductService.wsdl** that you previously uploaded. The Validate configuration window should look like the following image.

Schema Validation Method

- ☐ Validate Document via Attribute Rewrite Rule
- ☐ Validate Document via Schema Attribute
- ☐ Validate Document via Schema URL
- ☒ Validate Document via WSDL URL
- ☐ Validate Document with Encrypted Sections

WSDL URL

local:///

ProductService.wsdl

Upload... Fetch... Edit... View...

Var Builder

Asynchronous

☐ on ☒ off

- ___9. Click the **Done** button at the bottom of the window.
- ___10. Click the **Apply Policy** button at the top of the policy editor to activate your changes.
- ___11. Click the [Close Window](#) link in the upper right corner of the policy editor.

The WSDL's schema looks like the schema in the following listing. Notice that the product-id element restricts its values to the various WebSphere DataPower SOA Appliances models (XA35, XS40, etc.).

```
<xsd:element name="product-info">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="product-id">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="XA35"/>
            <xsd:enumeration value="XS40"/>
            <xsd:enumeration value="XI50"/>
            <xsd:enumeration value="XI52"/>
            <xsd:enumeration value="XB60"/>
            <xsd:enumeration value="XB62"/>
            <xsd:enumeration value="XM70"/>
            <xsd:enumeration value="XE82"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="brand" type="xsd:string"/>
      <xsd:element name="encoded-description" type="xsd:string"/>
      <xsd:element name="benefits" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```


- ___12. In soapUI, click the green submit button to POST the request again. The request should be successful as it was before. This indicates that the message successfully passed schema validation.
- ___13. In the Request tab, change the value of <product-id> to “1234”, then click the green submit button to post the message.

```
<getProductRequest xmlns="http://www.ibm.com/datapower/ProductService/">
  <Product>
    <brand>WebSphere DataPower</brand>
    <product-id>1234</product-id>
    <encoded-description>SUJNIFd1YlNwaGVyZSB5YXRhUG93ZXIgaWEklMCBTT0EgQXQ=
    <benefits>Security;Integration;Performance</benefits>
  </Product>
</getProductRequest>
```

Since “1234” is not a valid product-id, it failed schema validation resulting in a SOAP fault back to the client.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <env:Fault>
      <faultcode>env:Client</faultcode>
      <faultstring>Internal Error (from client)</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

The returned error message indicates that an internal error occurred but no other details are provided. This is by design to prevent malicious attackers from gaining detailed information about the underlying service. You can see detailed information about the failure in the DataPower log.

- ___14. In the Multi-Protocol Gateway configuration page, click on the [View Log](#) link towards the top right side of the page.

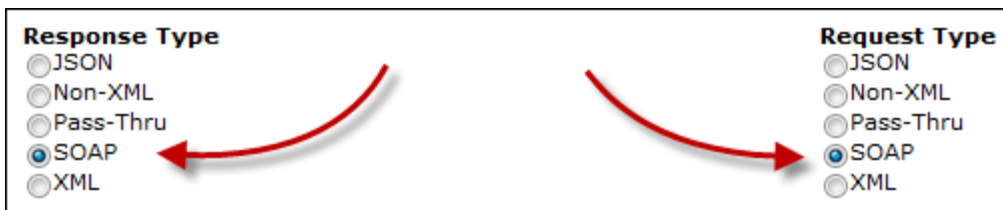
The log will reveal the underlying reason for the “Internal Error” message.

mpgw	error	28480		10.71.4.251	0x00230001	mpgw (ProductServiceProxy): Dynamic Execution Error
multistep	error	28480	request	10.71.4.251	0x80c00009	mpgw (ProductServiceProxy): request ProductServicePolicy_rule_0 #1 validate: 'INPUT wsdl local:///ProductService.wsdl' failed: http://10.71.17.13:44401/ProductService/ProductService: cvc-simple-type 1: element {http://www.ibm.com/datapower/ProductService/}product- id value '1234' is not a valid instance of the element type
multistep	error	28480	request	10.71.4.251	0x01d30003	mpgw (ProductServiceProxy): Schema Validation Error
schema	error	28480	request	10.71.4.251	0x80c00010	mpgw (ProductServiceProxy): Execution of 'local:///ProductService.wsdl' aborted: http://10.71.17.13:44401/ProductService/ProductService: cvc-simple-type 1: element {http://www.ibm.com/datapower/ProductService/}product- id value '1234' is not a valid instance of the element type

15. Close the log window by clicking on the Windows close button (upper right corner of window).

2.4 SOAP Envelope Schema Validation

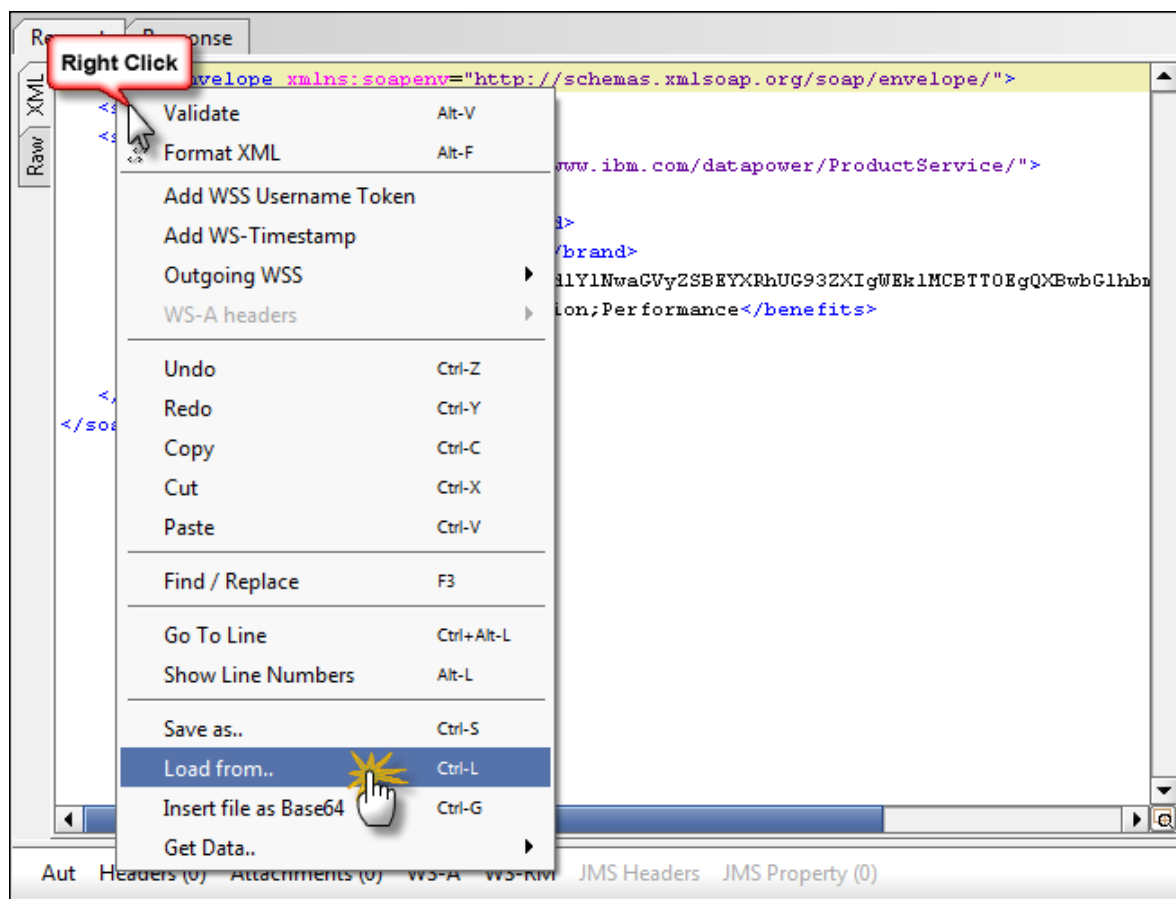
The Multi-Protocol Gateway service that you configured expects requests and responses to conform to SOAP standards. This setting is found towards the middle of the Multi-Protocol Gateway main configuration page (see following image).



Important!

The following steps show you how to reload the request payload with prebuilt SOAP messages. In future steps, these detailed steps will be omitted for brevity.

- ___1. In the soapUI Request tab, right click within the message body and select: **Load from...**



- ___2. In the *Load Editor Content* dialog, select `c:\labs\requests\noSoapEnv.xml`; then click **Open**.

- __3. Click the green submit button to POST the XML to ProductServiceProxy. The request should fail again. To see details about the failure, click on the [View Log](#) link in the Multi-Protocol Gateway configuration page.

2.5 Content-based Filtering

You can easily extend the built-in threat protection by defining custom filters. A custom filter is an XSL template that makes an “accept” or “reject” decision based on some custom logic that you define.

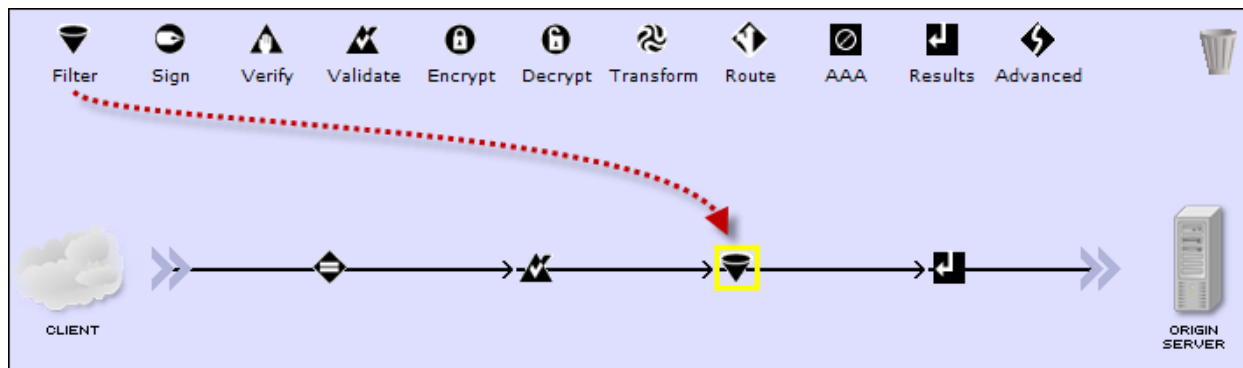
The “accept” and “reject” decision are accomplished using special built-in extension functions for XSL. The `<dp:accept>` and `<dp:reject>` extension functions are used to tell processing rule how to proceed with the message. The following XSL template inspects the `<brand>` element to make sure that it contains the word “DataPower”.

Listing of file: *customFilter.xsl*

```
<xsl:template match="/">
  <xsl:choose>
    <xsl:when test="contains(//prod:brand, 'DataPower')">
      <dp:accept/>
    </xsl:when>
    <xsl:otherwise>
      <dp:reject>Missing 'DataPower' trademark</dp:reject>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

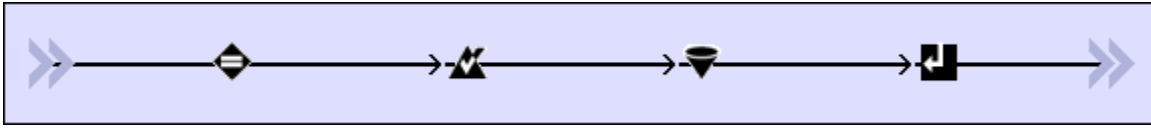
Now you'll add a filter action to your processing rule.

- __1. In the policy editor window, drag a filter action onto the rule as shown below.



- __2. Double click the yellow outlined filter action to complete its configuration.
- __3. In the *Transform* section:
- __a. In the upper dropdown, make sure `local:///` is selected.
 - __b. In the lower dropdown, select **customFilter.xsl**.
- __4. In the *Configure Filter Action* window, click **Done**.

The processing policy should now look like the following image.



- __5. Click the **Apply Policy** button to make your changes active.
- __6. In the soapUI request window, load the request from `c:\labs\requests\missingDp.xml`. Notice that the brand is missing the word “DataPower”.
- __7. Click the green submit button to POST the request to MyServiceProxy. You should receive a SOAP fault with an error message as shown in the following image.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <env:Fault>
      <faultcode>env:Client</faultcode>
      <faultstring>Missing 'DataPower' trademark (from client)</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

2.5.1 SQL Injection Threat Filtering

SQL Injection is an attack technique used to exploit Web sites and services that construct SQL statements from user-supplied input. For example, assume that a web service expects a SOAP request containing a <last-name> element used for looking up a customer.

```
<soap:Body>
  <customer-lookup>
    <last-name>KAPLAN</last-name>
  </customer-lookup>
</soap:Body>
```

The Web service uses an SQL statement with substitution parameters similar to the following SQL snippet:

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = ?
```

After the substitution takes place, the resultant SQL statement will be:

```
SELECT * FROM EMPLOYEE WHERE LASTNAME = 'KAPLAN'
```

However, if the value submitted in the <last-name> element contained a malicious SQL injection threat, it may look like this:

```
<soap:Body>
  <customer-lookup>
    <last-name>KAPLAN' OR '1'='1</last-name>
  </customer-lookup>
</soap:Body>
```

The SQL statement would become:

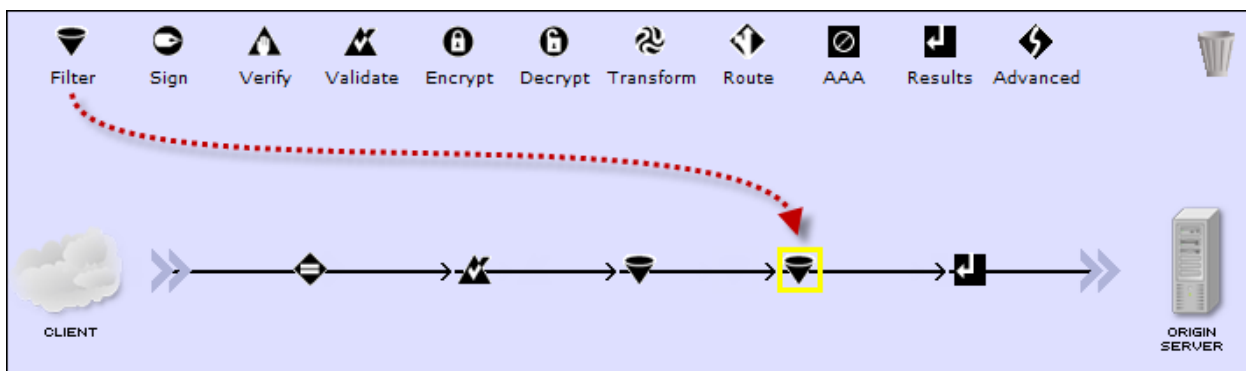
```
SELECT * FROM EMPLOYEE WHERE LASTNAME = 'KAPLAN' OR '1' = '1'
```

The service will return the details about ALL employees, since the WHERE clause will evaluate to true for every record in the EMPLOYEE table (because of the '1' = '1' clause).

WebSphere DataPower SOA Appliances can protect against such SQL injection threats using a special SQL injection threat filter. It works the same way as the filter you tried in the previous steps, except that the logic is a bit more complex.

The SQL Injection Threat filter has two parts: the base stylesheet filter (that uses <accept/> and <reject/>), and an XML file that contains the various patterns to search for. Keeping the patterns in a separate XML file allows you to create more customized patterns.

- __1. In the policy editor window, drag another Filter action onto the processing rule to the right of the previously added filter action.



- __2. Double click the yellow outlined filter action to complete its configuration.
- __3. In the *Transform* field:
 - __a. Change the upper dropdown to show: **store:///**
 - __b. In the lower dropdown box, select: **SQL-Injection-Filter.xsl**

- __4. Click the **Done** button.
- __5. Click the **Apply Policy** button to activate these changes.

The policy will now protect against malicious SQL injection threats. The file `sqlThreat.xml` contains a SOAP message with an SQL Injection Threat in it. The contents of the `<brand>` element contain the threat:

```
<product-info>
  <product-id>XI50</product-id>
  <brand>DataPower' or '1'='1</brand>
  <encoded-description>{omitted}</encoded-description>
  <benefits>Security;Integration;Performance</benefits>
</product-info>
```

- ___6. In the soapUI request window, load the request from `c:\labs\requests\sqlThreat.xml`.
- ___7. Click the green submit button to POST the message to `ProductServiceProxy`. The request should fail due to “Message contains restricted content (from client)”.

2.6 Transforming with XSL and XPath

At the heart of WebSphere DataPower SOA Appliances is a high speed XSL compiler and execution engine. In fact, most built-in functionality is engineered using XSL. Some of the built-in stylesheets can be found in the *store* directory. XSL developers can easily copy and modify the IBM provided stylesheets to create new functionality or support emerging standards before IBM makes them available.

When a stylesheet is referenced for the first time, it is compiled using a patented optimizing XSL compiler for execution on specialized WebSphere DataPower hardware, then cached in memory for high-speed recall and execution.

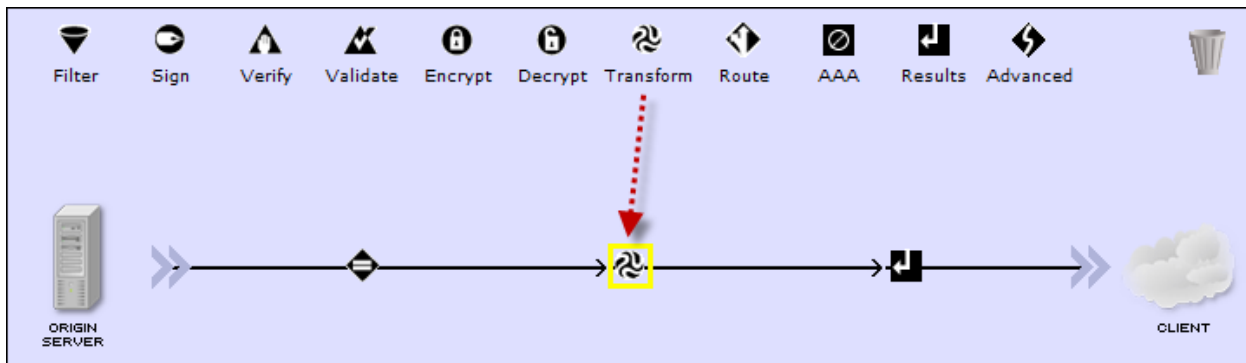
IBM has augmented XSL with a rich set of *extension functions* that enable you to easily add complex processing functionality to your processing rules. For example, there are extension functions for performing base-64 encoding and decoding, encryption and decryption, and date/time functions. There are also functions for communicating with off-box web services as well as LDAP servers.

In this section, you’ll be introduced to how XSL templates are used within processing rules. You’ll also get a chance to see the `decode()` extension function for decoding base-64 encoded text.

In the following steps, you’ll add a transform action to the response (server to client) rule instead of the request rule. Since the transform action will modify the overall structure of the message, it won’t match the schema that the backend service is expecting, therefore the request will fail. To avoid this, you’ll modify the response which is destined back to soapUI.

- ___1. In the policy editor, towards the bottom, click on the **Server to Client** rule to make it the active rule in the editor.

- __2. Click and drag a transform action and drop it after the match action.



- __3. Double click the yellow outlined transform action to expose its configuration settings.

For this transform, the stylesheet will be located on a remote HTTP server rather than in your local directory.

- __4. In the *Transform* field:
- __a. In the top dropdown, select **http://**.
 - __b. In the lower text box, type: `demosever/files/productTransform.xsl`
- __5. Click the **Done** button to save the transform action.
- __6. Click the **Apply Policy** button to apply the changes to the overall policy.
- __7. Click the [Close Window](#) link to dismiss the policy editor.
- __8. Click the **Apply** button in the Configure *Multi-Protocol Gateway* form.

You're now ready to run another transaction through your multi-protocol gateway service. Before you do that, let's take a look at what the XSL template will do to the message.

Here's the SOAP body of the response message. Notice the `<encoded-description>` tag contains base-64 encoded text (some of it has been omitted).

```
<soap:Body>
  <getProductResponse>
    <Product>
      <product-id>XI50</product-id>
      <brand>WebSphere DataPower</brand>
      <encoded-description>SUJNIFd1YlNw {omitted}</encoded-description>
      <benefits>Security;Integration;Performance</benefits>
    </Product>
  </getProductResponse>
</soap:Body>
```

The productTransform.xsl template looks for two different patterns:

- When a <encoded-description> tag is encountered, it will change it into a <description> tag and then decode the original tag's value. dp:decode() is an extension function that will perform the base-64 decoding.
- When a <benefits> tag is encountered, it will use the str:tokenize() function to tokenize the list of benefits (delimited by semicolons) into a small XML tree.
- An identity transform is found at the end of the template, which will match anything else that hasn't explicitly been matched, and copy it to the output document.

Partial Listing of file: *productTransform.xsl*

```
<xsl:template match="encoded-description">
  <description>
    <xsl:value-of select="dp:decode(., 'base-64')"/>
  </description>
</xsl:template>

<xsl:template match="benefits">
  <xsl:variable name="benefits" select="str:tokenize(., ';')"/>
  <benefits>
    <xsl:for-each select="$benefits">
      <benefit><xsl:value-of select="."/></benefit>
    </xsl:for-each>
  </benefits>
</xsl:template>
```

- ___9. In the soapUI request window, load the request from `c:\labs\requests\soapMsg.xml`.
- ___10. Click the green submit button to POST the message to ProductServiceProxy, then inspect the response. Notice that the <encoded-description> tag was replaced with a <description> tag, and that its contents are no longer base-64 encoded. Also, the benefits list was properly expanded into a multi-element <benefits> group.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:prod="http://www.ibm.com/datapower/ProductService">
  <soapenv:Body>
    <getProductResponse xmlns="http://www.ibm.com/datapower/ProductService">
      <Product>
        <product-id>XI50</product-id>
        <brand>WebSphere DataPower</brand>
        <prod:description xmlns="http://www.ibm.com/datapower/ProductService">IBM WebSphere DataPower XI50 SOA Appliance</prod:description>
        <prod:benefits xmlns="http://www.ibm.com/datapower/ProductService">
          <prod:benefit>Security</prod:benefit>
          <prod:benefit>Integration</prod:benefit>
          <prod:benefit>Performance</prod:benefit>
        </prod:benefits>
      </Product>
    </getProductResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

- ___11. If you've gotten everything working properly, you can save your configuration by clicking the [Save Config](#) link in the top of the browser window.

2.7 Stylesheet Caching

XSL stylesheets are compiled and then cached to improve performance. Previously you configured your processing rule to transform the request XML document against `productTransform.xsl`. The stylesheet was fetched from a remote server, compiled, and then cached. You can verify this by checking the status of the document cache.

- __1. In the left hand navigation pane, under the *Status* menu, scroll down to find the *XML Processing* section, and click on [Stylesheet Cache](#).

In the *cached stylesheets* column, you can see the number of stylesheets that have been compiled and cached (this value also includes some system stylesheets).

- __2. In the *XML Processing* section, click on [Stylesheet Status](#)

The *Stylesheet Status* page shows you all of the stylesheets that have been compiled and cached. Since schema documents (XSD) are compiled like stylesheets, they show up in this list too.

2.8 Implicit XML Threat Protection

Services that are configured to receive XML messages provide a wide array of additional XML threat protection.

2.8.1 Malformed XML Content Detection

In this next step, you'll post malformed XML to your service

- __1. In soapUI, click on the Request tab.
- __2. Load the request body with file `c:\labs\requests\malformed.xml`.

Notice that the closing `<Product>` tag is missing the leading slash resulting in the XML to be malformed. Don't correct the error.

- __3. Click the green submit button to POST the malformed message to `ProductServiceProxy`. Again, you should receive a generic SOAP fault. Click the [View Log](#) link at the top of the Multi-Protocol Gateway configuration page and notice the error message pertaining to the mismatched tag.

2.8.2 XML Denial of Service (XDoS)

WebSphere DataPower appliances can protect Web services against both single message denial of service (XDoS) and multiple message denial of service (MMXDoS) attacks.

Single message XDoS attacks may have any combination of the following characteristics:

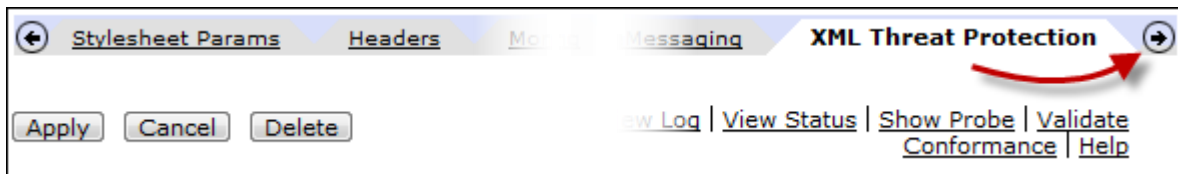
- Jumbo payloads – Sending a very large XML message to exhaust memory and CPU on the target system.
- Recursive elements – XML messages that can be used to force recursive entity expansion (or other repeated processing) to exhaust server resources

- MegaTags – Otherwise valid XML messages containing excessively long element names, or an excessive number of tags. This attack may also lead to buffer overruns.
- Coercive parsing – XML messages specially constructed to be difficult to parse, resulting in excessive resource consumption in the target machine.
- Public key DoS – Utilizing the asymmetric nature of public key operations to force resource exhaustion on the recipient by transmitting a message with a large number of long-key-length, computationally expensive digital signatures.

Multiple message XDoS (MMXDoS) attacks may have the following characteristics:

- XML flood – sending thousands of otherwise benign messages per second to tie up a Web service. This attack can be combined with Replay attack to bypass authentication, and with Single message XDoS to increase its impact.
- Resource hijack – sending messages that lock or reserve resources on the target server as part of a never-completed transaction.

- __1. At the top of the Multi-Protocol Gateway configuration form is a set of tabs. At the right and left side of the tabs are arrow images. Moving the cursor over the arrow (without clicking) will cause the tabs to shift left or right. Move the mouse over the right arrow until the [XML Threat Protection](#) tab is visible.



- __2. Click on the [XML Threat Protection](#) tab.
- __3. In the *Single Message XML Denial of Service* section, click the **on** radio button for *Gateway parser limits*.
- Notice that the XDoS protection is highly customizable.
- __4. Click the **off** button for *Gateway parser limits*.
- __5. In the *Multiple Message XML Denial of Service* section, click the **on** radio button for *Enable MMXDoS Protection*.
- __6. Click the **off** button for *Enable MMXDoS Protection*.

2.8.3 Virus Scanning

Viruses are typically contained in message attachments. XML Virus Protection sets parameters that handle the following types of attacks in attachments:

- XML virus attacks
- XML encapsulation attacks
- Payload hijack attacks
- Binary injection attacks

There are two levels of protection against virus threats.

- The first level is to determine whether or not to allow attachments. This is accomplished on the [XML Threat Protection](#) tab that is currently displayed in your browser in the *XML Virus (X-Virus) Protection* section.
- If attachments are allowed, the second level of protection occurs in the processing rule. A special “Virus Scan” action will extract the attachment from the message and send it to an Internet Content Adaption Protocol (ICAP) compatible virus scanner. If the scanner responds that a virus exists in the attachment, the virus scanning action will either strip the attachment or reject the message (based on configuration settings).

2.9 Summary

In this lab, you learned:

- That there are three service processing phases that occur each time a message arrives. The client-side processing performs all protocol related processing as well as shielding against a variety of malicious attacks. The service processing phase contains all of the specific rules and actions that you define. The server-side processing phase is where any backside protocol tasks are performed before the message is forwarded to the intended destination.
- WebSphere DataPower configurations are built using a pure object-oriented design. Every configuration object, such as a SSL Proxy Profile or a Processing Policy, can be reused.
- How to configure a Multi-Protocol Gateway service, along with an HTTP Front Side protocol handler, and a Processing Policy.
- A Processing Policy contains a set of Processing Rules; each rule begins with a Match Action that is evaluated to determine whether the rule should be executed. Each processing rule contains a list of processing actions that are executed against the message.
- Match rules can match on various aspects of a message, including the URL, HTTP headers, error codes, or an XPath that inspects the XML payload of a message.
- Clicking the [Save Config](#) link in the top navigation area will save your running configuration (for your domain) to the flash memory. The running configuration and the saved configuration are independent.
- How to add a schema validate action to the processing rule by dragging it from the action palette onto the processing rule. WebSphere DataPower appliances can perform schema validation against messages at near wire-speed, adding minimal latency to the overall transaction time.
- SOAP requests and responses are automatically checked against a SOAP schema, assuring that the SOAP envelope is well-formed and correct.
- Request and response XML documents are checked to assure they are well-formed. Malformed XML is rejected which assures backend applications receive only well-formed XML.
- Custom Filters can be used for content-based message filtering and SQL injection threat protection.
- How to transform an XML document using a Transform action.
- XSD and XSL stylesheets are compiled and cached.

Lab 3 Securing XML Message Content using WS-Security

Prerequisites: This lab requires the completion of labs 1 and 2.

In this lab, you'll be adding a few new processing rules to your multi-protocol gateway's processing policy to demonstrate various security features.

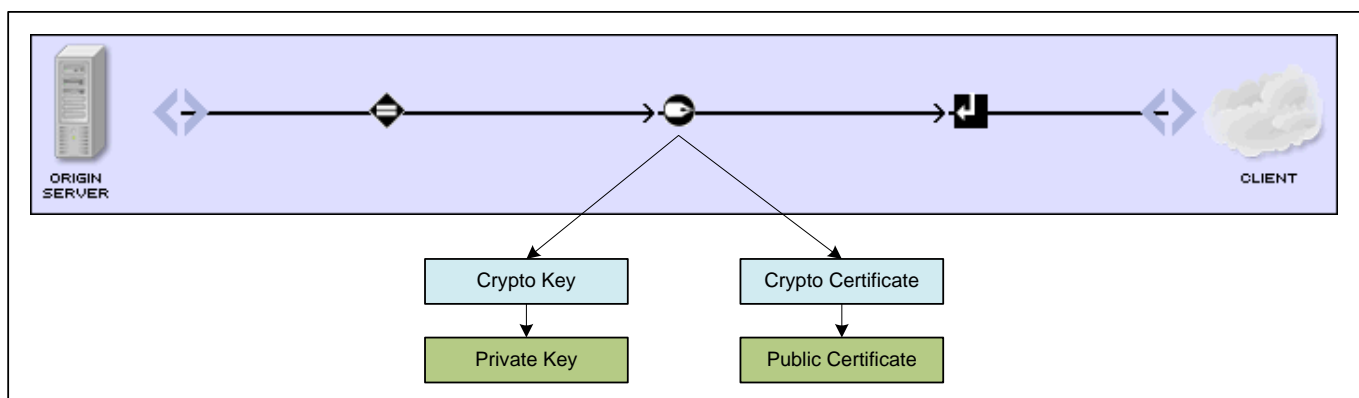
Upon completing this lab, you'll have a better understanding of:

- Private keys and public certs.
- How WebSphere DataPower handles digital keys and certificates.
- Support for WS-Security digital signatures, encryption, and decryption.
- Field-level encryption.
- The built-in authentication and authorization framework.
- Connecting to an LDAP server.
- Configuring SSL.

3.1 Public Key Infrastructure (PKI)

In the digital world, public and private keys are often employed to perform cryptographic operations, such as encryption of message data. The use of key pairs (public/private) is known as asymmetric encryption. It is vital that the private key is protected, while its public counterpart, the public key (often carried in a certificate), can be freely distributed. Certificates are typically validated by a Certificate Authority (CA). In the event that an authority needs to revoke a previously distributed certificate, it adds the revoked certificate to a globally published certificate revocation list (CRL).

On DataPower, public certificates and private keys are wrapped in *crypto objects* so that there is one level of indirection when using them. For example, when you upload a public certificate, it will be wrapped in a *Crypto Certificate* object. When a service object needs to use that public certificate, it will reference it using the crypto certificate instead of the actual certificate file. The following image shows a signing action that references a crypto key and crypto cert when digitally signing a message.



This single level of indirection allows the underlying key or certificate to be replaced without the need to reconfigure any services that are using it.

3.2 WS-Security Digital Signatures

The term “digital signature” refers to something created using public key technology. Two keys are involved: a private key, which only you know, and a corresponding public key, which you can freely give to anyone. What makes this key pair interesting is that anything encrypted using one key can only be decrypted with the other one.

The primary usage of digital signatures is to verify the integrity of a transmitted message. When a message travels over public networks, it can be intercepted, modified, and then forwarded without detection. Adding a digital signature to a message enables the recipient of the message to determine whether the message has been altered along the way.

For example, assume a business partner wants to send you a digitally signed message. First, they will compute a special checksum on the message they want to send (this is often referred to as a message digest). Then they encrypt the digest with their private key. The result is a digital signature for the message. They send this digital signature along with the original message to you.

When you receive the message, you'll first compute the message digest on the received message. You'll then use the sender's public key to decrypt the message digest that was sent along with the message. If the message digest that you calculated and the one that you decrypted are identical, then you can be certain that the data wasn't changed in transit (integrity) and that the data was signed by the business partner (authentication).

Creating and verifying digital signatures involve a considerable amount of mathematical computations, and are thus very processor intensive. WebSphere DataPower employs cryptographic hardware to do these calculations, thus freeing up costly processor cycles for business-related tasks.

In this section, you'll configure your ProductServiceProxy to:

1. Verify a digital signature generated by soapUI. If the verification fails, reject the request.
2. Strip the digital signature from the request before forwarding it to the backend service.
3. Add a digital signature to the service's response and return the signed response to soapUI.

3.2.1 Crypto objects

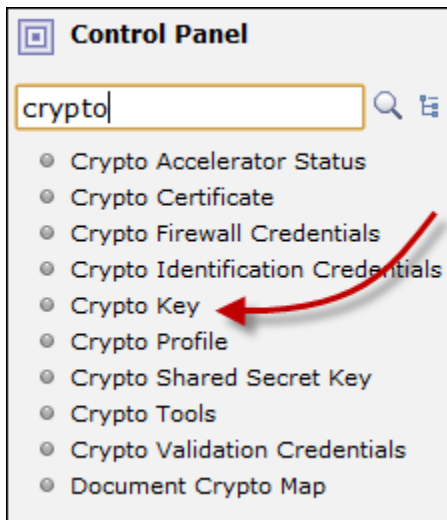
In lab 1, you uploaded several PKI files. Keys and certificates generally have expiration dates, thus requiring occasional replacement. To avoid the need to change multiple configurations each time a key or certificate needs updating, keys and certificates are wrapped in *Crypto Key* and *Crypto Certificate* objects. These crypto objects add a level of indirection to the underlying keys and certs, averting the need to update affected configurations each time key or certificate maintenance occurs.

The following steps will guide you in creating a Crypto Key object that wraps ProductService-privkey.pem, and Crypto Certificates that wrap ProductService-sscert.pem and soapUI.pem.

3.2.2 Create the Crypto Key and Cert to represent ProductServiceProxy

In previous steps, you either used the icons in the main control panel, or expanded the navigation tree to locate a specific object. In the following steps, you'll use the search function to quickly find the object you're interested in.

- __1. In the search field above the navigation tree, type the word "crypto" (case is not important). As you type, the results of the search will replace the navigation tree.



- __2. In the search results, locate and select: [Crypto Key](#)
- __3. Click the **Add** button to create a new crypto key.
- __4. In the *Name* field, type: `ProductServiceCryptoKey`
- __5. In the *File Name* field (lower dropdown), select: **ProductService-privkey.pem**
- __6. Click the **Apply** button.
- __7. In the left navigation, locate and select [Crypto Certificate](#)
- __8. Click the **Add** button to create a new crypto certificate.
- __9. In the *Name* field, type: `ProductServiceCryptoCert`
- __10. In the *File Name* field (lower dropdown), select: **ProductService-sscert.pem**
- __11. Click the **Apply** button.

3.2.3 Create the Crypto Cert that wraps soapUI's certificate

- __1. In the left navigation pane, locate and select [Crypto Certificate](#). If you cannot find it, you can use the navigation search box.

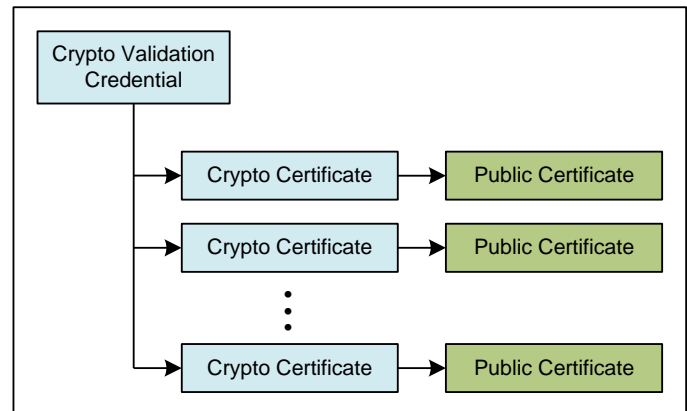
- __2. Click the **Add** button to create a new crypto certificate.
- __3. In the *Name* field, type: `SoapUICryptoCert`
- __4. In the *File Name* dropdown, select: **soapUI-sscert.pem**
- __5. Click the **Apply** button.

3.2.4 Verifying a Digital Signature

The process of securely verifying a digital signature requires that the recipient of the message have access to the signer's public certificate. The certificate is often included in the signed message, but the most reliable way of verifying the signature is with a certificate provided by the signer and uploaded to the cert: directory.

3.2.5 Crypto Validation Credential

Earlier, you created a crypto certificate object which wrapped a single PKI certificate. Consider the case where you need to create a processing rule that will verify a digital signature, but the signer may be one of many different business partners. Creating separate processing rules for each partner would be cumbersome and subject to constant modification when partners were added or dropped. The crypto validation credential object has the ability to group many crypto certificates together into a single object. With a crypto validation credential (often referred to as a valcred), you can create a single processing rule with a single signature verification action that will accommodate countless public certificates. Certificates can be added and removed from the validation credential independent of any verification actions that use it.



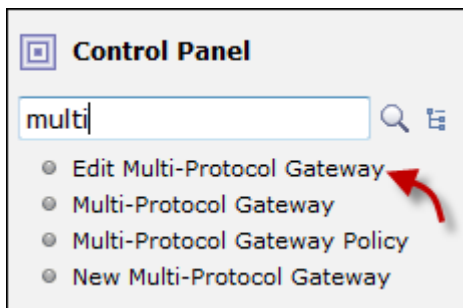
3.2.6 Creating a Crypto Validation Credential

- __1. In the search field above the navigation tree, type the word "crypto". In the search results, locate and select: **Crypto Validation Credentials**
- __2. Click the **Add** button to create a new validation credential object.
- __3. In the name field, type: `ProductSvcConsumersValcred`
- __4. In the *Certificates* field, dropdown the certificate list and choose: **SoapUICryptoCert**
- __5. Click the **Add** button to add the certificate to the list of certificates.
- __6. Click **Apply** to save the new validation credential object.

3.2.7 Verifying the request signature and signing the response

In the next steps, you'll modify the ProductServiceProxy Multi-Protocol Gateway to verify the digital signature sent by soapUI, and to put a digital signature on the response sent back to soapUI.

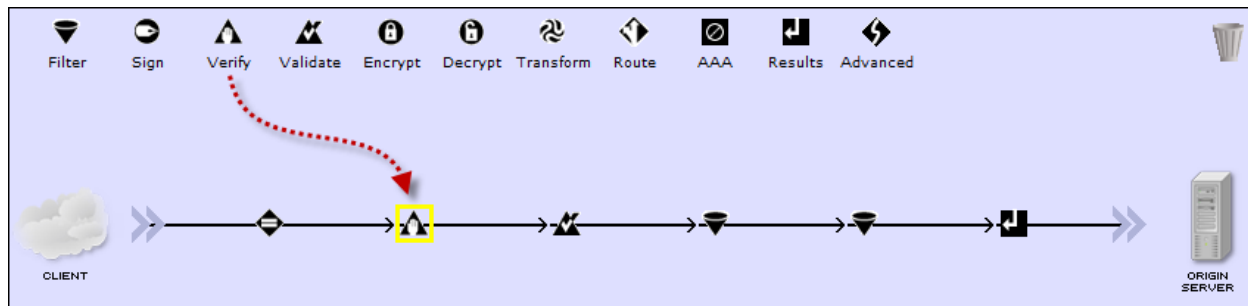
- ___1. In the search field above the navigation tree, type the word **multi**.



- ___2. In the search results, locate and click on: **Edit Multi-Protocol Gateway**
- ___3. Select [ProductServiceProxy](#) from the list.
- ___4. As you've done before, open the policy editor window by clicking on the ellipsis in the *Multi-Protocol Gateway Policy* field.

3.2.8 Verify the request's digital signature

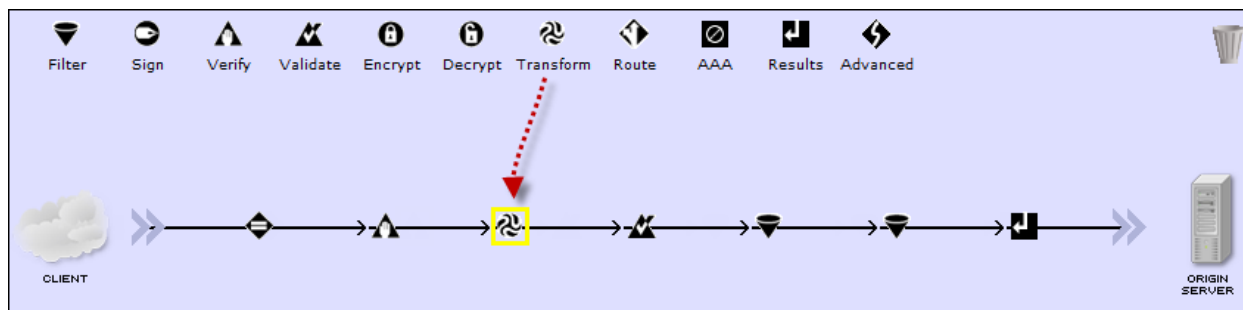
- ___1. Drag a Verify action onto the processing rule after the match action.



- ___2. Double click the yellow outlined verify action to complete its configuration.
- ___3. In the *Validation Credential* field's dropdown list, select: **ProductSvcConsumersValcred**. This validation credential contains soapUI's public certificate which will be used when verifying the digital signature.
- ___4. Click **Done**.

In the following steps, you'll add a transform action that will strip off the digital signature that soapUI created. This is not required, but is often done in order to reduce the overall message size. WebSphere DataPower appliances come with a library of pre-built stylesheets that perform various useful tasks such as stripping a digital signature. These pre-built stylesheets can be found in the store: directory.

- __5. In the policy editor, drag a transform action after the verify action.



- __6. Double click the yellow outlined transform action to complete its configuration.

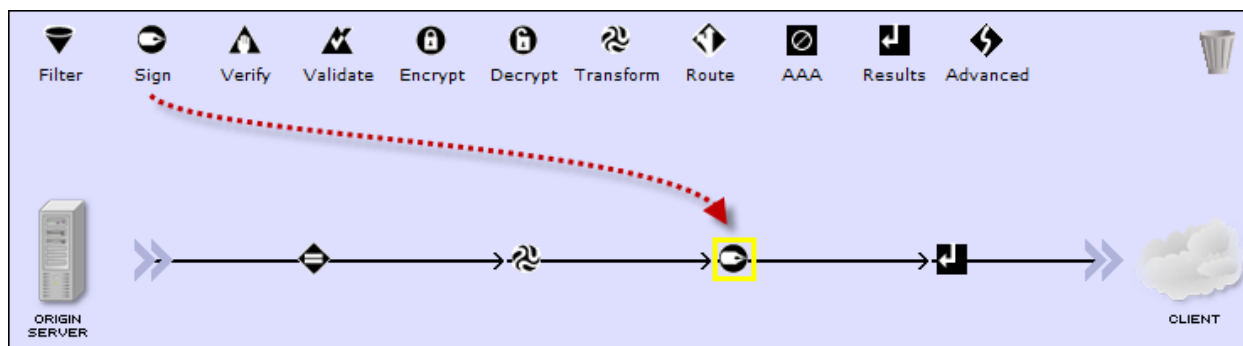
- __7. In the *Transform* field:

- __a. Select **store:///** in the upper dropdown box.
- __b. In the lower dropdown, select **strip-wssec-signature.xsl**

- __8. Click the **Done** button.

3.2.9 Sign the response

- __1. In the configured rules section in the policy editor (at the bottom), click on the **Server to Client** rule to make the response rule active in the editor.
- __2. Drag a Sign action onto the processing rule to the right of the transform action.



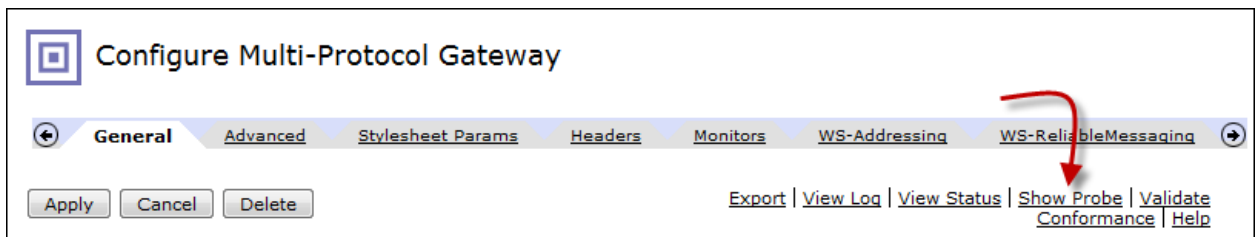
- __3. Double click the yellow outlined sign action to show its configuration.
- __4. In the *Key* dropdown, select **ProductServiceCryptoKey**
- __5. In the *Cert* dropdown, select: **ProductServiceCryptoCert**
- __6. Click the **Done** button to complete the sign action configuration.
- __7. Click the **Apply Policy** button to make your changes to the policy effective.

- __8. Click the [Close Window](#) link to close the policy editor.

3.2.10 The Transaction Probe

The transaction probe is a troubleshooting tool that provides insight into the state of a transaction as it moves through the processing rule. It allows you to see what the input context is to each of the actions as well as the values of system variables, protocol headers, etc. It is the single most important tool to use when troubleshooting a service's policy.

- __1. In the *Configure Multi-Protocol Gateway* form, towards the upper right corner, click on the [Show Probe](#) link.

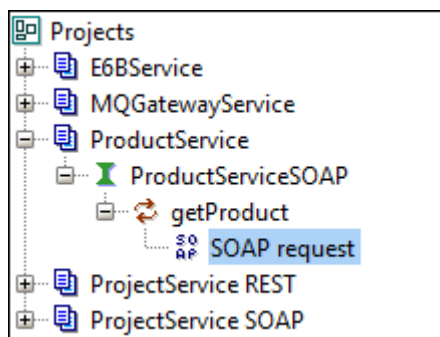


- __2. In the probe window, click on the **Enable Probe** button.
- __3. Click the **Close** button in the completion dialog.
- __4. Leave the Transaction List window open so you can easily access it in future steps. If you close the window by accident, you can always re-open it by clicking on the [Show Probe](#) link.

When you run transactions through the ProductServiceProxy gateway, the probe will capture all the details about the message.

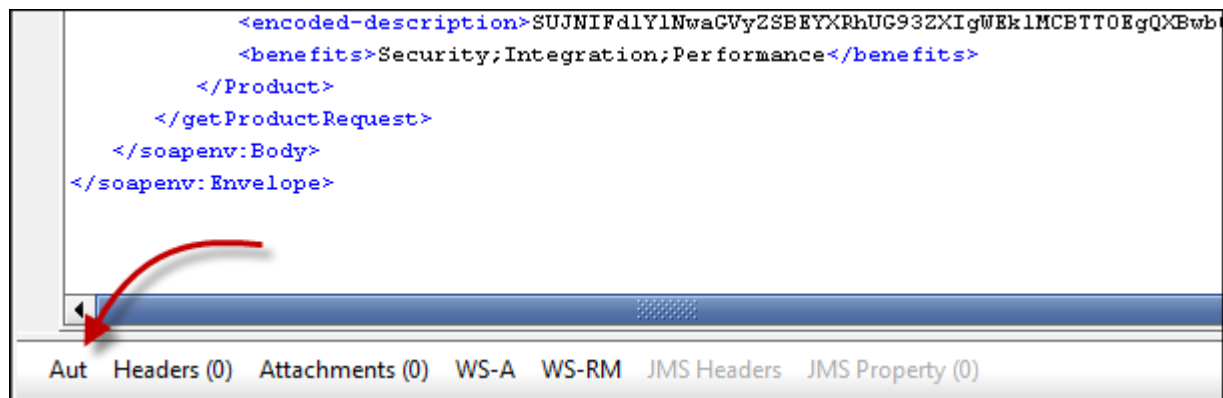
Now you'll tell soapUI to add a digital signature over the body of the request.

- __5. In soapUI, use the same request you did in the previous chapter. Expand the ProductService project until **SOAP request** is visible (see below), then double click it to open it.



- __6. Reload the request contents from file `c:\labs\requests\soapMsg.xml`.

7. At the bottom, locate and click the **Aut** button to reveal the authentication dialog.



8. Select the following from the Incoming and Outgoing WSS dropdowns:

a. In the *Outgoing WSS* dropdown, select **Sign**.

 b. In the *Incoming WSS* dropdown, select **Verify**.

Username:	<input type="text" value="david"/>
Password:	<input type="password" value="••••••"/>
Domain:	<input type="text"/>
Outgoing WSS:	<input type="button" value="Sign"/>
Incoming WSS:	<input type="button" value="Verify"/>

9. Click the **Aut** button to hide the authentication dialog.

___10. Click the green submit button to POST the request to your service proxy. The request should succeed.

__11. In the *Transaction List* (probe) window, click the **Refresh** button so you can see the transaction that you just posted.

The transaction list should now show the request you just posted. The [+] at the left side of the magnifying glass indicates that there is an associated response with the request.

__12. Click on the small [+] to show the response.

Refresh

Flush

Disable Probe

Export Capture

View Log

Send Message

Close

view trans#	type	inbound-url	outbound-url	rule	client-ip
<div><div></div><div></div></div> 118240	requ	http://10.71.17.13:44401/Produ	http://demosever:9080/Produc	ProductServicePolicy_rule_0	10.71.4.251
<div><div></div><div></div></div> 118240	respo	http://10.71.17.13:44401/Produ	http://demosever:9080/Produc	ProductServicePolicy_rule_1	10.71.4.251
	nse	ctService/ProductService	tService/ProductService		

- Click on the top (request) magnifying glass to show the execution details for that transaction. At the top of the displayed window will be a set of icons that represent each action in the rule that was executed. The main part of the window displays the contents of the INPUT context (the contents of the message sent by soapUI). Notice that soapUI properly added a digital signature to the message.

[Previous](#) **Input Context 'INPUT' of Step 1** [Next](#)

Step 1: Filter Action: Input=INPUT, Transform=store:///verify.xsl, NamedInOutLocationType=default, StylesheetParameters= , Transactional=off, SOAPValidation=body, SQLSourceType=static, Asynchronous=off, ResultsMode=first-available, RetryCount=0, RetryInterval=1000, MultipleOutputs=off, IteratorType=XPATH, Timeout=0, MethodRewriteType=GET, MethodType=POST, MethodType2=POST

Content [Headers](#) [Attachments](#) [Local Variables](#) [Context Variables](#) [Global Variables](#) [Service Variables](#)

Content of context 'INPUT':

```
<?xml version='1.0'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security>
```

3.2.11 DataPower Contexts

While configuring the various actions (sign, transform, etc.), you may have noticed that each action declares an input and an output context. In the case of a transform action, the input context will be the document that is fed to the transformer, and the results of the transformation will be written to the output context.

Some actions may only have an input context. For example, the verify action has an input context, but no output context; the signature verification either passes or fails. In contrast, some actions may only have an output context. The *fetch* action can fetch an XML document from a local file or a remote server, and the fetched document becomes the output context. Contexts are referred to by the following names:

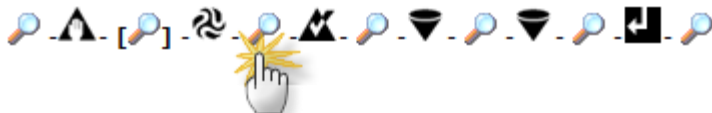
- **INPUT** – represents the original message as it arrived from the client.
- **OUTPUT** – represents the outbound message which will be forwarded to the destination. In the case of client-to-server processing, the OUTPUT context represents what will be sent to the backend server. In the case of server-to-client processing, the OUTPUT context represents what will be returned to the client.
- **NULL** – indicates that the output is not needed. In other words, the output from the action is sent to the bit bucket.
- **PIPE** – indicates that the output of the action should be piped into the input of the next action. In this case, the *input context* of the next action must also specify PIPE.
- **Named context** – in this case, you can assign a name to a context and use it at a later point in the processing rule. For example, a transform action can be configured with an input context of INPUT and an output context of `newRequest`. Later in the processing rule, another action can use `newRequest` as the input context.

__1. Click on the magnifying glass in front of the transform action.



The XML document shown in the window shows what will be fed into the transform action as the context document. In this case, the message with the digital signature will be the input context to the transformation.

__2. Click on the magnifying glass after the transform action (in front of the sign action).



The input to the schema validate action is the results of the prior transform action. In the content section, notice that the digital signature has been removed.

- ___3. Click on the last magnifying glass. It contains the contents of the OUTPUT context, which will be forwarded to the backend service.



- ___4. Close the transaction detail window.
- ___5. In the transaction list window, click on the magnifying glass to the left of the response.

view	trans#	type	inbound-url	outbound-url	rule	client-ip
	118240	request	http://10.71.17.13:44401/ProductService/ProductService	http://demoserver:9080/ProductService/ProductService	ProductServicePolicy_rule_0	10.71.4.251
	118240	response	http://10.71.17.13:44401/ProductService/ProductService	http://demoserver:9080/ProductService/ProductService	ProductServicePolicy_rule_1	10.71.4.251

The INPUT context shown in the transaction window shows the response that came back from the backend service.

- ___6. Click on the magnifying glass after the transform action. In the content section, you should that the transformation decoded the encoded description and expanded the benefits list into an XML nodeset.

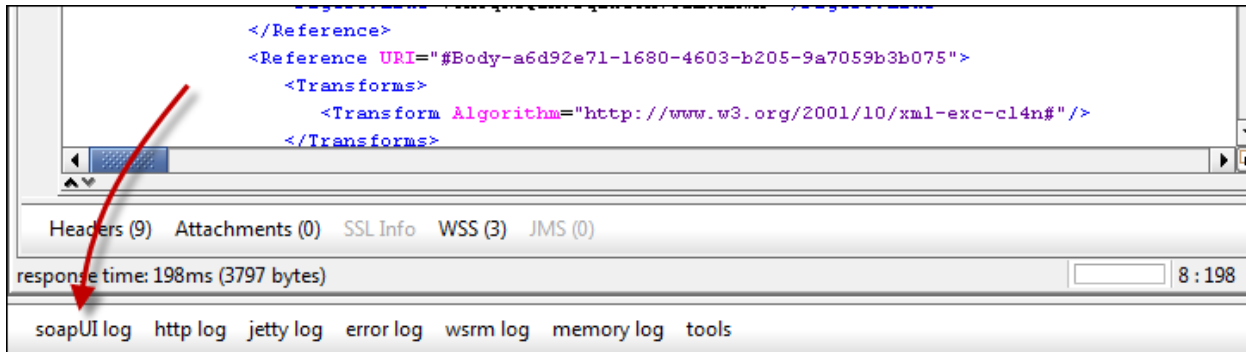


- ___7. Click on the magnifying glass after the sign action. In the content section, you should see that the message now contains a digital signature.



- ___8. Finally, click on the last magnifying glass. It represents the content that will be returned to soapUI.

- __9. You can verify that soapUI accepted the digital signature by looking at the soapUI log. At the bottom of the soapUI window is a button to show the soapUI log.



If soapUI could not verify the signature created by DataPower, the log would contain an error message.

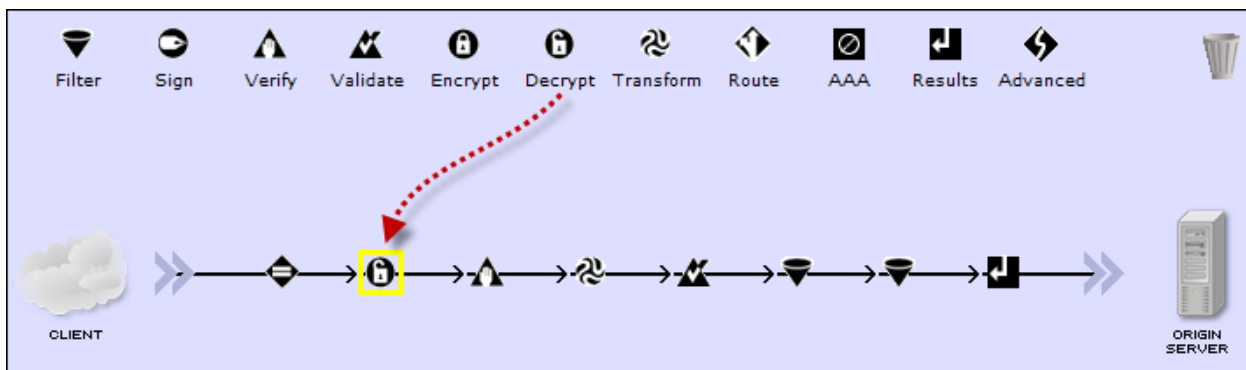
3.3 WS-Security Encryption & Decryption

Similarly to digital signatures, encryption use PKI keys and certificates for encryption and decryption. When encrypting a message, the recipient's public key is used; only the private key can decrypt the message.

3.3.1 Decrypting the request and encrypting the response

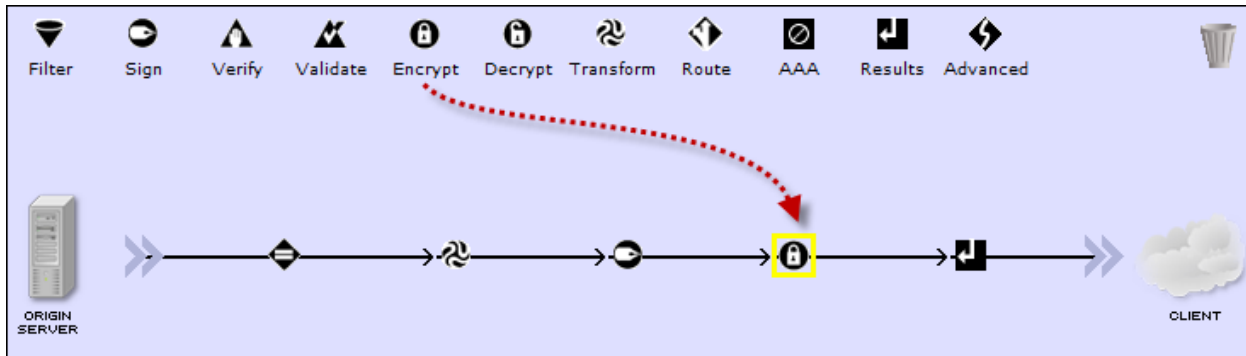
In the following steps, you'll add the necessary actions to decrypt the request (from soapUI) and then encrypt the response (going back to soapUI).

- __1. Reopen the policy editor by clicking the ellipsis in the Multi-Protocol Gateway page.
- __2. Drag a decrypt action in front of the verify signature action.



- __3. Double click the decrypt action to provide additional details for its configuration.
- __4. In the *Decrypt Key* dropdown, select **ProductServiceCryptoKey**.
- __5. Click **Done**.

- __6. In the list of configured rules at the bottom of the policy editor, click on the **Server to Client** rule to make it the active rule in the editor.
- __7. Drag an Encrypt action to the right of the sign action.



- __8. Double click the encrypt action to complete its configuration.
- __9. In the *Configure Encrypt Action* form, locate the *Recipient Certificate* field, then select **SoapUICryptoCert**.
- __10. Click the **Done** button.
- __11. Click the **Apply Policy** button in the policy editor.
- __12. Click the [Close Window](#) link in the upper right of the policy editor.
- __13. In soapUI, click the green submit button to test your service gateway.

If you inspect the response closely, you'll notice that the contents of the SOAP Body are completely encrypted.

```
<soapenv:Body wsu:Id="Body-1566d2d5-a6f4-4fe9-ba42-242edc2b40d1" xmlns:wsu="http://www.w3.org/2001/XMLSchema-instance" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <xenc:EncryptedData Id="body" Type="http://www.w3.org/2001/04/xmlenc#Data" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#RSA-OAEP" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:CipherData>
        <xenc:CipherValue>evx0ace6Dozasay4Bpoj1BjbHwqj7J/RG/myHcb9Qs
      </xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </soapenv:Body>
```

You just verified that DataPower is signing and encrypting the response. Now configure soapUI to sign+encrypt the request, and decrypt+verify the response.

- __14. In soapUI, click on the Request tab to make it the active tab.
- __15. Click the **Aut** button to show the authorization security settings. If the soapUI log is still visible, you may want to hide that too.
- __16. For *Outgoing WSS*, select: **SignAndEncrypt**

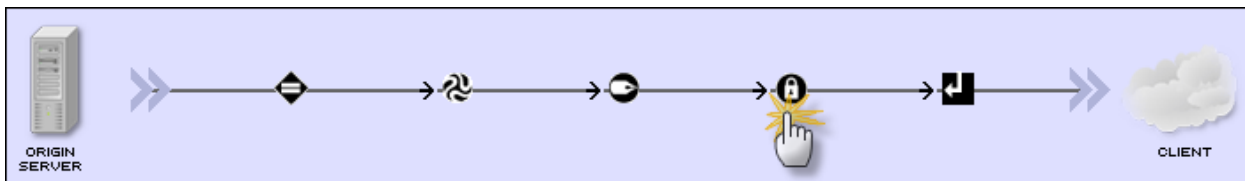
- __17. For *Incoming WSS*, select: **DecryptAndVerify**
- __18. Click the **Aut** button to hide the authorization security settings.
- __19. Click the green submit button to submit the request to ProductServiceProxy. Look closely at the response. This time, soapUI was able to decrypt the contents of the SOAP body.
- __20. Look back at the probe window; click the **Refresh** button. You should see two transactions in the list. Feel free to inspect the new transactions that include the encryption steps.

3.3.2 Field Level Encryption

In the previous steps, you saw how to encrypt the entire SOAP body. In some circumstances, it may be preferable to encrypt only specific elements.

Now you'll modify the encrypt action so that only the <brand> tag will be encrypted.

- __1. Reopen the policy editor by clicking on the ellipsis button in the *Multi-Protocol Gateway Policy* field.
- __2. In the configured rules section at the bottom, click on the **Server to Client** rule to make it the active rule in the editor.
- __3. Double click the encrypt action to open its configuration settings.



- __4. In the *Message Type* section, choose **Selected Elements (Field-Level)**. When you make this selection, a new field, *Document Crypto Map* will appear.

The Document Crypto Map is used to tell the encrypt action which element(s) are to be encrypted. Since the document will be in XML, the most natural way of selecting the target elements is with XPath expressions. The Document Crypto Map represents a collection of XPath expressions which identify the elements to be encrypted.

- __5. Click the plus (+) button next to the *Document Crypto Map* dropdown.
- __6. For the *Name* field, type: **MyCryptoMap**
- __7. For the *Operation*, make sure **Encrypt (WS-Security)** is selected.
- __8. In the *XPath Expression* field, type: `//*[local-name()='benefits']`
- __9. Click the **Add** button to add this XPath to the list of expressions.
- __10. Click the **Apply** button.

- ___11. Click the **Done** button in the *Configure Encrypt Action* window.
- ___12. Click the **Apply Policy** button in the policy editor.
- ___13. Click the [Close Window](#) link to close the policy editor.
- ___14. In soapUI, click the green submit button to POST the request to the ProductServiceProxy gateway.

Since soapUI is configured to decrypt the message, you won't have much proof that DataPower only encrypted the <benefits> element. You can verify this in two ways:

- ___a. Look at the transaction probe, in the response rule, look at the contents of the OUTPUT context (the last magnifying glass) to see what is being returned to soapUI. There you will see that the <benefits> element has been encrypted but the other elements are not.
 - ___b. Modify soapUI's authentication settings by changing the incoming WSS security to either "verify" or blank. This will prevent soapUI from decrypting the response.
- ___15. Click the **Apply** button in the main Multi-Protocol Gateway configuration form.



Applying changes disables the probe.

In order to preserve system resources, the probe is automatically disabled after pressing the *Apply* button in the main Multi-Protocol Gateway configuration window.

- ___16. If your configuration is working properly, click the Save Config link to persist your configuration to the flash memory.

3.4 Summary

In this lab, you saw a variety of ways in which WebSphere DataPower appliances can help secure data using its cryptographic capabilities. You learned:

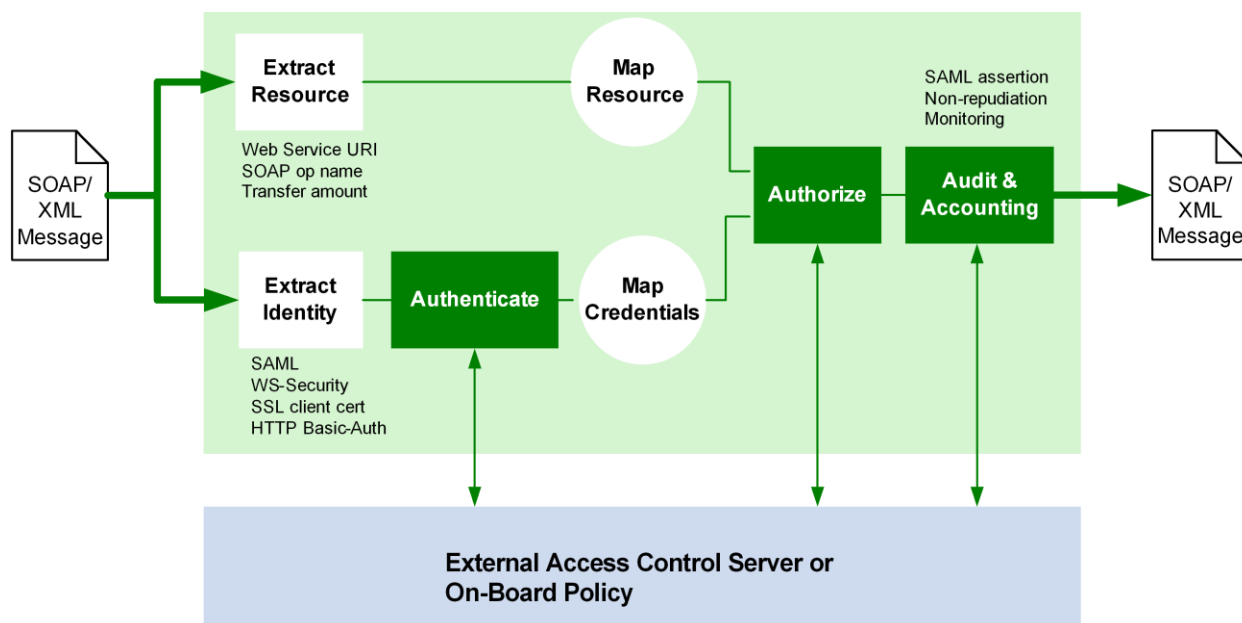
- How crypto certificates and crypto keys are used to dereference key and certificate files for maximum flexibility and ease of maintainability.
- Crypto keys and certificates are used when creating and verifying digital signatures, as well as during encryption and decryption.
- You can add a digital signature to an XML message simply by dragging a sign action onto the processing rule and identifying which key to use.
- Field level, as well as message level encryption and decryption can be performed without sacrificing performance as a result of hardware encryption technology.
- The transaction probe is a powerful tool that allows you to visually inspect every aspect of a transaction, helping to identify configuration or communication problems.

Lab 4 Access Control Framework

Prerequisites: This lab requires the completion of labs 1 through 3.

Up until now, you've seen how WebSphere DataPower can protect XML Web traffic using built-in XML threat protection, digital signatures, and encryption. This lab will introduce the access control framework which provides authentication, authorization, and audit services. Collectively, this is referred to as AAA.

An AAA policy identifies a set of resources and procedures used to determine whether or not a requesting client is granted access to a specific service, file, or document. AAA policies are thus filters in that they accept or deny a specific client request. Basic AAA processing is depicted in the figure below.



4.1 Extract identity & extract resource

The first action that occurs is to extract the claimed identity of the service requester and the requested resource from an incoming message and its protocol envelope. WebSphere DataPower appliances provide an extensive list of predefined identity and resource extraction methods. For example, the identity can be based on IP address, account name/password, SAML assertion, or other criteria, while the requested resource can be specified by (among others) an HTTP URL, a namespace, or a WSDL method.

4.2 Authenticate

If the identity is successfully extracted from the message, it will then be authenticated. Authentication is most commonly accomplished via an external service such as Tivoli Access Manager or LDAP. If the authentication is successful, the process enters the resource and credential mapping phase.

4.3 Credential and resource mapping

Successful server-based authentication generates a set of credentials attesting to the service requester's identity. These credentials can then be mapped into another set more appropriate for the authorization method selected. In addition, the extracted resource name can also be optionally mapped to something more appropriate for the authorization method selected.

The resulting credentials, along with the resulting resource name, form the basis for client authorization, which determines if the now identified client has access to the requested resource.

4.4 Authorize

Like authentication, authorization is most commonly accomplished via an external policy server such as Tivoli Access Manager or an LDAP. The result of the authorization phase is to either allow or deny the request to proceed.

If either authentication or authorization denies access, the system generates an error which is returned to the calling entity. This error may be handled, as with any other errors within multi-step processing, by an on-error action or an error rule. Either method allows for the creation of custom error messages.

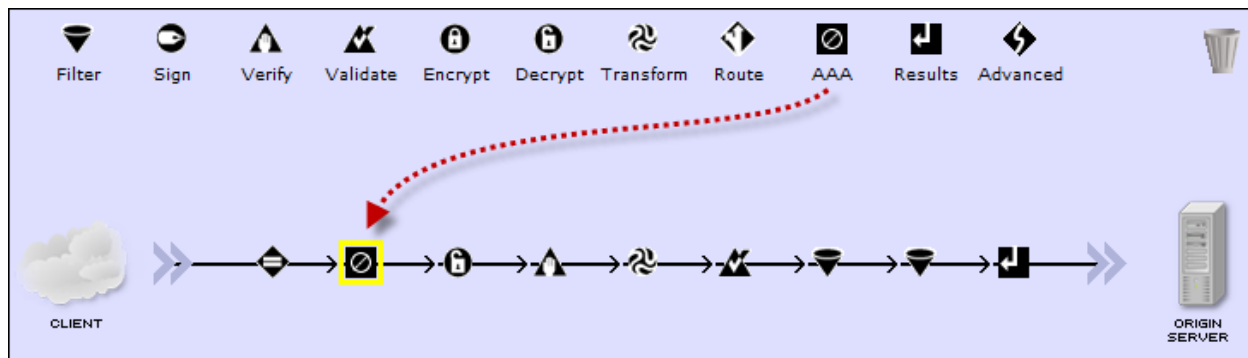
4.5 Audit & accounting

The final phase of the AAA policy performs auditing and security mediation tasks such as converting between WS-Security UsernameToken element and Kerberos/SPNEGO. This phase has the ability to generate various types of security tokens, including Kerberos/SPNEGO, LTPA, and SAML assertions. A stylesheet can also be identified for execution to do any custom auditing tasks.

4.6 LDAP authentication

In this section, you'll add an AAA action to your processing rule to authenticate requests against an LDAP.

- ___1. Re-open the policy editor by clicking on the ellipsis button in the *Multi-Protocol Gateway Policy* field.
- ___2. In the **Client-to-Server** rule, drag an AAA action and drop it after the initial match action.



- __3. Double click the yellow outlined AAA action to configure it.
- __4. The AAA processing action references an *AAA Policy*. Click the plus (+) sign next to the *AAA Policy* dropdown to create a new AAA policy.
- __5. For the *AAA Policy Name*, type: `MyAaaPolicy`
- __6. Click the **Create** button.

The next page identifies how to extract the user's identity (and optionally password) from the message. For this exercise, we'll indicate that the identity will be in a WS-Security Username Token element.

- __7. Select: **Password-carrying UsernameToken Element from WS-Security Header.**
- __8. Click the **Next** button.

Now you'll identify how to authenticate the user.

- __9. Select: **Bind to Specified LDAP Server.** When you make the selection, LDAP specific configuration parameters will be displayed.
- __10. In the *Host* field, type: `demosever`
- __11. Change the *LDAP version* to: **v3**
- __12. In the *LDAP Suffix* field, carefully type: `ou=members,ou=datapower,dc=ibmdemo,dc=com`
- __13. Click the **Next** button.

Now you will define how to extract the resource. Since the message is a SOAP request, you can expect that the first element in the SOAP body contains the operation being requested. In XPath terms, this is referred to as the *Local Name of the Request Element*.

- __14. In the *Extract Resource* form, check: **Local Name of Request Element**
- __15. Click the **Next** button.
- __16. For the authorization phase, leave the default set to: **Allow Any Authenticated Client.**
- __17. Click the **Next** button.

The last page of the AAA policy configuration wizard gives you the options of performing various post processing tasks. One powerful post-processing task is to perform security protocol mediation such as creating a Kerberos/SPNEGO token or generating a signed SAML assertion. For this lab, just leave everything with the default values.

- __18. Click the **Commit** button to save the new AAA policy.
- __19. Click the **Done** button to dismiss the success window.
- __20. Make sure **MyAaaPolicy** is selected in the *AAA Policy* field, and then click **Done**.

- __21. In the policy editor, click the **Apply Policy** button.
- __22. Click the [Close Window](#) link to close the policy editor.
- __23. Click the **Apply** button in the Multi-Protocol Gateway configuration page.
- __24. In the soapUI, click the green submit button to submit the request to your service. The request should now fail with an error message of “Rejected by policy.” To resolve this, you need to tell soapUI to include the WS-Security Usenametoken.
- __25. In soapUI, make sure the request tab is selected then click the **Aut** button to show the authentication settings.
- __26. Change the *Outgoing WSS* dropdown to be **UsrtokenSignEncrypt**.
- __27. Click the **Aut** button to close the authentication settings.
- __28. Click the green submit button to submit another request to the service. This time, the request should succeed. If you want to see the WS-Security UsernameToken that soapUI injected into the message, you can use the probe to inspect the transaction.
- __29. If you want to see the AAA policy reject a bad password, follow these steps:
 - __a. In the soapUI request tab, click on the **Aut** button to show the authentication settings.
 - __b. Select BadPassword_UNT from the *Outgoing WSS* dropdown – the password is incorrect for user david.
 - __c. Click the **Aut** button to hide the authentication settings.
 - __d. Click the green submit button to submit the request. The request should fail with the “Rejected by policy” message.
 - __e. Restore the *Outgoing WSS* setting back to **UsrtokenSignEncrypt** and verify that the request succeeds (this is important for future labs).
- __30. If your configuration is working properly, you can click the [Save Config](#) link in the upper right part of the DataPower banner. This will save your configuration to the flash memory.

4.7 Summary

In this lab, you saw how you can further secure your services with DataPower’s access control framework. You learned:

- Access Control Policies, also known as AAA policies, are a powerful and flexible way to prevent unauthorized access to your services. Through the point-and-click WebGUI, you can easily configure access policies to contact external authentication and policy servers.
- AAA policies can also do security mediation, such as converting between HTTP Basic Authentication and Kerberos/SPNEGO. AAA policies can also create a SAML assertion based on authenticated credentials extracted from the message.

Appendix A. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have

been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Appendix B. Trademarks and copyrights

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

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	
Rational	Redbooks	Red Brick	RequisitePro	System i	
<i>System z</i>	<i>Tivoli</i>	<i>WebSphere</i>	<i>Workplace</i>	<i>System p</i>	

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. See Java Guidelines

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

ITIL is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product and service names may be trademarks or service marks of others.

NOTES

[illegible]

NOTES

[illegible]



© Copyright IBM Corporation 2011.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others.



Please Recycle
